

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

¹Università 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 <code>2.so</code>	3
1.1.1.1	Stringhe stampabili rilevanti	3
1.1.1.2	Analisi assembler	5
1.1.2	Analisi del file <code>Injection_API_executable_e</code>	6
1.1.2.1	Stringhe stampabili rilevanti	6
1.1.2.2	Disassemblaggio	11

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...	9ddacbcd0700dc4b9babcd09ac1cebe23a0035099cb612e6c85ff4dff087a26
8efaabb7b1700686efedadb7949eba...	a9bc09a17d55fc790568ac864e3885434a43c33834551e027adb1896a463aafc
d0a8e0b685c2ea775a74389973fc92...	ab88f12f0a30b4601dc26dbae57646efb77d5c6382fb25522c529437e5428629
2.so	ca9ab48d293cc84092e8db8f0ca99cb155b30c61d32a1da7cd3687de454fe86c
Injection_API_executable_e	d465637518024262c063f4a82d799a4e40ff3381014972f24ea18bc23c3b27ee
Injection_API_log_generating_s	e03dc5f1447f243cf1f305c58d95000ef4e7dbcc5c4e91154daa5acd83fea9a8
inject_api	f3e521996c85c0cdb2bfb3a0fd91eb03e25ba6feef2ba3a1da844f1b17278dd2

1.1.1.1 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 UNIX¹ fino al 1999², anno della definitiva adozione dello standard **Executable and Linkable Format** o **ELF**. XCOFF rappresenta tuttavia uno standard proprietario sviluppato da IBM³ 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.so

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

1.1.1.1.1 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
```

¹Cfr. <https://it.wikipedia.org/wiki/COFF>

²Cfr. https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

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

⁴Cfr. <https://www.ibm.com/it-infrastructure/power/os/aix>

⁵Cfr. 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>

⁶Cfr. <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 1.2⁷, 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 2007⁸) 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.so

Descrizione	Valore	Comando Unix
Nome	2.so	stat -c "%n" 2.so
Dimensione (<i>byte</i>)	89088	stat -c "%s" 2.so
Data ultima modifica	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

Cominciamo lo studio del file **Injection_API_executable_e** partendo dall'analisi delle stringhe stampabili estratte attraverso il tool **strings**. Seguendo lo stesso ragionamento precedentemente descritto nella sezione 1.1.1.1, possiamo osservare dal listato 1.4 la versione di GCC e del sistema operativo AIX utilizzati per eseguire la *build* del malware, che risultano essere rispettivamente pari a 4.8.5 (la data pubblicazione risale al 23 giugno 2015¹⁰), e 7.1 (commercializzata a partire da settembre 2010).¹¹ Sfortunatamente non è stato possibile risalire alla versione degli aggiornamenti, che IBM identifica con il nome di *Technology Levels* (TLs)¹², installati sul sistema operativo bersaglio al momento dell'attacco in modo tale da conoscere l'entità del rischio a cui si sottoponeva il sistema bancario. In ogni caso il supporto ufficiale per la versione 7.1, sostituita dalla ben più moderna versione 7.2 rilasciato nel dicembre 2015, è già terminato il 30 Novembre 2013 benché la versione 7.1 TL5 riceverà supporto fino ad aprile 2022.¹³

Listing 1.4: Stringhe estratte dal file Injection_API_executable_e

