



Università degli Studi di Roma “Tor Vergata”

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

Il malware FASTCash

Sicurezza informatica e Internet

Candidati:

Andrea Graziani

Matricola 0273395

Alessandro Boccini

Matricola 0277414

Ricardo Gamucci

Matricola 0274716

Docente:

Maurizio Naldi

Indice

1	Introduzione	3
	<i>Autore: Andrea Graziani (0273395)</i>	
1.1	Descrizione generale dell'attacco	4
	<i>Autore: Ricardo Gamucci (0274716)</i>	
	<i>Editing: Andrea Graziani (0273395)</i>	
1.2	Manipolazioni dei messaggi ISO 8583	6
	<i>Autore: Alessandro Boccini (0277414)</i>	
	<i>Editing: Andrea Graziani (0273395)</i>	
1.2.1	Manipolazione	6
1.3	Gruppo Lazarus	7
	<i>Autore: Ricardo Gamucci (0274716)</i>	
	<i>Editing: Andrea Graziani (0273395)</i>	
2	Analisi dell'attacco di Process Injection	9
	<i>Autore: Andrea Graziani (0273395)</i>	
2.1	Introduzione	9
2.1.1	Definizione della forma di attacco	9
2.1.2	Descrizione della variante di attacco adottata	9
2.1.3	Gli strumenti usati per l'analisi	10
2.2	I file di FASTCash responsabili dell'attacco	11
2.2.1	Il file <code>Injection_API_executable.e</code>	11
2.2.1.1	Analisi delle stringhe	11
2.2.1.1.1	Le attività di logging di FASTCash	13
2.2.1.1.2	La gestione dei processi in AIX	14
2.2.1.1.3	L'accesso del malware ai file dei processi	15
2.2.1.2	Analisi del codice assembly	18
2.2.1.2.1	La procedura <code>main</code>	18
2.2.1.2.2	La procedura <code>inject</code>	19
2.2.2	Il file <code>Injection_API_log_generating_script</code>	22
2.2.3	Il file <code>2.so</code>	23
2.2.3.1	Analisi delle stringhe	23
2.2.3.2	Analisi del codice assembly	25
2.2.3.2.1	<code>DL_ISO8583_MSG_SetField_Bin</code>	25
2.2.3.2.2	<code>out_dump_log</code>	26
3	Analisi degli impatti subiti	27

<i>Autore: Andrea Graziani (0273395)</i>	
3.1	Impatti socio-economici 27
<i>Autore: Alessandro Boccini (0277414)</i>	
<i>Editing: Andrea Graziani (0273395)</i>	
3.2	Impatti sulla reputazione 28
3.3	Impatti sul sistema 29
4	Contromisure 30
<i>Autore: Andrea Graziani (0273395)</i>	
4.1	Aggiornamenti software 30
4.1.1	Analisi dell'efficacia degli aggiornamenti software 30
4.2	Principio del privilegio minimo 31
4.2.1	Le liste di controllo degli accessi 31
4.2.2	Meccanismi di Whitelisting 31
4.3	Riduzione della superficie di attacco 32
4.3.1	Gli Unikernel 33
4.3.2	Ridondanza e virtualizzazione 33
4.4	Monitoring 34
4.5	Autenticazione 34
4.6	Ripristino con le istantanee (snapshot) di sistema 35
4.7	Contrasto agli attacchi di phishing 35
<i>Autore: Ricardo Gamucci (0274716)</i>	
<i>Editing: Andrea Graziani (0273395)</i>	
5	I sistemi vulnerabili all'attacco 36
<i>Autore: Ricardo Gamucci (0274716)</i>	

Capitolo 1

Introduzione

Autore: Andrea Graziani (0273395)

Lo scopo della presente relazione è quella di fornire un'analisi il più possibile dettagliata del malware denominato **FASTCash**¹ la cui paternità, secondo le analisi forensi eseguite da varie aziende ed agenzie, tra cui la **Symantec**, **Kaspersky** e la **NCCIC**², è attribuita ad un gruppo di cyber-criminali conosciuto come *Lazarus Group*^{3 4}, di cui forniremo una descrizione più dettagliata nella sezione 1.3.

La scoperta del malware risale precisamente al **2 ottobre 2018** giorno in cui, grazie al lavoro congiunto effettuato da varie agenzie ed istituzioni americane, tra cui **FBI**, **Dipartimento del Tesoro Americano** e la **DHS** (*Department of Homeland Security*)⁵, venne pubblicato sul sito del CISA l'avviso **TA18-275A**, secondo la quale esponenti del gruppo Lazarus avevano condotto con successo una serie di attacchi, denominati dalla CISA come *FASTCash*, per il **prelievo fraudolento di denaro contante dagli ATM** (*Automated Teller Machines*) di varie banche in oltre 30 paesi diversi sparsi fra Asia e Africa⁶. E' importante sottolineare che, sebbene l'esistenza del malware sia stata resa di dominio pubblico solo allora, la CISA ha dichiarato che il gruppo Lazarus ha condotto attacchi di tipo FASTCash già a partire dal 2016.

Per essere precisi, come si evince dal rapporto **AR18-275A** pubblicato dalla CISA⁷, il malware FASTCash è in realtà composto da un'insieme di **12 file**, ognuno dei quali avente un ruolo e uno scopo specifico. Abbiamo riportato per comodità una lista completa dei file di FASTCash in 1.1; tre di essi sono stati analizzati in dettaglio nel capitolo 2.

¹<https://www.symantec.com/security-center/writeup/2018-110615-2942-99>

²Stiamo parlando della "National Cybersecurity and Communications Integration Center" un'agenzia federale americana specializzata in sicurezza informatica facente parte della CISA (*Cybersecurity and Infrastructure Security Agency*)

Per ulteriori informazioni in merito si veda: <https://www.us-cert.gov/about-us>

³<https://www.symantec.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware>

⁴<https://www.us-cert.gov/ncas/alerts/TA18-275A>

⁵*Ibidem*

⁶*Ibidem*

⁷Cfr. <https://www.us-cert.gov/ncas/analysis-reports/AR18-275A>

Tabella 1.1: Lista dei file del malware FASTCash

Nome file	SHA256 digest
Lost_File.so	10ac312c8dd02e417dd24d53c99525c29d74dcbc84730351ad7a4e0a4b1a0eba
Unpacked_dump_4a740227eeb82c20...	10ac312c8dd02e417dd24d53c99525c29d74dcbc84730351ad7a4e0a4b1a0eba
Lost_File1_so_file	3a5ba44f140821849de2d82d5a137c3bb5a736130dddb86b296d94e6b421594c
4f67f3e4a7509af1b2b1c6180a03b3...	4a740227eeb82c20286d9c112ef95f0c1380d0e90ffb39fc75c8456db4f60756
5cfa1c2cb430bec721063e3e2d144f...	820ca1903a30516263d630c7c08f2b95f7b65dffce21129c51c9e21cf9551c6
Unpacked_dump_820ca1903a305162...	9ddacbcd0700dc4b9babcd09ac1cebe23a0035099cb612e6c85ff4dffd087a26
8efaabb7b1700686efedadb7949eba...	a9bc09a17d55fc790568ac864e388543a43c33834551e027adb1896a463aafc
d0a8e0b685c2ea775a74389973fc92...	ab88f12f0a30b4601dc26dbae57646efb77d5c6382fb25522c529437e5428629
2.so	ca9ab48d293cc84092e8db8f0ca99cb155b30c61d32a1da7cd3687de454fe86c
Injection_API_executable.e	d465637518024262c063f4a82d799a4e40ff3381014972f24ea18bc23c3b27ee
Injection_API_log_generating_s	e03dc5f1447f243cf1f305c58d95000ef4e7dbcc5c4e91154daa5acd83fea9a8
inject_api	f3e521996c85c0cdb2bfb3a0fd91eb03e25ba6feef2ba3a1da844f1b17278dd2

