

ITIS P. PALEOCAPA
INFORMATICA E TELECOMUNICAZIONI

Analisi di Malware

REVERSE ENGINEERING DI MALWARE E REALIZZAZIONE DI UNA SIGNATURE

Pape Alpha Toure

Esame di Stato 2017

Abstract

L'obiettivo del lavoro è quello di proporre una Signature, un insieme di regole identificative di software malevoli (malware) o loro varianti. Il caso di studio prende in esame un banking trojan di nome Wirenet, un malware che si presenta all'utente come adattatore per Wi-Fi. Esso è dotato di numerose funzionalità che vanno dal furto di credenziali al controllo remoto della macchina infetta. Per analizzare Wirenet sono state utilizzate tecniche di reverse engineering. Si pone attenzione sulle peculiarità riscontrate nella sua implementazione e su un possibile profilo degli autori.

Indice

Introduzione	1
Reverse Engineering	1
I limiti dell'automazione nell' analisi di malware	2
Preparativi	3
Analisi del Malware	4
Vettori di infezione	4
Installazione	6
Keylogger	10
Architettura	12
Applicazioni bersagliate	15
Remote Administration Tools	17
George Orwell on Surveillance	18
Signature-Based Detection in YARA	19
Conclusioni	21
L'ultima fase	21
Pubblicazione del lavoro	21

Introduzione

Reverse Engineering

L'ingegneria inversa (Reverse Engineering, da ora in poi *RE*) è il processo che permette l'acquisizione di informazioni riguardo ad un sistema, disassemblandolo ed analizzando le sue parti. Le sue modalità variano ampiamente in base al dominio entro cui essa è applicata, ed agli obiettivi che si intendono raggiungere. Questi sono alcuni dei campi dell'informatica che vedono protagonista il RE:

- **Interoperabilità con software Legacy:** alcune aziende potrebbero avere la necessità di utilizzare software datato, la cui documentazione potrebbe non essere disponibile;
- **Formati di file proprietari:** lo sviluppo di alternative open source in grado di interpretare file dal formato proprietario, richiede una conoscenza della loro struttura;
- **Analisi di malware:** fornisce informazioni riguardanti funzionalità, vettori di attacco e metodi di contenimento;
- **Analisi di antivirus:** per ideare tecniche di elusione e verificare l'efficacia dei propri malware;
- **Algoritmi crittografici:** seppur sicuri teoricamente, le implementazioni di algoritmi crittografici possono presentare vulnerabilità.

I limiti dell'automazione nell'analisi di malware

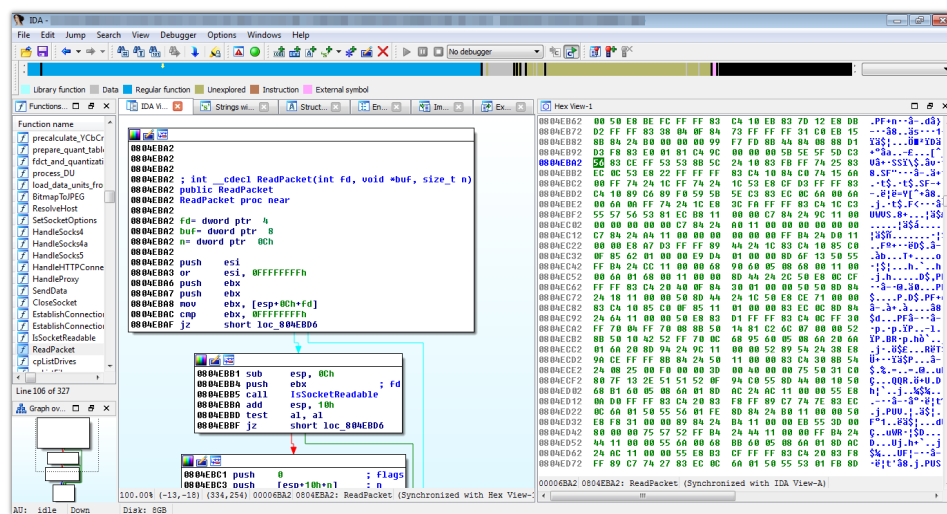
Per automatizzare l'analisi di file vengono impiegate delle sandbox: dei meccanismi, cioè, di suddivisione dell'ambiente di esecuzione programmi. Le sandbox sono spesso utilizzate per ottenere rapidamente informazioni su file sconosciuti, ma non sono totalmente affidabili a causa dei seguenti limiti:

