Progetto del corso di Sicurezza informatica e Internet A.A. 2017-2018

Andrea Graziani (0273395)[1], Alessandro Boccini (0277414)[1], and Ricardo Gamucci (0274716)[1]

[1]Università degli Studi di Roma Tor Vergata

27 marzo 2019

Indice

1 Analisi tecnica del malware **2**
1.1 Analisi dei file 2
1.1.1 Analisi del file `2.so` 3
1.1.2 Stringhe stampabili rilevanti 3
1.1.3 Analisi del file `Injection_API_executable_e` 6
1.1.3.1 Disassemblaggio 8

Capitolo 1

Analisi tecnica del malware

1.1 Analisi dei file

Tabella 1.1: Lista dei file facentiff parte del malware FASTCash

Nome	**SHA256**
Lost_File.so	10ac312c8dd02e417dd24d53c99525c29d74dcbc84730351ad7a4e0a4b1a0eba
Unpacked_dump_4a740227eeb82c20...	10ac312c8dd02e417dd24d53c99525c29d74dcbc84730351ad7a4e0a4b1a0eba
Lost_File1_so_file	3a5ba44f140821849de2d82d5a137c3bb5a736130dddb86b296d94e6b421594c
4f67f3e4a7509af1b2b1c6180a03b3...	4a740227eeb82c20286d9c112ef95f0c1380d0e90ffb39fc75c8456db4f60756
5cfa1c2cb430bec721063e3e2d144f...	820ca1903a30516263d630c7c08f2b95f7b65dffceb21129c51c9e21cf9551c6
Unpacked_dump_820ca1903a305162...	9ddacbcd0700dc4b9babcd09ac1cebe23a0035099cb612e6c85ff4dffd087a26
8efaabb7b1700686efedadb7949eba...	a9bc09a17d55fc790568ac864e3885434a43c33834551e027adb1896a463aafc
d0a8e0b685c2ea775a74389973fc92...	ab88f12f0a30b4601dc26dbae57646efb77d5c6382fb25522c529437e5428629
2.so	ca9ab48d293cc84092e8db8f0ca99cb155b30c61d32a1da7cd3687de454fe86c
Injection_API_executable_e	d465637518024262c063f4a82d799a4e40ff3381014972f24ea18bc23c3b27ee
Injection_API_log_generating_s	e03dc5f1447f243cf1f305c58d95000ef4e7dbcc5c4e91154daa5acd83fea9a8
inject_api	f3e521996c85c0cdb2bfb3a0fd91eb03e25ba6feef2ba3a1da844f1b17278dd2

1.1.1 Analisi del file `2.so`

In base all'output ottenuto dal tool unix `file`, `2.so` è un file di tipo **eXtended COFF** (**XCOFF**) che rappresenta la versione migliorata ed estesa del formato **Common Object File Format** (**COFF**), il formato di file standard che ha definito la struttura dei file eseguibili e delle librerie nei sistemi operativi UNIX[1] fino al 1999[2], anno della definitiva adozione dello standard **Executable and Linkable Format** o **ELF**. XCOFF rappresenta tuttavia uno standard proprietario sviluppato da IBM[3] adottato nei sistemi operativi **Advanced Interactive eXecutive** o **AIX**, una famiglia di sistemi operativi proprietari basati su Unix sviluppati dalla stessa IBM.[4]

In accordo alle nostre analisi, confermate anche dal report AR18-275A della NCCIC, il file `file 2.so` rappresenta una **shared library** che esporta una serie di metodi che consentono l'iterazione con i sistemi finanziari che utilizzano il protocollo **ISO8583**.[5]

Tabella 1.2: Dettagli del file `2.s0`

Descrizione	Valore	Comando Unix
Nome	`2.so`	`stat -c "%n" 2.so`
Dimensione (*byte*)	`110592`	`stat -c "%s" 2.so`
Data ultima modifca	`2018-11-09 11:08:40.000000000 +0100`	`stat -c "%y" 2.so`
Tipo di file	`64-bit XCOFF executable or object module`	`file 2.so`
MD5	`b66be2f7c046205b01453951c161e6cc`	`md5sum 2.so`
SHA1	`ec5784548ffb33055d224c184ab2393f47566c7a`	`sha1sum 2.so`
SHA256	`ca9ab48d293cc84092e8db8f0ca99cb1 55b30c61d32a1da7cd3687de454fe86c`	`sha256sum 2.so`
SHA512	`6890dcce36a87b4bb2d71e177f10ba27f517d1a53ab02500296f9b3aac021810 7ced483d70d757a54a5f7489106efa1c1830ef12c93a7f6f240f112c3e90efb5`	`sha512sum 2.so`

1.1.2 Stringhe stampabili rilevanti

Per estrazione di tutte le stringhe stampabili contenute nel file `2.so` ci siamo serviti del tool `strings`[6] di cui riportiamo frammenti dell'output ottenuto nei listati 1.1 e 1.3.