1.1 Descrizione generale dell'attacco

Autore: **Ricardo Gamucci (0274716)**

Editing: **Andrea Graziani (0273395)**

Per effettuare con successo un attacco FASTCash, i cyber-criminali violavano dapprima le reti delle banche bersaglio allo scopo di ottenere accesso completo ai cosiddetti **server di payment switch** (*switch application servers*), i server responsabili della gestione delle transazioni finanziarie tra gli ATM e il sistema informatico principale della banca.⁸

Si presuppone che gli attaccanti, al fine di ottenere accesso alle suddette reti, abbiano adottato tecniche di **spear-phishing**⁹, ossia una serie di attacchi di *phishing* che, a differenza di quelli più comuni, hanno come bersaglio individui *specifici*, vale a dire persone di cui gli attaccanti potevano conoscere diverse informazioni (nome, sesso, ruolo, abitudini, ecc.) al fine di aumentare le probabilità di successo¹⁰.

Il vero scopo dei suddetti attacchi di phishing riguardava l'installazione, nel computer della vittima di un apposito malware¹¹ che, secondo il report pubblicato dalla NCCIC, era in grado di colpire i sistemi operativi Windows dei dipendenti della banca al fine di consentire agli attaccanti l'accesso in remoto al sistema colpito¹².

Per conseguire il suddetto scopo, il malware eseguiva una serie di operazioni tra cui:

- Alterare le impostazioni di *Windows Firewall* allo scopo di consentire l'instaurazione di connessioni TCP verso la macchina dall'esterno.¹³

⁸<https://www.symantec.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware>

⁹<https://securityaffairs.co/wordpress/76798/hacking/fastcash-hidden-cobra-attacks.html>

¹⁰<https://www.kaspersky.it/resource-center/definitions/spear-phishing>

¹¹In base al rapporto NCCIC AR18-275A, il nome del malware in questione è 5cfa1c2cb430bec721063e3e2d144feb

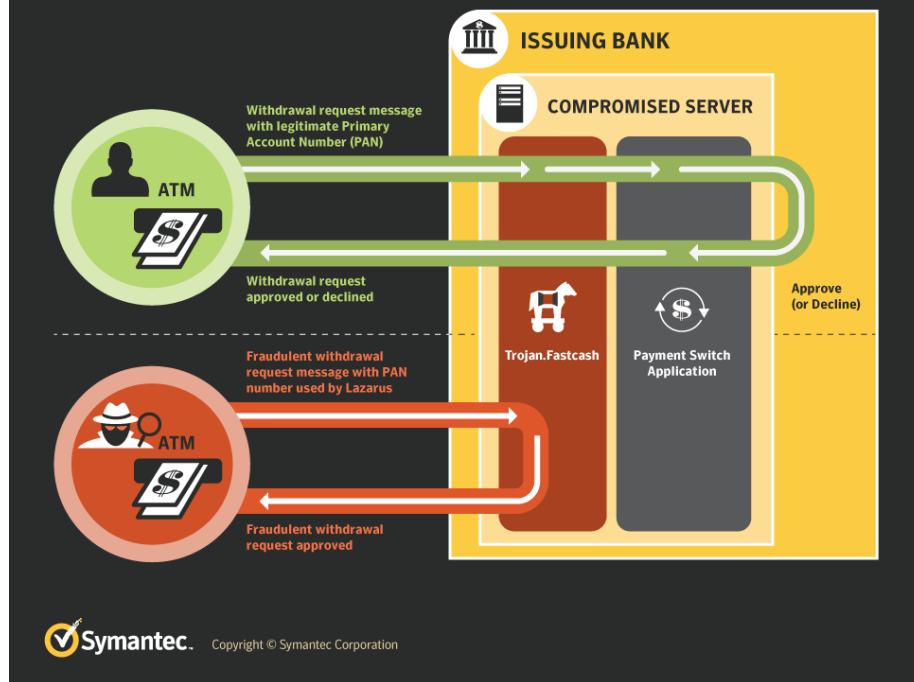
¹²<https://www.us-cert.gov/ncas/analysis-reports/AR18-275A>

¹³*Ibidem*

FASTCash

How the Lazarus Group is Emptying Millions from ATMs

Symantec uncovers Trojan.Fastcash, the tool used by North Korea-linked Lazarus group to mount ATM attacks



- Caricare in memoria uno speciale **modulo proxy** che, secondo la NCCIC, oltre ad essere stato progettato ad agire come una *command-line utility*, forzava la macchina ad agire come se fosse un proxy server, consentendo agli attaccanti di interagire in remoto con il malware.¹⁴

Le analisi condotte dalla NCCIC hanno dimostrato che gli attaccanti, per mezzo del suddetto malware, sono riusciti a trasferire un **injection tool** all'interno dei server di payment switch delle banche coinvolte dell'attacco; analizzeremo in dettaglio l'injection tool di FASTCash nella sezione 2.

Il suddetto injection tool è stato utilizzato per introdurre, all'interno di uno o più processi legittimi in esecuzione sui payment switch server, codice malevolo capace di interagire ed alterare i messaggi basati sul **protocollo ISO 8583** scambiati tra gli ATM e il sistema informatico bancario centrale, consentendo agli aggressori di prelevare denaro agli sportelli automatici.

¹⁴ *Ibidem*

1.2 Manipolazioni dei messaggi ISO 8583

Autore: Alessandro Boccini (0277414)

Editing: Andrea Graziani (0273395)

L'ISO 8583 è uno standard internazionale di messaggio nell'ambito delle transazioni finanziarie, utilizzato da molte reti finanziarie, tra cui Visa e MasterCard e dalla quasi totalità degli enti finanziari mondiali.

Dal punto di vista strutturale, un messaggio ISO 8583 è composto da 3 campi:

Message Type Identifier (MTI) Identifica la tipologia del messaggio ed è composto da un insieme di 4 valori dove:

- Il primo indica la versione del protocollo il quale, qualora assumesse, ad esempio, il valore 0, denota la versione del 1987 del protocollo.
- Il secondo è utilizzato per indicare la classe del messaggio; un valore pari a 2 indica un messaggio di tipo finanziario.
- Il terzo indica, invece, la funzione svolta dal messaggio; un valore pari a 0 è usato per rappresentare una richiesta mentre, se pari ad 1, una risposta.
- Infine, il quarto è usato per identificare il soggetto della transazione; il valore 0 indica un generico acquirente.

Bitmap Rappresenta un campo composto da 64 bit (o 16 caratteri esadecimali) in cui ogni bit posto a 1 denota la validità delle informazioni presenti nel campo *Data* del messaggio; per esempio, qualora il bit n. 2 assumesse valore pari ad 1, il campo **PAN** (*Primary Account Number*) risulta essere presente nel payload del messaggio.

Data Rappresenta il payload contenente tutte le informazioni relative alla transazione. Può contenere fino a 192 possibili elementi (128 nella versione del 1987) ognuno dei quali può contenere dati numerici, alfanumerici o binari di lunghezza fissa o variabile.

1.2.1 Manipolazione

Durante l'attacco, il malware interagisce esclusivamente con i messaggi dove:

1. Il MTI del messaggio contiene il valore 0200, ossia *una richiesta di prelievo di denaro da parte dell'acquirente*.
2. il valore del *Point of service entry modes*, indicante le modalità di accesso al sistema bancario, sia pari a 90, valore che indica l'uso di una carta di credito.