- Possono essere facilmente rilevate dai malware, i quali modificherebbero i propri comportamenti o smetterebbero di funzionare del tutto. Apparentemente potrebbe sembrare una soluzione al problema ma, di fatto non lo è perchè impedisce all'analista di comprendere i meccanismi di funzionamento del malware;
- Non sono in grado di eseguire librerie che necessitano di file eseguibili da cui essere invocate;
- Non forniscono descrizioni approfondite delle funzionalità del malware;
- Non sono necessariamente in grado di fornire l'interazione necessaria per la corretta esecuzione del file;
- Sono lente e dispendiose e per questo vengono raramente adottate in soluzioni per utenti finali.

Preparativi

Per questo studio è stato fatto uso di due macchine virtuali e software per analizzare in modo sicuro il malware.

Windows 7: Oltre ad eliminare accidentali esecuzioni, dispone di una versione Freeware di IDA, un disassembler, ovvero un programma in grado di tradurre il linguaggio macchina in codice assembly. E' lo standard di fatto più utilizzato tra i ricercatori di malware.



Ubuntu/Linux 14 LTS: Questa macchina virtuale è stata dedicata prevalentemente ad analisi di tipo dinamico

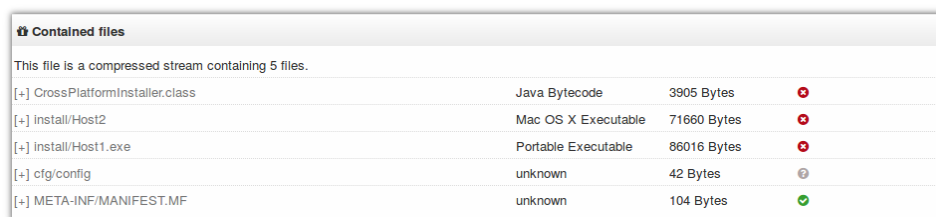
- **Binwalk:** è strumento utilizzato prevalentemente per l'analisi ed il reverse engineering di immagini di firmware;
- **YARA:** è strumento in grado di analizzare e classificare file. Si possono creare descrizioni di famiglie di malware basate su pattern in comune.

Analisi del Malware

Vettori di infezione

Del malware analizzato in questo studio non si è determinato con certezza il principale vettore di infezione nè il contesto in cui è stato utilizzato. Si suppone quindi che sia un malware multi stage, che utilizza un archivio Java .jar (nome del file java.PDF.jar) per raccogliere informazioni riguardo all'host installare ed eseguire la corretta versione del malware.

Non avendo avuto alcuna possibilità di ottenere il dropper (cioè un programma creato per installare un malware su un sistema) si è cercato di ricostruire la prima fase del ciclo di vita del malware partendo dalle informazioni ottenute da un'analisi comportamentale della sandbox di Virus Total.



The screenshot shows a window titled 'Contained files' with a list of files extracted from a compressed stream. The files are: CrossPlatformInstaller.class (Java Bytecode, 3905 Bytes), install/Host2 (Mac OS X Executable, 71660 Bytes), install/Host1.exe (Portable Executable, 86016 Bytes), cfg/config (unknown, 42 Bytes), and META-INF/MANIFEST.MF (unknown, 104 Bytes). Each file has a status icon in the rightmost column: red 'x' for the first three, a question mark for the fourth, and a green checkmark for the fifth.

