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

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`

1.1.1.1 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
468 =s4m
```

[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`

```
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] Dalla stessa riga osserviamo che l'architettura hardware del sistema bersaglio è equipaggiata con un processore PowerPC

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
```

[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

```
561 _DL_ISO8583_FIELD_Pack
562 _DL_ISO8583_FIELD_Unpack
563 ...
```

1.1.1.2 Analisi assembler

Non avendo a disposizione alcuna macchina equipaggiata con un processore

1.1.2 Analisi del file `Injection_API_executable_e`

In questa sezione dimostreremo come il file di tipo **eXtended COFF** denominato `Injection_API_executable_e` sia in grado di eseguire un attacco di **code injection** a danno di un processo in esecuzione in modo tale da modificarne il comportamento a favore degli attaccanti.

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`

1.1.2.1 Stringhe stampabili rilevanti

Seguendo lo stesso ragionamento descritto nella sezione 1.1.1.1, possiamo osservare dal listato 1.4 come la versione di GCC utilizzata è stata la 4.8.5 (versione rilasciata il 23 giugno 2015[10]) mentre la versione del sistema operativo target sia stato la V 7.1. Non abbiamo dati sufficienti per Quest'ultima informazione non è tuttavia sufficiente per risalire

Sfortunatamente non è possibile risalire echnology Levels (TLs) [11] 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. [12]

Listing 1.4: Stringhe estratte dal file `Injection_API_executable_e`

```
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
```

[10] Cfr. https://gcc.gnu.org/gcc-4.8/
[11] http://ibmsystemsmag.com/aix/tipstechniques/migration/oslevel_versions/
[12] Cfr: https://www-01.ibm.com/support/docview.wss?uid=isg3T1012517

```
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
357 shr_xpg5_64.o
```

Dalla riga 944 riportata nel listato 1.5 possiamo ricavare il nome e la versione del compilatore usato, lo XL C/C++ for AIX versione 11.1.0.1.

possiamo ricavare l'esatto compilatore utilizzato XL C for AIX, V11.1; Nativamente non supportato dal sistema operativo AIX V7.1[13] ma il supporto è stat oaggiunto successivamente nel stemmbre 2010. [14]

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 ...
```

Nei listati successivi, come la 1.6, possiamo notare numerose stringhe contenenti i ben noti *conversion specifier* utilizzati nella stringa format passata come input alla ben nota funzione `fprintf`; molti dei conversion specifier utilizzati sono nella forma %llX che permette di stampare dati interi senza segno in esadecimale. La copiosa presenza di stampe suggerisce che il malware fosse munito di un meccanismo di log, intuizione confermata anche dall'analisi della NCCIC.

Le stringhe riportate nelle righe 333, 334 e 335 suggeriscono che l'applicazione fosse una command-line utility tale da permettere agli attaccanti ci condurre l'attacco di code injection sui sistemi IBM AIX

Listing 1.6: Stringhe estratte dal file `Injection_API_executable_e`

```
320 ...
321 [main] Inject Start
322 [main] SAVE REGISTRY
323 [main] proc_readmemory fail
324 [main] toc=%llX
325 [main] path::%s
326 [main] data(%p)::%s
327 [main] Exec func(%llX) OK
328 [main] Exec func(%llX) fail ret=%X
329 [main] Inject OK(%llX)
330 [main] Inject fail ret=%llX
331 [main] Eject OK
332 [main] Eject fail ret=%llX
333 Usage: injection pid dll_path mode [handle func toc]
334        mode = 0 => Injection
335        mode = 1 => Ejection
```

[13] https://www-01.ibm.com/support/docview.wss?uid=swg21326972
[14] http://www-01.ibm.com/support/docview.wss?uid=swg1IZ84777

```
336 [main] handle=%llX, func=%llX, toc=%llX
337 [main] ERROR::g_pid(%X) <= 0
338 [main] ERROR::load_config fail
339 [main] ERROR::eject & argc != 7
340 [main] ERROR::g_dl_handle(%llX) <= 0
341 [main] WARNING::func_addr(%llX), toc_addr(%llX)
342 ...
```

Dalle stringhe riportate nel listato 1.7 possiamo intuire la presenza di una ipotetica funzione `out_regs` utilizzata dagli attaccanti per stampare forse su un file di log il contenuto dei seguenti registri del processore:

- GPRs (General Purpose Registers)
- IAR (Instruction Address Register)
- MSR (Machine State Register)
- LR (Link Register)
- CR (Condition Register)
- CTR (Control Register)
- GPRs (General Purpose Registers)

Listing 1.7: Stringhe estratte dal file `Injection_API_executable_e`

```
320 [out_regs] IAR=%llX
321 [out_regs] MSR=%llX
322 [out_regs] CR=%llX
323 [out_regs] LR=%llX
324 [out_regs] CTR=%llX
325 [out_regs] GPR%d=%llX
```

Come si può apprendere dalla documentazione ufficiale fornita dalla IBM, ogni informazione riguardante un processo con identificatore `pid` è rappresentata da un grande insieme di file contenuti nella directory `/proc/pid`[15] tra cui ricordiamo:

/proc/pid/status Questo file riporta lo **stato** del processo `pid`.