Una volta intercettato un messaggio con tali caratteristiche, il malware estrae il valore corrispondente al campo PAN e lo confronta con un insieme di valori PAN posseduti dagli attaccanti.

Tale verifica avviene per mezzo di una funzione contenuta all'interno delle librerie introdotte nel sistema; la funzione `CheckPan()`^{15 16}.

Qualora la verifica abbia avuto esito positivo, ovvero il PAN estratto dal messaggio corrisponde ad uno dei PAN in possesso agli attaccanti, il messaggio viene manipolato invocando la funzione `GenerateResponseTransaction1`, la quale modifica il terzo valore del campo MIT del messaggio impostandolo ad 1. Infine, il messaggio, con valore MIT pari ad 0210, viene inoltrato all'ATM che rilascerà i contanti ai criminali, completando così l'attacco.

In tutti gli altri casi, i messaggi vengono rilasciati e inoltrati al sistema bancario seguendo il normale iter.

1.3 Gruppo Lazarus

Autore: **Ricardo Gamucci (0274716)**

Editing: **Andrea Graziani (0273395)**

Il *Lazarus Group*, conosciuto anche come *HIDDEN COBRA* o *Guardians of Peace*, è un gruppo di cyber-criminali, composto da un numero imprecisato di individui, ritenuto responsabile di una serie di attacchi informatici perpetrati a danno di istituti bancari, software house e casino oltre ad una serie di attività legate al mining di cripto-valute.¹⁷

La paternità degli attacchi effettuati dal gruppo è stata dimostrata da Kaspersky Lab, nota azienda attiva nel campo della sicurezza informatica, la quale, attraverso un'analisi forense molto dettagliata sulle attività del gruppo in cui è stato eseguito un confronto minuzioso dei codici assembly estratti dai malware utilizzati in diversi attacchi, ha dimostrato l'uso di uno stesso *core base* attribuito al gruppo stesso.¹⁸

La NCCIC, Kaspersky Lab e la Symantec attribuiscono al gruppo Lazarus la responsabilità di attacchi molto famosi, tra cui spiccano per importanza ed entità il furto di dati a danno della Sony Pictures Entertainment, il furto di denaro dalla Banca Centrale del Bangladesh nonché dei danni provati dal ben noto ransomware WannaCry¹⁹.

Per essere precisi, la NCCIC la ritiene colpevole di ben 17 attacchi, tra cui ovviamente FASTCash, sebbene la certezza di tale numero possa essere messa in discussione dal fatto che il governo statunitense attribuisca alla HIDDEN COBRA la quasi totalità degli attacchi provenienti dalla nord-corea.²⁰

La presunta origine nordcoreana del gruppo è stata dimostrata, ancora una volta, da Kaspersky; l'azienda, attraverso un'analisi di vari server C2²¹, localizzati in Europa e usati dal gruppo per le loro attività criminali, ha rilevato l'uso di un piccolo intervallo di indirizzi IP di origine nordcoreana da cui si presup-

¹⁵<https://security.arizona.edu/content/ta18-275a-hidden-cobra-0T1\textendash-fastcash-campaign>

¹⁶<https://www.symantec.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware>

¹⁷<https://securelist.com/lazarus-under-the-hood/77908/>

¹⁸https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07180244/Lazarus_Under_The_Hood_PDF_final.pdf

¹⁹<https://www.theguardian.com/technology/2017/may/15/wannacry-ransomware-north-korea-lazarus-group>

²⁰<https://www.us-cert.gov/HIDDEN-COBRA-North-Korean-Malicious-Cyber-Activity>

²¹ *Command-and-control*, indicato a volte con la sigla C&C o C2, server usati dagli attaccanti per mantenere le comunicazioni con i sistemi compromessi

The Geography of financial attacks by Lazarus group

The malware by Lazarus group, infamous for its theft of \$81 million from Central Bank of Bangladesh, has been active since at least 2009. It has been spotted in the last couple of years in at least 18 countries.



pone avessero origine tutte le richieste di connessione verso i server C2, sebbene queste avvenissero per mezzo di server proxy e connessioni VPN.²²

²²<https://securelist.com/lazarus-under-the-hood/77908/>

Capitolo 2

Analisi dell'attacco di Process Injection

Autore: Andrea Graziani (0273395)

2.1 Introduzione

In questo capitolo analizzeremo dettagliatamente la tecnica di attacco usata dai cyber-criminali per alterare a loro vantaggio il corretto funzionamento dei server bancari presso le quali erano in esecuzione le applicazioni di *payment switch*.

2.1.1 Definizione della forma di attacco

I rapporti pubblicati dalla NCCIC¹ ² e dalla Symantec³ indicano che la forma di attacco adottata per compromettere i server dell'istituto bancario fosse stata una **process injection**.

Con la locuzione *process injection* si intende una tecnica che rende possibile l'esecuzione di codice arbitrario precedentemente introdotto all'interno dello spazio d'indirizzamento di un processo distinto in esecuzione.⁴

L'esecuzione di codice maligno nel contesto di un processo legittimo, oltre a garantire ai cyber-criminali l'accesso a tutte le risorse assegnate al suddetto processo da parte del SO (memoria, risorse di rete, dati ecc.), non viene generalmente individuata dai prodotti commerciali per la sicurezza informatica, essendo l'esecuzione del malware *nascosta*.⁵

2.1.2 Descrizione della variante di attacco adottata

Esistono molte varianti di attacco di process injection che, sfruttando diverse tipologie di vulnerabilità esposte dal sistema operativo, sono in grado di introdurre con successo codice arbitrario all'interno di un processo; la variante

¹<https://www.us-cert.gov/ncas/analysis-reports/AR18-275A>

²<https://www.us-cert.gov/ncas/alerts/TA18-275A>

³<https://www.symantec.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware>

⁴<https://attack.mitre.org/techniques/T1055/>

⁵*Ibid.*

adottata dai cyber-criminali nel malware FASTCash è conosciuta come **SIR**, acronimo di **Suspend-Inject-Resume**.

Come facilmente intuibile dal nome, tale tipologia di attacco prevede 3 fasi:⁶

1. La **sospensione del processo** bersaglio o, più specificatamente, di tutti i suoi thread.
2. **Alterazione dello stato del processo** attraverso la modifica del suo spazio di indirizzamento (*Address Space*) o dei valori contenuti nel suo PCB, come il valore che detiene l'indirizzo della successiva istruzione (*Program Counter/Instruction Pointer*) o i dati contenuti nei registri.
3. Il **riavvio del processo** attaccato in modo tale che esegua il codice maligno precedentemente introdotto.

2.1.3 Gli strumenti usati per l'analisi

Prima di procedere con la descrizione dettagliata dell'attacco perpetrato dal malware FASTCash, riportiamo di seguito i vari tool utilizzati durante le nostre analisi:

strings ⁷ Usato per l'estrazione di tutte le stringhe stampabili contenuti in un file.

stat ⁸ Utilizzato per ottenere alcune informazioni di base dei file tra cui nome, dimensione, data di ultima modifica, ecc.

file ⁹ Usato per determinare la tipologia di appartenenza di uno specifico file.

onlinedisassembler ¹⁰ Il de-assemblaggio dei file è stato eseguito utilizzando il servizio cloud **onlinedisassembler** che ci ha permesso di ricavare facilmente i listati di codice assembly dei file scritti per le architetture PowerPC™.

⁶<https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-t>

⁷<https://linux.die.net/man/1/strings>

⁸<https://linux.die.net/man/1/stat>

⁹<https://linux.die.net/man/1/file>

¹⁰<https://onlinedisassembler.com/>

2.2 I file di FASTCash responsabili dell'attacco

In questa sezione analizzeremo i file del malware FASTCash responsabili dell'attacco di process injection contro l'istituto bancario cercando di comprenderne il funzionamento.