Listing 1.1: Stringe estratte dal file `2.so`

```
465 ...
466 _GLOBAL__FI_eg64_so
467 _GLOBAL__FD_eg64_so
```

[1] Cfr. `https://it.wikipedia.org/wiki/COFF`

[2] Cfr. `https://en.wikipedia.org/wiki/Executable_and_Linkable_Format`

[3] Cfr. IBM - *XCOFF Object File Format* - `https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.files` (data ultima consultazione 27-03-2019)

[4] Cfr. `https://www.ibm.com/it-infrastructure/power/os/aix`

[5] Cfr. The National Cybersecurity and Communications Integration Center's (NCCIC), *Malware Analysis Report (AR18-275A)* - 2 Ottobre 2018 - `https://www.us-cert.gov/ncas/analysis-reports/AR18-275A)`

[6] Cfr. `https://linux.die.net/man/1/strings`

```
468 =s4m
469 /opt/freeware/lib/gcc/powerpc-ibm-aix6.1.0.0/4.2.0/ppc64:/
        opt/freeware/lib/gcc/powerpc-ibm-aix6.1.0.0/4.2.0:/opt/
        freeware/lib/gcc/powerpc-ibm-aix6.1.0.0/4.2.0/../../..:/
        usr/lib:/lib
470 libc.a
471 shr_64.o
472 libpthread.a
473 shr_xpg5_64.o
474 ...
```

Poiché nei sistemi operativi AIX la directory all'interno del quale sono contenute tutte le librerie di GCC assume la forma mostrata nel listato 1.2[7], possiamo dedurre dalla riga 496 del listato 1.1 che la versione di GCC utilizzata è stata la 4.2.0 (versione rilasciata il 13 Maggio 2007[8]) mentre la versione del sistema operativo bersaglio fosse stata la V6.1, versione ormai obsoleta del sistema operativo AIX il cui supporto è terminato ufficialmente il 30 Aprile del 2017.[9]

Ovviamente il riferimento alla libreria standard `libc.c` e di GCC suggeriscono che il malware è stato scritto in C/C++.

Listing 1.2: Formato del percorso di installazione delle librerie GCC nei sistemi operativi AIX

```
1 /opt/freeware/lib/gcc/<architecture_AIX_level>/<GCC_Level>
```

Il listato 1.3 mostra ciò che dovrebbero essere i nomi delle procedure esportate dalla libreria il che dimostra in modo inequivocabile il fatto che il malware è in grado di interagire con i sistemi informatici che fanno uso del protocollo ISO8583.

Listing 1.3: Stringhe estratte dal file `2.so`

```
545 ...
546 DL_ISO8583_MSG_Init
547 DL_ISO8583_MSG_Free
548 DL_ISO8583_MSG_SetField_Str
549 DL_ISO8583_MSG_SetField_Bin
550 DL_ISO8583_MSG_RemoveField
551 DL_ISO8583_MSG_HaveField
552 DL_ISO8583_MSG_GetField_Str
553 DL_ISO8583_MSG_GetField_Bin
554 DL_ISO8583_MSG_Pack
555 DL_ISO8583_MSG_Unpack
556 DL_ISO8583_MSG_Dump
557 _DL_ISO8583_MSG_AllocField
558 DL_ISO8583_COMMON_SetHandler
559  DL_ISO8583_DEFS_1987_GetHandler
560  DL_ISO8583_DEFS_1993_GetHandler
561 _DL_ISO8583_FIELD_Pack
```

[7] http://www.perzl.org/aix/index.php%3Fn%3DMain.GCCBinariesVersionNeutral
[8] http://www.gnu.org/software/gcc/gcc-4.2/
[9] https://www-01.ibm.com/support/docview.wss?uid=swg21634678#AIX

```
562 _DL_ISO8583_FIELD_Unpack
563 ...
```

1.1.3 Analisi del file `Injection_API_executable_e`

Il file `Injection_API_executable_e` rappresenta un file di tipo AIX-executable, ovvero un file seguitibile su sistemi oprativi UNIX svilupati da ibm il cui scopo è quello

This file is an AIX (Advanced Interactive Executive) executable, intended for a proprietary UNIX operating system developed by IBM. This application is designed to inject a library into a currently running process

Dalle analisi effettuate dalla National Cybersecurity and Communications Integration Center's (NCCIC), l'istituzione governativa dedicata agli attacchi infromatici, ha stabilito che l'eseguibile scopo di eseguire code injection all'interno di un processo in esecuzione.