Contained files			
This file is a compressed stream containing 5 files.			
[+] CrossPlatformInstaller.class	Java Bytecode	3905 Bytes	✗
[+] install/Host2	Mac OS X Executable	71660 Bytes	✗
[+] install/Host1.exe	Portable Executable	86016 Bytes	✗
[+] cfg/config	unknown	42 Bytes	?
[+] META-INF/MANIFEST.MF	unknown	104 Bytes	✓

I nomi dei file, i loro tipi ed i rispettivi hash sono più che sufficienti per poter affermare con certezza che si tratta della payload della prima fase dell'attacco in quanto gli archivi Java quando aperti vengono automaticamente eseguiti. L'assenza della versione Linux del malware all'interno dell'archivio fa sorgere un dubbio in relazione alla provenienza sua infatti Host1 e Host2 sono le versioni Windows e OSX rispettivamente.

La ricostruzione del possibile attacco è basata sull'assenza dei seguenti componenti fondamentali:

-
- Meccanismi di autopropagazione
 - Meccanismi di occultamento
 - Exploit⁽¹⁾

Di seguito sono illustrate due possibili dinamiche attraverso cui avvenivano le infezioni nel 2012:

Email: sono ancora il vettore di attacco più efficace utilizzato dai criminali informatici. Lo spearphishing consiste in un attacco di ingegneria sociale⁽²⁾ mirato a specifici individui oppure ad organizzazioni. Esso si distingue dallo spam di email nel far largo uso di informazioni riguardanti i bersagli raccolte online per il successo dell'attacco.

Exploit Kit: un exploit kit è un insieme di programmi realizzati dai criminali per ovviare alla altrimenti lenta e complessa distribuzione di malware. Essi automatizzano il processo, individuando vulnerabilità nei client e sfruttandole per installare silenziosamente dei malware. Poiché exploit kit semplice può costare 500 dollari al mese, si ipotizza che gli autori del malware non ne avessero uno a disposizione.

(1). Un exploit è un termine usato in informatica per identificare una tipologia di script che sfruttando una specifica vulnerabilità presente in un sistema informatico, permette l'esecuzione di codice malevolo su di esso con lo scopo di far ottenere all'attaccante l'acquisizione dei privilegi amministrativi.

(2). Nel campo della sicurezza informatica, l'ingegneria sociale è lo studio del comportamento individuale di una persona al fine di carpire informazioni utili.

Installazione

Dopo essere stato scaricato ed avviato sulla macchina vittima, Wirenet esegue le seguenti istruzioni:

```
1 .text:0804CCE4 call    InitAESTables
2 .text:0804CCE9 lea     ebp, [esp+0C2Ch+var_C26]
3 .text:0804CCED call    InitTransfersList
4 .text:0804CCF2 call    ReadSettings
5 .text:0804CCF7 call    InstallPath
```

Si ponga l'attenzione sulle funzioni chiamate:

InitAESTables: si limita ad inizializzare le P-BOX e le S-BOX dell'AES
ReadSettings: questa funzione recupera la configurazione default, composta da stringhe cifrate in RC4 all'interno del file stesso. Questo rende più complicata la scrittura di una firma che identifichi una famiglia malware in quanto le stesse informazioni, cifrate con chiavi diverse genererebbero risultati diversi. Questo non significa che le informazioni non possano però essere recuperate, in quanto la chiave è presente nel file. Utilizzando Binwalk si possono ottenere gli offset a cui si trovano le diverse sequenze di byte.

```
$ binwalk -R "\xc7\xd6\xac" wirenet
```

DECIMAL	HEXADECIMAL	DESCRIPTION

62682	0xF4DA	\xc7\xd6\xac

Ripetendo la procedura per la chiave e tutte le stringhe contenute nella funzione `DecryptSettings`, si ottengono tutti gli ingredienti necessari per scrivere un programma in grado di decifrare le stringhe e restituire quanto segue

```
$ python src/decode_settings.py wirenet
ConnectionString: 212.7.208.65:4141;
ProxyString: -
Password: sm0k4s523syst3m523
HostId: LINUX
MutexName: vJEewiWD
InstallPath: %home%/WIFIADAPT
StartupKeyName1: WIFIADAPTER
StartupKeyName2: -
KeyLoggerFileName: %Home%\m8d.dat
BoolSettingsByte: 237
ConnectionType: 001
```

InstallPath: La funzione attraverso `/proc`⁽³⁾ ottiene il percorso a Wirenet si copia in `~/WIFIADAPT` e si esegue. Per garantire una persistenza sul sistema, viene creato un file `~/config/autostart.desktop` per avviare il malware all'avvio del sistema.

```
[Desktop Entry]
Type=Application
Exec=%s
```

(3). Nei sistemi operativi Unix-like `procfs` è uno pseudo-filesystem usato per accedere alle informazioni relative ai processi fornite dal kernel. Il filesystem si trova solitamente montato nella directory `/proc`; poiché non è un filesystem reale, esso non occupa spazio sul disco rigido ed una limitata quantità di memoria.