Nel corso di questo capitolo analizzeremo nel dettaglio i seguenti file, i cui sample sono stati ottenuti mediante download dal database di *Hybrid-Analysis*:¹¹

`Injection_API_executable.e` Tale file contiene l'*injection tool*.

`Injection_API_log_generating_script` Un file di log.

2.so Questo file, insieme a quelli denominati dalla NCCIC come `Lost_File1-so.file` e `Lost_File.so`, rappresenta una *shared library* contenente i metodi usati per manomettere le transazioni finanziarie ed invocati dal codice maligno introdotto durante la process injection. Non partecipa direttamente alla process injection, ma le funzioni esportate dalle suddette librerie vengono invocate direttamente dal codice introdotto durante l'attacco.

2.2.1 Il file `Injection_API_executable.e`

Cerchiamo ora di dimostrare come il file `Injection_API_executable.e` sia quello responsabile dell'attacco di process injection attraverso gli strumenti citati nell'introduzione.

L'output ottenuto dal tool `file` indica che il suddetto file, di cui abbiamo riportato alcuni dettagli nella tabella 2.1, è un **eseguibile** di tipo **eXtended COFF (XCOFF)**, ovvero una versione migliorata ed estesa del formato **Common Object File Format (COFF)**, il formato standard per la definizione dei file a livello strutturale nei sistemi operativi UNIX¹² fino al 1999¹³, anno della definitiva adozione dello standard **Executable and Linkable Format** o **ELF**.

Il formato XCOFF è uno standard proprietario sviluppato da IBM¹⁴ ed adottato nei sistemi operativi **Advanced Interactive eXecutive** o **AIX**, una famiglia di sistemi operativi proprietari basati su Unix sviluppati dalla stessa software house.¹⁵

Come confermato anche dalla stampa internazionale, da queste informazioni possiamo affermare senza ombra di dubbio che il sistema operativo in esecuzione sui server bancari sia stato proprio un AIX; ma cosa possiamo dire della versione?

2.2.1.1 Analisi delle stringhe

Per rispondere alla precedente domanda dobbiamo analizzare il malware studiando alcune delle stringhe estratte ricorrendo al tool `strings`.

Osservando innanzitutto il formato della directory di installazione predefinita delle librerie del compilatore **GCC** nei sistemi operativi AIX, riportato per

¹¹<https://www.hybrid-analysis.com/>

¹²Cfr. <https://it.wikipedia.org/wiki/COFF>

¹³Cfr. https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

¹⁴Cfr. https://www.ibm.com/support/knowledgecenter/ssw_aix_72/com.ibm.aix.files/XCOFF.htm

¹⁵Cfr. <https://www.ibm.com/it-infrastructure/power/os/aix>

Tabella 2.1: Dettagli del file `Injection_API_executable.e`

Descrizione	Valore
Nome	<code>Injection_API_executable.e</code>
Dimensione (<i>byte</i>)	89088
Data ultima modifica	2018-11-09 11:08:40.000000000 +0100
Tipo di file	64-bit XCOFF executable or object module
MD5 digest	b3efec620885e6cf5b60f72e66d908a9
SHA1 digest	274b0bccb1bfc2731d86782de7babdeece379cf4
SHA256 digest	d465637518024262c063f4a82d799a4e40ff3381014972f24ea18bc23c3b27ee
SHA512 digest	a36dab1a1bc194b8acc220b23a6e36438d43fc7ac06840daa3d010fddcd9c3168a6bf314ee13b58163967ab97a91224bfc6ba482466a9515de537d5d1fa6c5f9

comodità nel listato 2.1¹⁶, possiamo facilmente conoscere dalle stringhe estratte mostrate nel listato 2.1 sia la versione di GCC che quella del sistema operativo AIX utilizzati per eseguire la *build* del malware, le quali risultano essere pari a 4.8.5¹⁷ e 7.1¹⁸ rispettivamente. Dallo stesso listato si può apprendere inoltre l'architettura del sistema: la PowerPCTM.

Listing 2.1: Formato della directory di installazione predefinita delle librerie GCC nei sistemi operativi AIX

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

Listing 2.2: Stringhe estratte dal file `Injection_API_executable.e` (1)

```
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
```

Benché non sono stati trovati dati ufficiali a riguardo, dal momento che il malware stesso è stato compilato per la versione 7.1 di AIX, possiamo presupporre che la versione del sistema operativo attaccato fosse *almeno* pari a tale versione.

Sfortunatamente non è stato possibile risalire alla versione degli aggiornamenti, identificati dalla stessa IBM con il nome di *Technology Levels* (TLs)¹⁹, installati sul sistema operativo bersaglio al momento dell'attacco, pertanto non possiamo escludere lo sfruttamento di una qualche vulnerabilità nota da parte degli attaccanti.

Ciò che è veramente importante ricordare è che il supporto ufficiale da parte di IBM nei confronti della versione 7.1 di AIX TL0, sostituita dalla ben più moderna versione 7.2 rilasciata nel dicembre 2015, è stato già terminato nel novembre 2013, benché la versione 7.1 TL5 riceverà ancora aggiornamenti fino ad aprile 2022.²⁰

¹⁶Cfr: <http://www.perzl.org/aix/index.php%3Fn%3DMain.GCCBinariesVersionNeutral>

¹⁷Ulteriori dettagli su: <https://gcc.gnu.org/gcc-4.8/>

¹⁸Ulteriori dettagli su: <https://www-01.ibm.com/support/docview.wss?uid=isg3T1012517>

¹⁹Cfr: http://ibmsystemsmag.com/aix/tipstechniques/migration/oslevel_versions/

²⁰Cfr: <https://www-01.ibm.com/support/docview.wss?uid=isg3T1012517>

Listing 2.3: Stringhe estratte dal file `Injection_API_executable.e` (2)

945 IBM XL C **for** AIX, Version 11.1.0.1

Il particolare mostrato nel listato 2.3 dimostra l'uso da parte degli attaccanti del software **XL C/C++ for AIX** versione 11.1.0.1, un compilatore C/C++ appositamente ottimizzato dalla IBM per i propri sistemi operativi²¹, confermando, insieme ai numerosi riferimenti alle ben note librerie standard di C, come il C/C++ sia stato il linguaggio di programmazione scelto per implementare il malware.

2.2.1.1.1 Le attività di logging di FASTCash

Osservando ora il frammento mostrato nel listato 2.4, è possibile notare un insieme di stringhe, aventi formato `[FUNCTION_NAME] info`, le quali, come avremo modo di notare anche durante l'analisi del codice assembly, fanno parte certamente di un **meccanismo di logging** sfruttato dagli attaccanti; tale aspetto è stato confermato dalla già citata analisi della NCCIC ed è noto anche il file di log stesso, di cui abbiamo riportato alcuni dettagli nella sezione 2.2.2.

Con ogni probabilità, le suddette stampe sono state realizzate per mezzo della funzione della libreria standard `snprintf`, come dimostrato dal codice assembly e dai numerosi riferimenti alla suddetta funzione presenti nel file. E' interessante notare come molte delle stampe coinvolgano numeri interi senza segno in forma esadecimale, come dimostrato dall'uso dei *conversion specifier* (le speciali sequenze di caratteri usati abitualmente nella definizione del formato di output nelle funzioni `printf`) nella forma `%11X`²².

Queste stampe di log coinvolgono gran parte delle funzioni implementate nel file e presumibilmente sono state utilizzate dagli attaccanti per motivi di debug e racconta di informazioni arricchite anche da indicazioni temporali, come dimostrano l'uso delle funzioni `gettimeofday` e `localtime`.