```
347 ...  
348 /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  
349 ...
```

¹⁰Cfr. <https://gcc.gnu.org/gcc-4.8/>

¹¹Cfr. <https://www-01.ibm.com/support/docview.wss?uid=isg3T1012517>

¹²http://ibmsystemsmag.com/aix/tipstechniques/migration/oslevel_versions/

¹³Cfr. <https://www-01.ibm.com/support/docview.wss?uid=isg3T1012517>

Un altro riferimento ai tool utilizzati dagli attaccanti lo possiamo ricavare dalla riga 944 riportata nel listato 1.5 dove apprendiamo l'utilizzo del compilatore lo **XL C/C++ for AIX** versione 11.1.0.1, quest'ultimo appositamente ottimizzato dalla IBM per i propri sistemi operativi. Ci risulta che tale versione del compilatore non fosse disponibile per AIX 7.1 al momento del rilascio e che sia divenuto disponibile in seguito ad un aggiornamento.¹⁴

Listing 1.5: Stringhe estratte dal file `Injection_API_executable_e`

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

Analizziamo ora in dettaglio le varie operazioni compiute dal malware durante la sua esecuzione. Innanzitutto, osservando la particolare configurazione dei listati successivi come la numero 1.6, notiamo quello che dovrebbe essere un insieme di stampe nella forma `[FUNCTION NAME] [...] eseguito probabilmente da un meccanismo di log`, il che è stato confermato dalla già citata analisi della NCCIC; infatti, notiamo un gran numero di stringhe contenenti i ben noti *conversion specifier* utilizzati nelle stringhe che specificano il formato delle stampe eseguite dalla funzione `fprintf` di cui molti sono nella forma `%lX`, usata per stampare numeri interi senza segno in forma esadecimale¹⁵. Come è stato confermato da altre analisi, le stringhe riportate nelle righe 333, 334 e 335 suggeriscono come l'applicazione è stata progettata per essere una command-line utility interattiva e di come il meccanismo di log sia stato utilizzato per ottenere informazioni e consentire agli attaccanti un attacco mirato.

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=%lX
325 [main] path::%s
326 [main] data(%p)::%s
327 [main] Exec func(%lX) OK
328 [main] Exec func(%lX) fail ret=%X
329 [main] Inject OK(%lX)
330 [main] Inject fail ret=%lX
331 [main] Eject OK
332 [main] Eject fail ret=%lX
333 Usage: injection pid dll_path mode [handle func toc]
334         mode = 0 => Injection
335         mode = 1 => Ejection
336 [main] handle=%lX, func=%lX, toc=%lX
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(%lX) <= 0
```

¹⁴<https://www-01.ibm.com/support/docview.wss?uid=swg21326972>

¹⁵Cfr. <http://man7.org/linux/man-pages/man3/printf.3.html>


```

341 [main] WARNING::func_addr(%lX), toc_addr(%lX)
342 ...

```

Prima di descrivere le varie fasi dell'attacco, è indispensabile comprendere come vengono rappresentati e gestiti i **processi** nei sistemi operativi AIX versione 6.1 e 7.1. Ogni processo del sistema viene opportunamente rappresentato da un insieme di file ognuno dei descrive un particolare aspetto di un processo come ad esempio il suo stato, le informazioni sui file descriptors, privilegi ecc. Tutti i processi sono raccolti all'interno della directory `/proc` mentre tutti i file di un dato processo con identificatore pari a `pid` sono raccolti all'interno della directory `/proc/pid`; ciò permette agli attaccanti di ricavare tutte le informazioni necessarie di tutto il sistema attraverso le system call standard per la lettura e scrittura sui file come `open()`, `close()`, `read()` e la `write()`.¹⁶ Di tutti i file contenuti in una generica directory `/proc/pid`, di cui ne riportiamo una piccola frazione nella tabella 1.4¹⁷, è importante ricordare:

`/proc/pid/as` Contiene l'immagine dello spazio degli indirizzi del processo e può essere aperto sia per la lettura che per la scrittura e supporta la subroutine `lseek` per accedere all'indirizzo virtuale di interesse.¹⁸

`/proc/pid/ctl` Un file di sola scrittura in cui vengono scritti messaggi strutturati che consentono di modificare lo stato del processo e dunque il suo comportamento. I messaggi di controllo vengono scritti direttamente sul file `ctl` del processo e gli effetti sono visibili immediatamente attraverso i file di stato del processo.¹⁹

`/proc/pid/status` Contiene informazioni sullo stato del processo.²⁰

Tabella 1.4: Sottoinsieme dei file contenuti in `/proc/pid`

File	Descrizione
<code>/proc/pid/status</code>	Status of process <code>pid</code>
<code>/proc/pid/ctl</code>	Control file for process <code>pid</code>
<code>/proc/pid/as</code>	Address space of process <code>pid</code>
<code>/proc/pid/cred</code>	Credentials information for process <code>pid</code>
<code>/proc/pid/sigact</code>	Signal actions for process <code>pid</code>
<code>/proc/pid/sysent</code>	System call information for process <code>pid</code>

Dal momento che, come mostrato nel listato 1.7, sono stati individuate tre stringhe che fanno riferimento ai suddetti file descrittori di processo, in particolare ai file `ctl`, `status` e `as`, si può affermare che il malware, ricostruendo probabilmente le directory attraverso una chiamata `sprintf` come dimostrano la presenza del *conversion specifier* `%d` e la presenza di varie stringhe `sprintf` (riga 372 e 824), è stato progettato per accedere a questi file con lo scopo di manipolare il normale flusso di esecuzione del processo bersaglio.