Code injection is the exploitation of a computer bug that is caused by processing invalid data. Injection is used by an attacker to introduce (or "inject") code into a vulnerable computer program and change the course of execution. [10]

Tabella 1.3: Dettagli tecnici del file `2.s0`

Descrizione	**Valore**	**Comando Unix**
Nome	2.so	`stat -c "%n" 2.so`
Dimensione (*byte*)	89088	`stat -c "%s" 2.so`
Data ultima modifca	2018-11-09 11:08:40.000000000 +0100	`stat -c "%y" 2.so`
Tipo di file	64-bit XCOFF executable or object module	`file 2.so`
MD5	b3efec620885e6cf5b60f72e66d908a9	`md5sum 2.so`
SHA1	274b0bccb1bfc2731d86782de7babdeece379cf4	`sha1sum 2.so`
SHA256	d465637518024262c063f4a82d799a4e 40ff3381014972f24ea18bc23c3b27ee	`sha256sum 2.so`
SHA512	a36dab1a1bc194b8acc220b23a6e36438d43fc7ac06840daa3d010fddcd9c316 8a6bf314ee13b58163967ab97a91224bfc6ba482466a9515de537d5d1fa6c5f9	`sha512sum 2.so`

Listing 1.4: Stringhe estratte dal file `Injection_API_executable_e` usando il comando `strings -d ./2.so`

```
347 ...
348 ../../../../gcc-4.8.5/libgcc/config/rs6000/cxa_atexit.c
349 @(#)23  1.6  src/bos/usr/ccs/lib/libpthreads/init.c, libpth,
        bos610 6/21/07 15:28:59
350 @(#)61  1.16  src/bos/usr/ccs/lib/libc/__threads_init.c,
       libcthrd, bos61B, b2007_33A0 8/2/07 13:09:21
351 _GLOBAL__FI_eng64
352 _GLOBAL__FD_eng64
353 /opt/freeware/lib/gcc/powerpc-ibm-aix7.1.0.0/4.8.5/ppc64:/
       opt/freeware/lib/gcc/powerpc-ibm-aix7.1.0.0/4.8.5:/opt/
       freeware/lib/gcc/powerpc-ibm-aix7.1.0.0/4.8.5/../../..:/
       usr/lib:/lib
354 libc.a
355 shr_64.o
356 libpthread.a
```

[10] dashttps://www.us-cert.gov/ncas/analysis-reports/AR18-275A

```
357 shr_xpg5_64.o
```

Estraendo tutte le stringhe stampabili attraverso il tool `strings` dal file `Injection_API_executable_e`, possiamo notare come quest'ultimo, a differenza del file `2.so` visto precedentemente, sia stato realizzato utilizzando non solo una differente versione di GCC, la versione GCC 4.8.5 data 23 giugno 2015[11], ma anche è stato compilato per una diversa versione del sistema operativo AIX, AIX V7.1. Sfortunatamente non è possibile risalire echnology Levels (TLs) [12] del sistema operativo, una sorta di aggiornamento che introduce funzionalità. In ogni caso si tratta di un sistema operativo obsoleto: la prima versione è stata rilasciata a settembre 2010 e il supporto è terminato 30 Novembre 2013 tuttavia sono ancora supportate la versione V7.1 TL4 fino a dicembre 2019 mentre la versione V.7.1 TL5 fino ad aprile 2022. [13]

CONGETTURA: Motlo interessanti sono le righe 357 a 359, sono due object file e librerie (/usr/lib/libpthreads.a) e due object file (shr_64.o) (shr_xpg5_64.o),, curiosamente sono due librerie richieste da vmstat is a tool that collects and reports data about your system's memory, swap, and processor resource utilization in real time. It can be used to determine the root cause of performance and issues related to memory use. [14]

```
1  $ ldd /usr/bin/vmstat
2  /usr/bin/vmstat needs:
3           /usr/lib/libc.a(shr_64.o)
4           /usr/lib/libwlm.a(shr_64.o)
5           /usr/lib/libperfstat.a(shr_64.o)
6           /usr/lib/libcorcfg.a(shr_64.o)
7           /unix
8           /usr/lib/libcrypt.a(shr_64.o)
9           /usr/lib/libpthreads.a(shr_xpg5_64.o)
10          /usr/lib/libcfg.a(shr_64.o)
11          /usr/lib/libodm.a(shr_64.o)
12          /usr/lib/liblvm.a(shr_64.o)
13          /usr/lib/libsrc.a(shr_64.o)
```