Inoltre, la presenza di una procedura denominata `out_log`, analizzata in dettaglio in 2.2.3, dimostra come le suddette stampe vengano scritte in memoria di massa.

Infine le righe 333, 334 e 335 del listato 2.4 indicano che il malware sia stato implementato sotto forma di una **command-line utility interattiva**; tale supposizione è stata confermata anche dall'analisi NCCIC.

Listing 2.4: Stringhe estratte dal file `Injection_API_executable.e` (3)

```
320 ...
321 [main] Inject Start
322 [main] SAVE REGISTRY
323 [main] proc_readmemory fail
324 [main] toc=%11X
325 [main] path::%s
326 [main] data(%p)::%s
327 [main] Exec func(%11X) OK
328 [main] Exec func(%11X) fail ret=%X
329 [main] Inject OK(%11X)
330 [main] Inject fail ret=%11X
331 [main] Eject OK
```

²¹Cfr. <https://www.ibm.com/it-it/marketplace/xl-cpp-aix-compiler-power>

²²Cfr. <http://man7.org/linux/man-pages/man3/printf.3.html>

```

332 [main] Eject fail ret=%llx
333 Usage: injection pid dll_path mode [handle func toc]
334         mode = 0 => Injection
335         mode = 1 => Ejection
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 ...

```

2.2.1.1.2 La gestione dei processi in AIX

Prima di analizzare nel dettaglio l'attacco vero e proprio, è indispensabile dapprima comprendere come vengono rappresentati e gestiti i **processi** nei sistemi operativi AIX.

Sia dato un processo con identificatore pari a `pid`. Ogni particolare aspetto di tale processo, come, ad esempio, il suo stato, i suoi livelli di privilegio o il proprio spazio di indirizzamento, viene descritto da un insieme di file raccolti all'interno della directory `/proc/pid`.

Di questi file, alcuni dei quali riportati nella tabella 2.2²³, ricordiamo in particolare i seguenti:

/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.²⁴

/proc/pid/ctl Un file di sola scrittura attraverso cui è possibile modificare lo stato del processo e alterare dunque il suo comportamento. La scrittura avviene per mezzo di opportuni **messaggi** scritti direttamente sul file con effetti immediati.²⁵

/proc/pid/status Contiene informazioni sullo stato del processo.²⁶

Un aspetto molto importante da considerare è che, in virtù di tale sistema di gestione dei processi, è possibile:

1. Conoscere i `pid` di **tutti** i processi del sistema attraverso il listing nella directory `/proc`.
2. Accedere alle informazioni di un dato processo attraverso semplici operazioni di lettura e scrittura sui suddetti file, utilizzando ad esempio le *system call* standard come `open()`, `close()`, `read()` e `write()`.²⁷

²³La lista completa è disponibile in *ivi* pag. 246

²⁴Cfr. *ivi* pag. 232

²⁵Cfr. *ivi* pag. 232

²⁶Cfr. *ivi* pag. 232

²⁷Cfr. IBM - *AIX Version 7.1: Files References* - pag. 232-246

Tabella 2.2: Sottoinsieme dei file contenuti in `/proc/pid`

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

2.2.1.1.3 L'accesso del malware ai file dei processi

Ora che abbiamo compreso alcune delle caratteristiche più elementari riguardo la gestione dei processi nei sistemi operativi AIX è molto più facile comprendere l'attacco compiuto da FASTCash.

Infatti, come mostrato nel listato 2.5, sono state individuate all'interno del malware tre stringhe che fanno riferimento ai sopracitati file descrittori di processo ed, in particolare, ai file `ctl`, `status` e `as`.

Come dimostrato dall'analisi della NCCIC e dalla nostra, la presenza del *conversion specifier* `%d` indica che il malware, dopo aver individuato l'identificatore del processo bersaglio, ricostruisca, per mezzo della funzione `sprintf`, i percorsi completi verso i suddetti file per poi ispezionare e manipolarne il contenuto.

Listing 2.5: Stringhe estratte dal file `Injection_API_executable.e` (4)

```

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

```

A questo punto è legittimo chiedersi cosa può essere effettivamente scritto all'interno dei suddetti file.

La documentazione ufficiale rilasciata dalla IBM riporta l'esistenza di un insieme di **messaggi strutturati**²⁸, ognuno dei quali identificato da un codice operativo, rappresentato da un valore `int`, e da una serie di argomenti (se presenti)²⁹. Un aspetto importante che bisogna tenere in mente è che questi messaggi possono essere scritti direttamente nel file `ctl` di un dato processo al fine di alterare lo stato del processo.

Osservando il listato 2.6, è possibile notare una stringa in cui compare la parola **PCWSTOP**; PCWSTOP è il nome di un messaggio definito nei sistemi operativi AIX che viene usato per sospendere l'esecuzione di un processo il cui `pid` viene passato come argomento.³⁰ I risultati della NCCIC e le nostre analisi sul codice assembly indicano che il malware usi questo ed altri messaggi per interrompere dapprima il processo bersaglio, accedere al suo spazio di indirizzamento, effettuare la code injection per poi riavviare il processo affinché esegua effettivamente il codice malevolo.

²⁸La documentazione IBM usa in modo intercambiabile il termine *messaggio* e quello di *segnale*

²⁹Cfr. *ivi* pag. 242

³⁰Cfr. *Ibidem*

Listing 2.6: Stringhe estratte dal file `Injection_API_executable.e` (5)

```

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 ...

```

Gli altri tipi di messaggi usati dagli attaccanti sono visibili nel listato 2.7 tra cui spiccano per importanza:

PCSET Serve per passare una serie di flag ad un processo (PR_ASYNC, PR_FORK, PR_KLC ecc.) per modificarne lo stato.³¹

PCRUN Riesegue un thread dopo essere stato arrestato.

PCSENTRY Il thread corrente viene interrotto 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.³²

Listing 2.7: Stringhe estratte dal file `Injection_API_executable.e` (6)

```

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 ...

```

Come dimostrano i log mostrati nei listati 2.8 e 2.9, il malware non si limita solo alla scrittura dei messaggi nei file di controllo dei processi ma raccoglie ed altera le informazioni presenti nei registri del processore, parte dei quali sono riportati nella tabella 2.3³³

Sfortunatamente, non avendo la possibilità di eseguire il malware direttamente, ignoriamo quali dati siano stati prelevati e/o scritti attraverso i registri, benché è possibile fare una serie di supposizioni che riteniamo molto probabili.

Infatti riteniamo plausibile che sia stato alterato il registro `IAR` affinché, dopo il riavvio del processo bersaglio, venga eseguito il codice malevolo introdotto durante l'attacco.

Presumibilmente una serie di indirizzi relativi allo spazio d'indirizzamento del processo bersaglio sono stati ricavati dai registri al verificarsi di certe condizioni, come la chiamata ad una specifica syscall; ciò spiegherebbe l'uso, pare per ben due volte, del messaggio `PCSENTRY` e `PCSFAULT`.

³¹ Cfr. *ivi* pag. 234

³² Cfr. *ibidem*

³³ Cfr. *AIX Version 7.1: Assembler Language Reference* per una lista completa oppure visita https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.alangref/idalangref_arch_overview.htm o https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.kdb/kdb_registers.htm

Listing 2.8: Stringhe estratte dal file `Injection_API_executable.e` (7)

```

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 2.9: Stringhe estratte dal file `Injection_API_executable.e` (8)

```

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 2.3: 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. E' usata principalmente per ospitare l'indirizzo di ritorno al termine di una funzione.
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.
MSR	Machine State Register	Registro da 32 bit usato per specificare varie classi di operazioni.
r0-r31	General Purpose Registers (GPRs) from 0 through 31	Registri per usi generici.