Hidden=false

Name=%s

E' stato correttamente previsto il caso in cui non fossero installati Desktop Environments e che quindi il file di configurazione appena illustrato non venisse utilizzato. Si è ovviato inserendo nel file `~/.xinitrc` un comando apposito, eseguito all'avvio del server X dal programma `xinit`.

filename&

L'aggiunta di un `'&'` avvia il programma in background.

Si rilevano due grandi mancanze: l'aver eliminato la prima versione del file approdata sul sistema e la mancata gestione dei segnali ricevuti dal programma. Mirai⁽⁴⁾ ha correttamente affrontato entrambe le problematiche

```
1 unlink(args[0]);  
2 [...]
```

La prima istruzione ad essere eseguita è un `unlink` che permette di eliminare il file di nome specificato come argomento di funzione, in questo caso se stesso. La persistenza fisica dei file su disco rende molto più semplice il lavoro dei ricercatori, in quanto fintantochè non si riesce ad ottenere una versione del malware non lo si può analizzare per creare firme.

(4). Mirai è un malware che trasforma i sistemi informatici in botnet controllabili da remoto, le quali possono essere utilizzate in attacchi informatici su larga scala.

Si osservi ancora una volta la premura che Mirai dedica nel impedire che qualunque altro programma interferisca con se stesso

```
1 // Signal based control flow
2 sigemptyset(&sigs);
3 sigaddset(&sigs, SIGINT);
4 sigprocmask(SIG_BLOCK, &sigs, NULL);
5 signal(SIGCHLD, SIG_IGN);
6 signal(SIGTRAP, &anti_gdb_entry);
```

La chiamata `signal()` assegna ad un segnale, uno specifico handler. Questo permette ad ogni programma di ridefinire i propri comportamenti in risposta a specifici segnali. `SIGTRAP` è il segnale utilizzato da un processo per poter fermare l'esecuzione di un altro.

In sistemi UNIX la chiamata di sistema `ptrace()` permette ad un processo di controllarne un altro, costituendo quindi la base di tutti i debugger. Più specificatamente, in architetture che supportano breakpoints, nel momento in cui la CPU esegue un'istruzione che genera un interrupt, l'esecuzione viene affidata all'handler corrispondente nel IDT⁽⁵⁾. Nel momento in cui la CPU riceve un interrupt, essa passa in kernel-mode ed infine notifica il debugger del breakpoint raggiunto dal programma debuggato.

(5). La Interrupt Descriptor Table (IDT) è una struttura dati usata dalle architetture x86 per implementare una tabella di vettori interrupt. La IDT è usata dal processore per determinare la corretta risposta a interrupt e eccezioni.

Durante l'esecuzione di InstallHost oltre all'opzione di essere eseguito come demone, ne segue una caratteristica dei banking trojan, ovvero un keylogger

```
1 .text:08052D95 push    esi
2 .text:08052D96 push    esi
3 .text:08052D97 push    0          ; arg
4 .text:08052D99 push    offset cpStartKeyLogger ; start_routine
5 .text:08052D9E call     cpBeginThread
```

Keylogger

Il keylogger utilizza `XOpenDisplay()` per connettersi al server X⁽⁶⁾ del sistema, dopo di che usa `XQueryExtension()` per sapere se l'estensione del protocollo X che gestisce gli input è presente ("`XInputExtension`"). Con questo ultimo handle all'estensione, sfrutta `XListInputDevices` per ottenere la lista dei dispositivi connessi ricercandone due specifici di nome `Systemkeyboard` e `AT`. A questo punto con `XOpenDevice()` si apre il dispositivo appena individuato e si selezionano gli eventi da intercettare con `XSelectExtensionEvent()`. Da da questo momento in poi, tutti i tasti premuti vengono salvati nel file `~/.m8d.dat`. Ricostruendo le strutture dati utilizzate si decompila parte del keylogger nel seguente modo:

(6). X Window System è un gestore grafico, standard de-facto per molti sistemi Unix-like.

```
1 XKeyEvent *hooks;
2
3 [...]
4
5 while (TRUE) {
6     // Gets an event from the 'XEvent' queue
7     // Intercept the event
8     XNextEvent (&display, &single_event);
9
10    // Acquire its information
11    hooks.type = single_event.type;
12    hooks.display = single_event.xcreatewindow.display;
13    hooks.root = single_event.xproperty.time;
14    hooks.time = single_event.xkeymap.key_vector[12];
15    hooks.y = single_event.pad[10]
16    hooks.y_root = single_event.pad[12]
17    hooks.keycode = single_event.pad[14]
18
19    // Save it to a file
20    LogKey (hooks, hooks, &display);
21 }
22
23 XCloseDisplay ();
24 [...]
```

Il codice del keylogger viene eseguito all'interno di un processo user-space, un processo che non appartiene al sistema operativo. I keylogger in kernel-space possono essere implementati come driver che registrano gli eventi ed in quanto tali possono facilmente bypassare analisi di applicazione in user-space.

Architettura

L'architettura di rete è molto semplice: gli host infetti stabiliscono una connessione con il server di controllo detto Command and Control (C&C), eventualmente attraverso server SOCKS⁽⁷⁾ aperti da altre macchine infette.

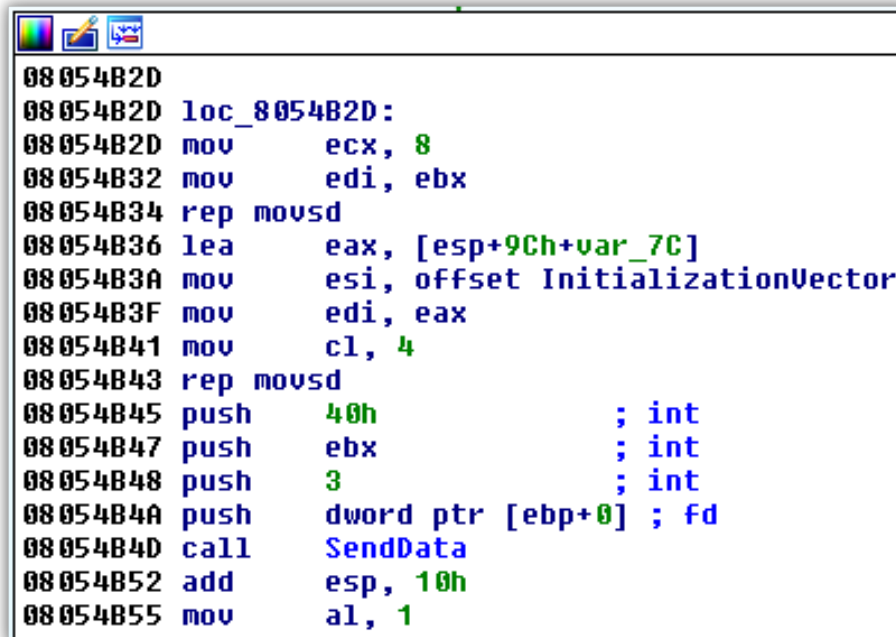
Wirenet invia un pacchetto contenente RGI28DQ30QB8Q1F7, una stringa di autenticazione. Le comunicazioni vengono rese sicure tramite un implementazione dell' AES in modalità CFB⁽⁸⁾ utilizzando sm0k4s523syst3m523 come password. I dettagli specifici riguardanti le modalità di funzionamento dei cifrari a blocchi non sono utili ai fini di questo lavoro e per tale ragione sono stati omessi. Dopo la generazione di numeri casuali viene cifrato la stringa di autenticazione

```
.text:08054B02 push    offset TestPacket
.text:08054B07 push    offset InitializationVector
.text:08054B0C push    10h
.text:08054B0E push    1
.text:08054B10 push    eax
.text:08054B11 call    AESCryptCFB
```