Listing 1.5: Stringhe estratte dal file `Injection_API_executable_e` usando il comando `strings -d ./2.so`

```
944 ...
945 IBM XL C for AIX, Version 11.1.0.1
946 ...
```

Dalla riga 944 possiamo ricavare l'esatto compilatore utilizzato XL C for AIX, V11.1; Nativamente non supportato dal sistema operativo AIX V7.1[15] ma il supporto è stat oaggiunto successivamente nel stemmbre 2010. [16]

[11] http://www.gnu.org/software/gcc/gcc-4.8/
[12] http://ibmsystemsmag.com/aix/tipstechniques/migration/oslevel_versions/
[13] Cfr: https://www-01.ibm.com/support/docview.wss?uid=isg3T1012517
[14] http://www-01.ibm.com/support/docview.wss?uid=isg3T1023954
[15] https://www-01.ibm.com/support/docview.wss?uid=swg21326972
[16] http://www-01.ibm.com/support/docview.wss?uid=swg1IZ84777

Listing 1.6: Stringhe estratte dal file `Injection_API_executable_e` usando il comando `strings -d ./2.so`

```
...
[main] Inject Start
[main] SAVE REGISTRY
[main] proc_readmemory fail
[main] toc=%llX
[main] path::%s
[main] data(%p)::%s
[main] Exec func(%llX) OK
[main] Exec func(%llX) fail ret=%X
[main] Inject OK(%llX)
[main] Inject fail ret=%llX
[main] Eject OK
[main] Eject fail ret=%llX
Usage: injection pid dll_path mode [handle func toc]
       mode = 0 => Injection
       mode = 1 => Ejection
[main] handle=%llX, func=%llX, toc=%llX
[main] ERROR::g_pid(%X) <= 0
[main] ERROR::load_config fail
[main] ERROR::eject & argc != 7
[main] ERROR::g_dl_handle(%llX) <= 0
[main] WARNING::func_addr(%llX), toc_addr(%llX)
...
```

D notiamo vari format specifiers %llX usati ad esempio dalla funzione printf dove vengono stampati per lo più long long-sized integer argument. Unsigned hexadecimal integer (uppercase)

%[flags][width][.precision][length]specifier

Inoltre pare evidente che queste stampe fanno riferimento alla funzione main.

Tali indizi ci fanno supporre che l'eseguibile era munito di una quancleh meccanismo di logging che riportasse la funzione attuamlemten in esecuzion o un evento .comunicasse agli attaccanti quanto avvenivsse. curiosamente dalle righe ... c'è anche la descrizione dell'utilizzo del file che pare abbia due modalità, injection o ejection.

1.1.3.1 Disassemblaggio

La traduzione dal linguaggio macchina all'assembler del file è stato usufrendo del servizio web https://onlinedisassembler.com/ per motivi di semplicità con le seguenti impostazioni

architettura powerpc620 processore POWER 7 64 bit

Queste impostazioni ci hanno permesso di ottenere un output sostanzialmente identico a quello mostrato in vari screenshot dalla CISA

ftp://public.dhe.ibm.com/systems/power/docs/aix/72/idalangref_pdf.pdf

LI

Specifies a 24-bit signed two's-complement integer that is concatenated on the right with 0b00 and sign-extended to 64 bits (PowerPC®) or 32 bits (POWER® family). This is an immediate field.

https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.alangref/idalangref_inst_fiel

mflr r0 # move LR into GPR0

Tabella 1.4: Dettagli tecnici del file `2.s0`

Comando	SIntassi	Descrizione
`Brach Link (bl)`	bl target_address	Branches to a specified target address.

If a branch instruction has the Link bit set to 1, then the Link Register is altered to store thereturn address for use by an invoked subroutine. The return address is the address of the instructionimmediately following the branch instruction (pag 33)

The following code transfers the execution of the program to here and sets the Link Register:

https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.alangref/idalangref_bbra

Elenco delle figure

Elenco delle tabelle

1.1 Lista dei file facentiff parte del malware FASTCash 2
1.2 Dettagli del file `2.s0` 3
1.3 Dettagli tecnici del file `2.s0` 6
1.4 Dettagli tecnici del file `2.s0` 9

Listings

1.1 Stringe estratte dal file `2.so` 3
1.2 Stringe estratte dal file `2.so` 4
1.3 Stringhe estratte dal file `2.so` usando il comando `strings -d ./2.so` 4
1.4 Stringhe estratte dal file `Injection_API_executable_e` usando il comando `strings -d ./2.so` 6
1.5 Stringhe estratte dal file `Injection_API_executable_e` usando il comando `strings -d ./2.so` 7
1.6 Stringhe estratte dal file `Injection_API_executable_e` usando il comando `strings -d ./2.so` 8