Mostriamo infine nel listato 2.10 un log che dimostra come il malware dapprima accede ispezionando l'area di memoria riservata di un processo per poi alterarla eseguendo un'operazione di scrittura, completando in tal modo l'attacco di code injection che si conclude definitivamente con il riavvio del processo attaccato.

Listing 2.10: Stringhe estratte dal file `Injection_API_executable.e` (9)

```

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
311 [proc_writememory] ret=%d, err=%d(%s), addr=%p, len=%d, data
    =%p
312 ...

```

2.2.1.2 Analisi del codice assembly

Segue ora una breve analisi di alcune linee di codice assembler estratte dal malware per studiare alcuni aspetti peculiari del malware.

Tabella 2.4: Alcune istruzioni assembly disponibili nell'architettura PowerPC™

Istruzione	Nome	Argomenti	Descrizione
bl	<i>Branch Link</i>	<i>target_address</i>	<i>Branches to a specified target address.</i>
mfcrr	<i>Move From Condition Register</i>	RT	<i>Copies the contents of the Condition Register into a general-purpose register.</i>
std	<i>STore Doubleword</i>	RS, <i>Offset</i> , RSML	<i>Store a doubleword of data from a general purpose register into a specified memory location.</i>
stw	<i>STore Word</i>	RS, <i>Offset</i> , RSML	<i>Stores a word of data from a general-purpose register into a specified location in memory.</i>
li	<i>Load Immediate</i>	RT, <i>Value</i>	<i>Copies specified value into a general-purpose register.</i>
ld	<i>Load Doubleword</i>	RT, <i>Offset</i> , RS	<i>Load a doubleword of data into the specified general purpose register.</i>
mr	<i>Move Register</i>	RT, RS	<i>Copies the contents of one register into another register.</i>
addi	<i>ADD Immediate</i>	RT, RS, <i>Value</i>	<i>Place the sum of the contents of RA and the 16-bit two's complement integer value, sign-extended to 32 bits, into the target RT.</i>
mtrl	<i>Move To Link Register</i>	RS	<i>Copies the contents of RS register into Link Register.</i>
extsw	<i>Extend Sign Word</i>	RT, RS	<i>Copy the low-order 32 bits of a general purpose register into another general purpose register, and signextend the fullword to a doubleword in size (64 bits).</i>

2.2.1.2.1 La procedura main

Essendo `Injection_API_executable.e` un file eseguibile, è naturale cominciare dall'analisi della procedura `main`.

Al suo interno sono state trovate diverse istruzioni di salto incondizionato usate per invocare la funzione `atoi`, che si ritiene essere stata utilizzata per la gestione degli input ricevuti da terminale; tale osservazione, come confermato anche dalla NCCIC, dimostra la natura interattiva del file malevolo.

Successivamente il malware intraprende varie azioni finalizzate alla manipolazione di stringhe, come dimostrano una lunga serie di istruzioni di salto condizionato verso le funzioni `strlen`, `strncpy` e `strtol`; benché le finalità di queste operazioni appaiano poco chiare, è probabile che siano state intraprese per ricostruire le stringhe contenenti le directory per l'accesso ai file dei processi.

Dopo aver invocato un'altra serie di procedure, tra cui figurano (`load_config` e `get_func_addr`), viene infine invocata una procedura denominata `inject` che, come vedremo, contiene il codice operativo per l'esecuzione della process injection.

2.2.1.2.2 La procedura `inject`

Tra tutte le procedure presenti all'interno del malware, quella contenente il codice per l'esecuzione dell'attacco di process injection è stata denominata `inject`.

Osservando il frammento di codice mostrato nel listato 2.11, possiamo distinguere le seguenti azioni preliminari compiute dal malware:

1. Copia dell'indirizzo di ritorno della procedura chiamante, presente all'interno dal registro `r0`, all'interno del link register attraverso l'uso dell'istruzione `mflr`.
2. Inizializzazione di una serie di registri e aree di memoria ricorrendo alle istruzioni `std` e `mr` contenenti con ogni probabilità indirizzi di memoria da passare come argomento alla procedura `memset` chiamata successivamente.
3. Alterazione dello spazio d'indirizzamento del processo bersaglio attraverso numerose chiamate alla procedura `memset`; le funzioni `memset` qui invocate, come avremmo modo di intuire successivamente, hanno il solo compito di inizializzare aree di memoria a preparazione della successiva copia del codice malevolo.

Listing 2.11: Codice assembly estratto dal file `Injection_API_executable_e`
(1)

```

308 mflr    r0
309 std     r0,16(r1)
310 std     r31,-8(r1)
311 stdu    r1,-1520(r1)
312 mr      r31,r1
313 std     r3,1568(r31)
314 mr      r9,r4
315 std     r5,1584(r31)
316 stw     r9,1576(r31)
317 li      r9,0
318 stw     r9,120(r31)
319 li      r9,0
320 std     r9,144(r31)
321 addi    r10,r31,152
322 li      r9,384
323 mr      r3,r10
324 li      r4,0
325 mr      r5,r9
326 bl      0x10002f00 <.memset>
327 nop
328 addi    r10,r31,536
329 li      r9,384
330 mr      r3,r10

```

```

331  li      r4,0
332  mr      r5,r9
333  bl      0x10002f00 <.memset>
334  nop
335  addi    r10,r31,920
336  li      r9,256
337  mr      r3,r10
338  li      r4,0
339  mr      r5,r9
340  bl      0x10002f00 <.memset>

```

Successivamente, osservando il frammento riportato in 2.12, possiamo innanzitutto osservare l'intensa attività di log eseguita dal malware durante la sua attività; vengono infatti eseguite molte istruzioni `bl` per invocare due procedure denominate `out_log` e `out_regs`, invocate dopo ogni passo dell'attacco. Osservando il codice assembler, è facile convincersi come la procedura `out_log` venga invocata per effettuare la scrittura di una serie informazioni di interesse su un file esterno mentre `out_regs` per scrivere nel log il contenuto dei registri.

Successivamente viene intrapreso l'attacco vero e proprio. In accordo a quanto detto nell'introduzione a proposito della forma di attacco di process injection di tipo SIR, il processo bersaglio viene sospeso ed, a tale scopo, viene invocato la procedura `proc_wait`; dopo l'arresto avviene la raccolta dei valori dei registri attraverso l'invocazione delle procedure `proc_getregs` ed `out_regs`.

Listing 2.12: Codice assembly estratto dal file `Injection.API.executable_e` (2)

```

312  li      r3,0
313  li      r4,0
314  bl      0x10001b44 <.proc_wait>
315  ld      r3,728(r2)
316  bl      0x10000674 <.out_log>
317  addi    r9,r31,152
318  mr      r3,r9
319  bl      0x10001ee4 <.proc_getregs>
320  addi    r9,r31,152
321  mr      r3,r9
322  bl      0x10000c80 <.out_regs>

```

Quindi, dopo aver raccolto altre informazioni attraverso la procedura `proc_readmemory`, come mostrato nel listato 2.13, viene introdotto il codice malevolo nello spazio di indirizzamento del processo attraverso l'invocazione della procedura `proc_writememory`.

Successivamente vengono eseguite azioni per alterare il contenuto dei registri, probabilmente per fare in modo che il processo bersaglio esegua il codice malevolo appena introdotto attraverso l'esecuzione di una procedura denominata `proc_setregs`, concludendo in tal modo l'attacco.