¹⁶Cfr. IBM - *AIX Version 7.1: Files References* - pag. 232-246

¹⁷Per una lista completa Cfr. *ivi* pag. 246

¹⁸Cfr. *ivi* pag. 232

¹⁹Cfr. *ivi* pag. 232

²⁰Cfr. *ivi* pag. 232

Listing 1.7: Stringhe estratte dal file `Injection_API_executable_e`

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

Come abbiamo detto in precedenza, il flusso di esecuzione di un processo può essere modificato eseguendo la scrittura di appositi messaggi nel file `ctl` i quali da un codice operativo rappresentato da un `int` che identifica la specifica operazione seguita da ulteriori argomenti (se presenti).²¹ Il listato 1.9 dimostra che il malware interrompe esplicitamente l'esecuzione del processo bersaglio attraverso l'uso del messaggio **PCWSTOP** il cui scopo è quello di arrestare l'esecuzione di un processo `pid` passato come argomento.²²

Listing 1.8: 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 ...
```

Il listato 1.9 mostra invece l'uso di vari messaggi tra cui:

PCSET Serve per impostare una serie di flag ad un processo (`PR_ASYNC`, `PR_FORK`, `PR_KLC` ecc.).²³

PCRUN Riesegue un thread dopo essere stato arrestato.

PCSENTRY Indica al thread di interrompere la sua esecuzione nel momento in cui richiama una specifica system call.

PCSFAULT Definisce un insieme di *hardware faults* tracciabili nel processo. Il thread si interrompe quando si verifica una fault.

Listing 1.9: Stringhe estratte dal file `Injection_API_executable_e`

```
299 ...
300 [proc_attach] PCSET pid=%d, ret=%d, err=%d(%s)
301 [proc_attach] PCSTOP pid=%d, ret=%d, err=%d(%s)
302 [proc_attach] PCSTRACE pid=%d, ret=%d, err=%d(%s)
303 [proc_attach] PCSFAULT pid=%d, ret=%d, err=%d(%s)
304 [proc_attach] PCSENTRY pid=%d, ret=%d, err=%d(%s)
305 [proc_detach] PCSTRACE pid=%d, ret=%d, err=%d(%s)
306 [proc_detach] PCSFAULT pid=%d, ret=%d, err=%d(%s)
307 [proc_detach] PCSENTRY pid=%d, ret=%d, err=%d(%s)
308 [proc_detach] PCRUN pid=%d, ret=%d, err=%d(%s)
309 ...
```

²¹ Cfr. *ivi* pag. 242

²² *Ibidem*

²³ Cfr. *ivi* pag. 234

I listati 1.10 e 1.11 dimostrano come il malware raccolga le informazioni necessarie al suo scopo attraverso l'accesso in lettura alle informazioni di stato del processo e al contenuto dei registri. Abbiamo riportato nella tabella 1.5 una descrizione dei registri ispezionati dal malware²⁴. del processore tra cui risultano

Listing 1.10: Stringhe estratte dal file `Injection_API_executable_e`

```

299 ...
300 [proc_getregs] GETREG pid=%d, ret=%d, err=%d(%s)
301 [proc_getregs] GETSTATUS pr_syscall=%d, pr_why=%d, pr_what=%
    d, pr_flags=%d, pr_cursig=%d
302 [proc_setregs] SETREG pid=%d, ret=%d, err=%d(%s)
303 ...

```

Listing 1.11: Stringhe estratte dal file `Injection_API_executable_e`