(7). Un server SOCKS è un particolare tipo di proxy che permette di effettuare connessioni TCP dirette (e, dalla versione 5, di veicolare traffico UDP oltre che TCP) tra computer su due reti IP differenti nei casi in cui un instradamento diretto (routing) non sia disponibile

(8). In crittografia la modalità di funzionamento dei cifrari a blocchi è una serie di procedimenti standard di un algoritmo che operi tramite cifratura a blocchi per garantire la sicurezza di testi o messaggi di lunghezza a piacere.

In caso AESCryptCFB avvenga con successo, viene utilizzata SendData per trasmettere i dati cifrati al server C&C



```
00054B2D
00054B2D loc_8054B2D:
00054B2D mov     ecx, 8
00054B32 mov     edi, ebx
00054B34 rep movsd
00054B36 lea     eax, [esp+9Ch+var_7C]
00054B3A mov     esi, offset InitializationVector
00054B3F mov     edi, eax
00054B41 mov     cl, 4
00054B43 rep movsd
00054B45 push    40h           ; int
00054B47 push    ebx          ; int
00054B48 push    3             ; int
00054B4A push    dword ptr [ebp+0] ; fd
00054B4D call    SendData
00054B52 add     esp, 10h
00054B55 mov     al, 1
```

La dispendiosa analisi dell'AES non ha prodotto risultati poiché l'implementazione non presenta bug rilevanti sfruttabili per un exploit. L'utilizzo di un timestamp per generare gli Initialization Vectors ovvero i numeri pseudo-casuali utilizzati dal Cipher Feedback, viola uno dei requisiti richiesti per mantenere l'integrità del CFB ovvero la non prevedibilità del IV. Lo studio delle implementazioni degli algoritmi crittografici trova una più ampia applicazione nei ransomware, per trovare modi di recuperare file cifrati.

Dopo aver controllato che la dimensione dei pacchetti ricevuti e il tipo siano validi viene invocato l'handler dei comandi, il nucleo del malware

```
1 .text:0804CDAA dec     ebx
2 .text:0804CDAB push    ebx                ; int
3 .text:0804CDAC lea     eax, [esp+0C30h+filename]
4 .text:0804CDB0 push    eax                ; filename
5 .text:0804CDB1 push    edi                ; int
6 .text:0804CDB2 push    esi                ; arg
7 .text:0804CDB3 call    ProcessData
```

Questa funzione (ProcessData) è una delle più grandi all'interno del file. E' composta da uno switch case in grado di gestire 64 diverse casi, ognuna delle quali corrisponde ad una funzioni del malware.

Applicazioni bersagliate

Il malware è stato progettato per rubare le credenziali utilizzate in una grande varietà di prodotti, tutti comunemente utilizzati come browser e client per email. I prodotti Mozilla condividono lo stesso meccanismo per salvare email e password: utilizzano un database SQLite criptato di nome `key3.db`. Per ottenere i dati utilizzati in Firefox per esempio, è necessario conoscere i vari profili presenti, e ripetere il processo di estrazione per ognuno di essi. Tramite espressioni regolari `firefox-3*`, `firefox-4*`, `thunderbird-*` vengono cercate in `/usr/lib` le cartelle contenenti le rispettive librerie necessarie per l'interazione con il database a seconda del servizio. Firefox, Thunderbird e Seamonkey usano `NetworkSecurityServices`, un insieme di librerie in grado di fornire servizi crittografici. Dopo aver caricato le librerie necessarie, si è in grado di accedere alle credenziali normalmente sfruttando la stessa identica procedura utilizzata da programmi legittimi. Non viene però gestito il caso in cui un profilo adotti una master password, ovvero una password con cui vengono cifrate tutte le credenziali. La query al database viene decompilata come segue:

```
1 // Connect to the database
2 sqlite3_open("...", db_handle);
3
4 // Prepare the SQL statement
5 sqlite3_prepare_v2 (database,
6                     "select * from moz_logins",
7                     25,
8                     stmt,
```