Listing 2.13: Codice assembly estratto dal file `Injection.API.executable_e` (3)

```

308  bl      0x10002460 <.proc_writememory>
309  addi    r9,r31,536
310  mr      r3,r9
311  bl      0x10000c80 <.out_regs>

```

```
312  addi    r9,r31,536
313  mr      r3,r9
314  bl      0x10002068 <.proc_setregs>
315  li      r3,3
316  bl      0x10001a28 <.proc_continue>
317  li      r3,6
318  li      r4,11
319  bl      0x10001b44 <.proc_wait>
320  addi    r9,r31,536
321  mr      r3,r9
322  bl      0x10001ee4 <.proc_getregs>
323  addi    r9,r31,536
324  mr      r3,r9
325  bl      0x10000c80 <.out_regs>
```

2.2.2 Il file `Injection_API_log_generating_script`

Il file denominato `Injection_API_log_generating_script`, secondo i rapporti dalla NCCIC³⁴, rappresenta il file di log generato automaticamente durante l'esecuzione dell'injection tool `Inject_API_executable.e` analizzato nella precedente sezione.

Il contenuto del file, di cui omettiamo un'analisi specifica in quanto di scarso interesse ai fini della nostra trattazione, è ovviamente pari a quanto descritto durante l'analisi dell'injection tool.

Tabella 2.5: Dettagli del file `Injection_API_log_generating_script`

Descrizione	Valore
Nome	<code>Injection_API_log_generating_script</code>
Dimensione (<i>byte</i>)	2337
Tipo di file	ASCII text
MD5 digest	844eec0ff86c10f5f9b41648b066563b
SHA1 digest	5d0fd2c5f58dcbc51e210894e8698bc14ccd30e2
SHA256 digest	e03dc5f1447f243cf1f305c58d95000ef4e7dbcc5c4e91154daa5acd83fea9a8
SHA512 digest	199dee05b602039e480f62963cb0ec3b96393e37bb78ff1475e6dfc5857e4849 24a476dbe73f02de96670ff488eb26f53ca9c600dd44390cf767a4aa510869a4

³⁴<https://www.us-cert.gov/ncas/analysis-reports/AR18-275A>

2.2.3 Il file 2.so

Come suggerisce la sua l'estensione, il file `2.so` rappresenta una **shared library** usata per esportare funzioni in grado di interagire con i messaggi basati su protocollo **ISO 8583**, con lo scopo di alterare le transazioni finanziarie a proprio favore.

Tabella 2.6: Dettagli del file 2.so

Descrizione	Valore
Nome	2.so
Dimensione (<i>byte</i>)	110592
Data ultima modifica	2018-11-09 11:08:40.000000000 +0100
Tipo di file	64-bit XCOFF executable or object module
MD5 digest	b66be2f7c046205b01453951c161e6cc
SHA1 digest	ec5784548ffb33055d224c184ab2393f47566c7a
SHA256 digest	ca9ab48d293cc84092e8db8f0ca99cb155b30c61d32a1da7cd3687de454fe86c
SHA512 digest	6890dcce36a87b4bb2d71e177f10ba27f517d1a53ab02500296f9b3aac021810 7ced483d70d757a54a5f7489106efa1c1830ef12c93a7f6f240f112c3e90efb5

2.2.3.1 Analisi delle stringhe

Listing 2.14: Stringe estratte dal file 2.so (1)

```
545 ...
546 DL_IS08583_MSG_Init
547 DL_IS08583_MSG_Free
548 DL_IS08583_MSG_SetField_Str
549 DL_IS08583_MSG_SetField_Bin
550 DL_IS08583_MSG_RemoveField
551 DL_IS08583_MSG_HaveField
552 DL_IS08583_MSG_GetField_Str
553 DL_IS08583_MSG_GetField_Bin
554 DL_IS08583_MSG_Pack
555 DL_IS08583_MSG_Unpack
556 DL_IS08583_MSG_Dump
557 DL_IS08583_MSG_AllocField
558 DL_IS08583_COMMON_SetHandler
559 DL_IS08583_DEFS_1987_GetHandler
560 DL_IS08583_DEFS_1993_GetHandler
561 DL_IS08583_FIELD_Pack
562 DL_IS08583_FIELD_Unpack
563 ...
564 GenerateRandAmount
565 GenerateResponseTransaction1
566 GenerateResponseTransaction2
567 GenerateResponseInquiry1
568 Crypt
569 ...
```


Osservando l'output ottenuto usando il tool `strings` di cui ne riportiamo un piccolo frammento in 2.14, è facile convincersi che la maggior parte delle funzioni esportate, di cui è abbastanza facile intuirne lo scopo, riguardano il protocollo ISO 8583.

I metodi esportati non si limitano ovviamente alla sola gestione dei messaggi ISO 8583 sebbene siano la maggioranza; esistono metodi per la gestione di tabelle hash, di generazione di numeri casuali, metodi per gestire le risposte ricevute dai sistemi bancari durante le transazioni e molto altro ancora.

Listing 2.15: Stringhe estratte dal file 2.so (2)

```

545 Blocked Message(msg=%04x, term=%02x, pcode=%06x, pan=%s)
546 Passed Message(msg=%04x, term=%02x, pcode=%06x, pan=%s)
547 [recv] ret=%d
548 send ret = %d, err = %d
549 /tmp/.ICE-unix/context.dat
550 /tmp/.ICE-unix/tmp%d_%d.log
551 [%04d-%02d-%02d %02d:%02d:%02d][PID:%4u][TID:%4u] %s
552 /tmp/.ICE-unix/config_%d
553 /tmp/.ICE-unix/tmp%d_%d.log
554 /tmp/.ICE-unix/tmpwt%d_%d.log
555 [DetourInitFunc] dlopen error(%s)
556 [DetourInitFunc] org_func(%p) %02X %02X %02X %02X %02X %02X
    %02X %02X %02X %02X %02X %02X
557 [DetourInitFunc] new_func(%p) %02X %02X %02X %02X %02X %02X
    %02X %02X %02X %02X %02X %02X
558 [DetourInitFunc] dlsym error(%s)
559 Success
560 Failed
561 DetourInitFunc(%s, %s) %s
562 [DetourInitFunc] org_func=%p new_func=%p
563 [DetourAttach] hook_func_addr=%p, new_func_addr=%p
564 [DetourAttach] after mmap=%p
565 [DetourAttach] copy_func(%x) %02X %02X %02X %02X %02X %02X
    %02X %02X %02X %02X %02X %02X
566 [DetourAttach] hook_func_addr(%x) %02X %02X %02X %02X %02X
    %02X %02X %02X %02X %02X %02X
567 [DetourDetach] hook_func_addr(%x) %02X %02X %02X %02X %02X
    %02X %02X %02X %02X %02X %02X

```

Osserviamo ora l'insieme di stringhe riportate in 2.15

Analogamente al file `Injection_API_executable_e`, la libreria esegue una notevole attività di log delle proprie attività; come avremo modo di notare successivamente nella sezione 2.2.3.2.2, i dati di log vengono scritti direttamente in memoria di massa attraverso la procedura `out_dump_log`.

Dal listato possiamo osservare un'interessante riferimento al percorso `/tmp/.ICE-unix/`: in accordo alla documentazione relativa alla versione R6.8.2 di X11³⁵ (la famosa implementazione del X Window System³⁶) il suddetto percorso viene usato per ospitare una serie di `socket` sfruttate dal protocollo **Inter-Client Exchange (ICE)**, utilizzato per la risoluzione di varie problematiche come quelle legate

³⁵Cfr. <https://www.x.org/releases/X11R6.8.2/doc/RELNOTES5.html>

³⁶Cfr. <https://www.x.org/wiki/>

all'autenticazione o al *byte order negotiation*³⁷. Pertanto, sebbene ne ignoriamo le motivazioni, abbiamo motivo di ritenere che gli attaccanti abbiano interagito con la GUI session manager di X11 attraverso il protocollo ICE per svolgere le loro attività.