```

320 [out_regs] IAR=%11X
321 [out_regs] MSR=%11X
322 [out_regs] CR=%11X
323 [out_regs] LR=%11X
324 [out_regs] CTR=%11X
325 [out_regs] GPR%d=%11X

```

Tabella 1.5: Breve descrizione dei registri ispezionati dal malware

Registro	Nome esteso	Descrizione
LR	Link Register	E' usato per ospitare l'indirizzo dell'istruzione successiva ad una operazione di salto.
CR	Condition Register	Un registro da 32 bit usato per specificare varie classi di operazioni.
CTR	Control Register	Un registro da 32 bit usato per specificare varie classi di operazioni.
IAR	Instruction Address Register	Usato per contenere l'indirizzo dell'istruzione successiva.
GPRs	General Purpose Registers	Un registro da 32 bit per usi generici. E' spesso usato spesso come storage momentaneo per varie classi di operazioni.
MSR	Machine State Register	Registro da 32 bit usato per specificare varie classi di operazioni.

Infine nei listati 1.13 e 1.12 viene mostrato l'accesso in lettura e in scrittura allo spazio di indirizzamento del processo bersaglio e di come riavvi l'esecuzione del processo dopo la code injection.

²⁴Cfr. IBM - *AIX Version 7.1: Assembler Language Reference*

Listing 1.12: Stringhe estratte dal file Injection_API_executable_e

```

308 ...
309 [proc_readmemory] ret=%d, err=%d(%s), addr=%p, len=%d, data
    =%p
310 [proc_readmemory] (%X~%X) %02X %02X %02X %02X %02X %02X %02X
    %02X %02X %02X %02X %02X %02X %02X %02X %02X
311 [proc_writememory] ret=%d, err=%d(%s), addr=%p, len=%d, data
    =%p
312 ...

```

Listing 1.13: Stringhe estratte dal file Injection_API_executable_e

```

308 ...
309 [proc_continue] PCRUN pid=%d, arg=%d, ret=%d, err=%d(%s)
310 ...

```

1.1.2.2 Disassemblaggio

```

1  bl      0x10000674 <.out_log>
2  bl      0x10001220 <.proc_attach>
3  li      r3,0
4  bl      0x10001a28 <.proc_continue>
5  li      r3,0
6  li      r4,0
7  bl      0x10001b44 <.proc_wait>
8  ld      r3,728(r2)
9  bl      0x10000674 <.out_log>
10 addi    r9,r31,152
11 mr      r3,r9
12 bl      0x10001ee4 <.proc_getregs>
13 addi    r9,r31,152
14 mr      r3,r9
15 bl      0x10000c80 <.out_regs>
16 addi    r8,r31,536
17 addi    r10,r31,152
18 li      r9,384
19 mr      r3,r8
20 mr      r4,r10
21 mr      r5,r9
22 bl      0x1000324c <.memmove>
23 nop
24 ld      r9,536(r31)
25 addi    r9,r9,-16
26 mr      r3,r9
27 li      r4,16384
28 bl      0x10000b48 <.file_dump>

```

La traduzione dal linguaggio macchina all'assembler del file è stato usufruendo 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 1.6: 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_field_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 the return address for use by an invoked subroutine. The return address is the address of the instruction immediately 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 <code>2.s0</code>	3
1.3	Dettagli tecnici del file <code>2.s0</code>	6
1.4	Sottoinsieme dei file contenuti in <code>/proc/pid</code>	8
1.5	Breve descrizione dei registri ispezionati dal malware	10
1.6	Dettagli tecnici del file <code>2.s0</code>	12

Listings

1.1	Stringhe estratte dal file <code>2.so</code>	3
1.2	Formato del percorso di installazione delle librerie GCC nei sistemi operativi AIX	4
1.3	Stringhe estratte dal file <code>2.so</code>	4
1.4	Stringhe estratte dal file <code>Injection_API_executable_e</code>	6
1.5	Stringhe estratte dal file <code>Injection_API_executable_e</code>	7
1.6	Stringhe estratte dal file <code>Injection_API_executable_e</code>	7
1.7	Stringhe estratte dal file <code>Injection_API_executable_e</code>	9
1.8	Stringhe estratte dal file <code>Injection_API_executable_e</code>	9
1.9	Stringhe estratte dal file <code>Injection_API_executable_e</code>	9
1.10	Stringhe estratte dal file <code>Injection_API_executable_e</code>	10
1.11	Stringhe estratte dal file <code>Injection_API_executable_e</code>	10
1.12	Stringhe estratte dal file <code>Injection_API_executable_e</code>	11
1.13	Stringhe estratte dal file <code>Injection_API_executable_e</code>	11