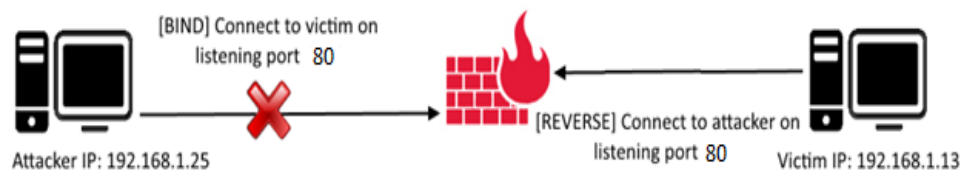
```
9             ...);
10 // Initialize a NSS object
11 NSS_Init (...);
12
13 // Get a Slot ID, just an ID
14 keySlot = PK11_GetInternalKeySlot ();
15
16 PK11_Authenticate (keySlot);
17
18 // Get the return values of the previous query
19 sqlite3_column_text (stmt, ...);
20
21 // Data is base64encoded
22 NSSBase64_DecodeBuffer ();
23
24 // A blank password is used if no master password is set
25 PK11SDR_Decrypt ();
26
27 // See if the query was executed successfully
28 sqlite3_step ();
```

Oltre a Thunderbird, Firefox, e SeaMonkey le seguenti applicazioni sono vulnerabili ad un attacco simile a quello appena illustrato

- Opera
- Google Chrome
- Google Chromium
- Pidgin

Remote Administration Tools

A differenza dei classici banking trojan gli autori di Wirenet hanno investito notevolmente nell'implementazione di funzionalità atte alla sorveglianza dell'utente come il controllo del mouse, delle finestre e catture dello schermo. Non sono state tralasciate funzionalità di amministrazione remota come la navigazione nei drive, i download, la rimozione di file e la terminazione dei processi. In fine è presente una classica reverse bind shell: una shell locale i cui descrittori di file vengono duplicati ed assegnati al socket su cui è in corso la comunicazione. Questa procedura permette di fornire terminali agli host della rete anche attraverso eventuali firewall.



George Orwell on Surveillance

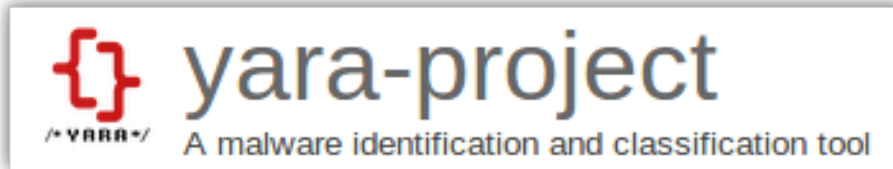
"Those who do not move, do not notice their chains."

- Rosa Luxemburg,

George Orwell's 1984 showcases the life of Winston, a man living in a country ruled by a totalitarian government. The Party, the name of the government ruled by a mysterious figure called Big Brother, truly stretches the meaning of totalitarianism because of the means by which it controls people. It does so by employing a wide variety of methods most notably the constant audio video surveillance of people's houses. Moreover, it makes use of an authority called Thought Police entitled of finding anyone who commits a Thoughtcrime, namely an illegal thought. On top of using an aggressive propaganda and actively destroying citizens' critical thinking skills, the Party has adopted Newspeak: a simplified version of English that removes concepts entirely, thus making certain thoughts impossible.

Edward Snowden leaked classified NSA documents, disclosing multiple global surveillance programs run by NSA and the Five Eyes Intelligence Alliance. The threat Orwell was warning us about has already struck: global surveillance programs like PRISM, XKeyScore have gone far beyond their initial scope, national security. The ever-green argument "I have nothing to hide" which is never supported by actions, delivers multiple fallacious beliefs one of them is that only bad people have things to hide, consequentially making us only more and more willing to be compliant to whatever expectation has to be met. As Orwell showed, privacy has never been a matter of hiding malicious things, as it has always been about freedom.