/proc/pid/ctl Rappresenta il **Control File** del processo `pid` ed è usato per manipolare l'esecuzione del processo `pid`.

/proc/pid/as Rappresenta l'Address space del processo `pid`.

Nel listato ... si apprende come il malware ricostruisce tali percorsi con una chiamata sprintf (lo sappiamo perché è stata trovato un riferimento nel file) e quindi il malware è in grado di conoscere lo stato del processo attaccato, di manipolarne l'esecuzione e modificarne la memoria.

[15] pag 244

Listing 1.8: Stringhe estratte dal file `Injection_API_executable_e`

```
320 /proc/%d/ctl
321 /proc/%d/status
322 /proc/%d/as
```

Il contenuto presente nel listato 1.9 indica come l'attacco preveda la sospensione del processo bersaglio contro il quale eseguire la code injection: dalla documentazione ufficiale AIX si apprende come l'esecuzione dei processi possa essere alterata scrivendo opportuni messaggi all'interno di particoali file chiamti **ctl** (*control*) e **lwpctl** (*thread control*)[16]. Tutti i messaggi di controllo sono descritti da un numero intero che ne identifica l'operazione seguita da altri operandi numerici se previsti [17]. In particolare esiste il comando PCSTOP che permette di arrestare i thread di un particolare processo. Presumibilmente nel listato viene stampato l'identificatore del processo a cui è stato inviato il messaggio PCWSTOP.

Listing 1.9: Stringhe estratte dal file `Injection_API_executable_e`

```
319 ...
320 [proc_wait] PCWSTOP pid=%d, ret=%d, err=%d(%s)
321 [proc_wait] tid=%d, why=%d, what=%d, flag=%d, sig=%d
322 ...
```

Dalla documentazione ufficiale si può facilmente apprendere come l'operatore PCRUN, di cui possiamo notarne il riferimento nel listato 1.10, possa essere utilizzato per riavviare l'esecuzione di un processo dopo essere stato arrestato. Ciò indica come l'attacco preveda il riavvio del processo dopo aver eseguito la *code injection*.

Listing 1.10: Stringhe estratte dal file `Injection_API_executable_e`

```
308 [proc_continue] PCRUN pid=%d, arg=%d, ret=%d, err=%d(%s)
```

Infatti come dimostra in modo inequivocabile il listato, il malware è in grado di leggere la memoria allocata dal sistema operativo di un processo e di modificarla per effettuare la code injection vera e propria.

```
1 [proc_readmemory] ret=%d, err=%d(%s), addr=%p, len=%d, data
    =%p
2 [proc_readmemory] (%X~%X) %02X %02X %02X %02X %02X %02X %02X
     %02X %02X %02X %02X %02X %02X %02X %02X %02X %02X
3 [proc_writememory] ret=%d, err=%d(%s), addr=%p, len=%d, data
    =%p
```

La manipolazione del processo bersaglio avviene per mezzo di una serie di segnali tra cui:

PCSET sets one or more modes of operation for the traced process.

[16]Cfr IBM - *AIX Version 6.1:Reference File* - pp. 230
[17]Cfr IBM - *AIX Version 6.1:Reference File* - pp. 239

PCRUN Riesegue un thread dopo essere stato arrestato; l'operando è un set di flag contenuto in un int.

PCSENTRY Instructs the process's threads to stop on entry to specified system calls. T

PCSFAULT Defines a set of hardware faults to be traced in the process. When incurring one of these faults, a thread stops.

```
1 [proc_attach] PCSET pid=%d, ret=%d, err=%d(%s)
2 [proc_attach] PCSTOP pid=%d, ret=%d, err=%d(%s)
3 [proc_attach] PCSTRACE pid=%d, ret=%d, err=%d(%s)
4 [proc_attach] PCSFAULT pid=%d, ret=%d, err=%d(%s)
5 [proc_attach] PCSENTRY pid=%d, ret=%d, err=%d(%s)
6 [proc_detach] PCSTRACE pid=%d, ret=%d, err=%d(%s)
7 [proc_detach] PCSFAULT pid=%d, ret=%d, err=%d(%s)
8 [proc_detach] PCSENTRY pid=%d, ret=%d, err=%d(%s)
9 [proc_detach] PCRUN pid=%d, ret=%d, err=%d(%s)
```

1.1.2.2 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

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

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

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

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

Listings