Signature-Based Detection con YARA



*The pattern matching swiss knife for malware researchers
(and everyone else)*

- Virus Total, *Documentation*

Yara è un programma utilizzato principalmente nell'analisi di malware nel loro rilevamento. Utilizza un approccio basato su regole che descrivono famiglie di malware. Ogni regola consiste in un pattern testuale oppure binario da ricercare all'interno del file. Si considerino le seguenti istruzioni e si cerchi di scrivere una regola in Yara per identificarle.

```
1          sub_100012BC proc near
2  0F 20 C0      mov     eax, cr0
3  25 FF FF FE FF and     eax, 0FFFFFFFh
4  0F 22 C0      mov     cr0, eax
5  C3           retn
6          sub_100012BC endp
```

Nell'architettura x86 il registro CR0 contiene diversi flag che modificano il comportamento della CPU quali la paginazione, la cache e la modalità protetta. Il bit più interessante è il 16esimo che, se non

settato, permette la scrittura su pagine normalmente read-only. Questo è un componente chiave per i rootkit⁽⁹⁾ che hanno bisogno di applicare modifiche al sistema operativo perchè rimangano nascosti.

Di seguito si descrive una regola che segnala tutti i file che presentano le stringhe corrispondenti alle operazioni che interagiscono con elementi critici del sistema: x86_rootkit

```
1 rule x86_rootkit {
2
3     strings:
4         $rk1 = { 0F 20 C0 }          // mov     eax, cr0
5         $rk2 = { 25 FF FF FE FF } // and      eax, 0FFFEFFFFh
6         $rk3 = { 0F 22 C0 }          // mov     cr0, eax
7
8     condition:
9         all of them
10 }
```

Per scrivere una buona regola bisogna considerare gli aspetti più difficili da cambiare tra varianti. Il denominatore comune tra i banking trojan è la mappatura dei tasti tipica di keylogger necessaria a salvare su file i vari tasti premuti, un'altra caratteristica comune sono i percorsi dei file appartenenti alle varie applicazioni sulle varie piattaforme.

(9). Il rootkit è una collezione di software che si preoccupano di mascherare sé stessi o altri programmi.

Conclusioni

L'ultima fase

L'ultima ma non meno importante fase del progetto consiste nel rendere disponibile l'analisi effettuata agli sviluppatori di antivirus, in modo che quest'ultimi possano integrare i propri software con il gli elementi necessari per riconoscere le varianti di questo malware. WannaCry è un ransomware⁽¹⁰⁾ che ha infettato più di 400.000 computer grazie ad ETERNALBLUE, un exploit del NSA che sfrutta una vulnerabilità per cui Microsoft aveva rilasciato una patch due mesi prima, dimostrando che non è sufficiente migliorare misure di sicurezza se queste poi non vengono adottate. Sono ancora numerosi gli Antivirus non in grado di riconoscere Wirenet, sono 14 secondo le API Virus Total tra cui alcuni di alta qualità come Malwarebytes.

Pubblicazione del lavoro

Il codice decompilato di Wirenet, il file utilizzato, la signature in YARA e questo documento sono disponibili su GitHub sotto la licenza BSD-2 all'indirizzo seguente

- <https://github.com/shxdow/wirenet-analysis>

La versione di Wirenet si può identificare con i seguenti hash

- MD5 9A0E765EECC5433AF3DC726206ECC56E
- SHA1 5996d02c142588b6c1ed850e461845458bd94d17

(10). Un ransomware è un tipo di malware che limita l'accesso del dispositivo che infetta, ne cifra i file e richiede un riscatto per ripristinare l'accesso al computer.

Bibliografia

- [1] Bruce Dang and Alexandre Gazet. *Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation* Feb 17, 2014
- [2] Michael Sikorski and Andrew Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software* by Michael Sikorski and Andrew Honig Mar 3, 2012
- [3] Ryan elfmaster O'Neill. *Learning Linux Binary Analysis* Feb, 2016
- [4] Dennis Yurichev. *Reverse Engineering for Beginners*
- [5] Robert Love. *Linux Kernel Development (3rd Edition)* Jul 2, 2010
- [6] Chris Eagle. *The IDA Pro Book 2nd edition* Jul, 2011
- [7] George Orwell. *1984* Jun 8, 1949