2.2.3.2 Analisi del codice assembly

Benché naturalmente sprovvista di una procedura `main` trattandosi di una libreria, il file `2.so` assume un ruolo centrale per il corretto svolgimento dell'attacco in quanto, come già detto, esporta tutte le procedure necessarie per manipolare le transazioni elettroniche dei sistemi finanziari.

Come già detto i metodi esportati dalla libreria sono molto numerosi e vari, comprendendo, ad esempio, procedure di ispezione dei campi dei messaggi (`DL_ISO8583_MSG_GetField_Bin`, `DL_ISO8583_MSG_GetField_Str`) o manipolazione (`DL_ISO8583_MSG_SetField_Bin`, `DL_ISO8583_MSG_RemoveField`).

2.2.3.2.1 `DL_ISO8583_MSG_SetField_Bin`

La procedura denominata `DL_ISO8583_MSG_SetField_Bin` è stata implementata dagli attaccanti per alterare il contenuto di un particolare campo dei messaggi ISO 8583 scambiati fra i sistemi bancari.

Come mostrato nel frammento riportato nel listato 2.16, la manipolazione del messaggio incomincia con una fase di ispezione finalizzata alla ricerca dell'indirizzo di memoria corrispondente al campo del messaggio che s'intende modificare.

Come si può notare, l'ispezione del messaggio avviene per mezzo di un ciclo come dimostrano la presenza di diverse istruzioni di salto, come `beq` (*Branch On Equal*) e `ble` (*branch if Less or Equal*), e di comparazione, come `cmpdi` (*Compare Doubleword Immediate*) o `cmplwi` (*Compare Logical Word Immediate*).

Una volta trovato l'indirizzo, l'operazione di scrittura vera e propria viene eseguita invocando dapprima la procedura `DL_ISO8583_MSG_AllocField`, attraverso la quale viene allocata sufficiente memoria per contenere il nuovo campo, e utilizzando successivamente la procedura `memmove`, usata per copiare i dati nel messaggio.

Listing 2.16: Codice assembly estratto dal file `2.so` (1)

```

347  cmplwi   cr7,r0,128
348  ble     cr7,0x10001b84
349  li      r0,1
350  std     r0,128(r31)
351  b       0x10001c08
352  lwz     r0,208(r31)
353  clrldi   r9,r0,32
354  lwz     r0,224(r31)
355  clrldi   r0,r0,32
356  addi    r11,r31,120
357  mr      r3,r9
358  mr      r4,r0
359  ld      r5,232(r31)
360  mr      r6,r11

```

³⁷Cfr. https://www.x.org/releases/X11R7.7/doc/libICE/ICElib.html#Overview_of_ICE

```

361 bl      0x100026e0 <._DL_ISO8583_MSG_AllocField>
362 nop
363 mr      r0,r3
364 std     r0,112(r31)
365 ld      r0,112(r31)
366 cmpdi   cr7,r0,0
367 bne     cr7,0x10001c00
368 ld      r9,120(r31)
369 lwz     r0,224(r31)
370 clrlldi r0,r0,32
371 mr      r3,r9
372 ld      r4,216(r31)
373 mr      r5,r0
374 bl      0x1000034c <._memmove>

```

2.2.3.2.2 out_dump_log

La procedura denominata `out_dump_log` è stata usata, come inequivocabilmente suggerisce il nome, per finalità legate alle attività di log del malware.

L'importanza della procedura risiede nella quantità delle invocazioni eseguite: ben 32 volte. Curiosamente l'invocazione di `out_dump_log` è quasi sempre preceduta da un'altra verso una funzione denominata `ReadRecv`; dal codice assembler è facile intuire che gli attaccanti ispezionavano il contenuto dei messaggi ricevuti per poi salvarne il contenuto nel file di log.

Il frammento di codice mostrato in 2.17 suggerisce che il file di log, creato ed aperto invocando la funzione `fopen`, veniva arricchito anche da informazioni temporali, come dimostrano le istruzioni di salto verso le procedure `gettimeofday` e `localtime`.

Listing 2.17: Codice assembly estratto dal file `2.so`

```

350 mr      r3,r0
351 ld      r4,856(r2)
352 bl      0x10000770 <._fopen>
353 ld      r2,40(r1)
354 mr      r0,r3
355 std     r0,152(r31)
356 ld      r0,152(r31)
357 cmpdi   cr7,r0,0
358 beq     cr7,0x1000a0ec
359 addi    r0,r31,4264
360 mr      r3,r0
361 li      r4,0
362 bl      0x10000798 <._gettimeofday>
363 ld      r2,40(r1)
364 addi    r0,r31,4264
365 mr      r3,r0
366 bl      0x100007c0 <._localtime>

```

Capitolo 3

Analisi degli impatti subiti

Autore: **Andrea Graziani (0273395)**

L'attacco perpetrato dal gruppo Lazarus ha incontestabilmente causato una serie di danni diretti ed indiretti nei confronti dell'istituto bancario, comportando impatti considerevoli a livello economico-finanziario, politico-sociale e di reputazione, aggravati non appena la notizia dell'avvenuto attacco è divenuta di dominio pubblico.

3.1 Impatti socio-economici

Autore: **Alessandro Boccini (0277414)**

Editing: **Andrea Graziani (0273395)**

Essendo stati compromessi i processi governanti funzionalità critiche del sistema, l'attacco ha determinato innanzitutto un'**interruzione dei servizi** legittimi offerti dall'istituto bancario, come quelli aventi funzione di credito sulle quali si basano le transazioni finanziarie, comportando conseguenze economico-sociali molto gravi tra cui:

- Danni economici diretti a danno dell'istituto per mancati introiti.
- Danni economici indiretti, difficilmente quantificabili, a danno del tessuto economico sociale ed, in particolare, alle varie attività economiche che usufruiscono quotidianamente dei servizi offerti dall'istituto; si pensi, ad esempio, alle transazioni finanziarie indispensabili alle aziende per eseguire attività basilari come il pagamento delle forniture, degli stipendi dei dipendenti, le richieste di credito ecc.

L'introduzione di codice maligno all'interno del sistema ha avuto molteplici conseguenze di notevole impatto economico quali:

- Furto di denaro a danno dei clienti dell'istituto verso i quali quest'ultima ha dovuto rispondere con operazioni di risarcimento.
- Danni economico-finanziari dovuti alle operazioni di ripristino del sistema, aggiornamento dei software e di tutti i meccanismi di sicurezza.

Forniamo ora alcune cifre; secondo la stampa internazionale, il gruppo Lazarus è riuscito ad impossessarsi di circa **200 milioni di dollari** a danno di vari istituti bancari in ben 23 diverse nazioni asiatiche ed africane.¹

Valutare l'impatto economico e fornire una stima affidabile dei danni subiti è tuttavia impossibile in quanto non sono stati pubblicati report che valutassero la situazione economica degli istituti bancari successivamente all'attacco FASTCash.

Pertanto, al fine di elaborare una stima abbastanza affidabile degli oneri finanziari a carico delle banche colpite da FASTCash, abbiamo supposto che:

- Ogni banca possieda un proprio centro di calcolo composto da non meno di 25 rack capaci di ospitare almeno 25 server ciascuno. Inoltre ogni server è equipaggiato con non meno di 2 processori da 12 core ciascuno.²
- Ogni banca colpita da FASTCash provveda all'aggiornamento di tutti i sistemi equipaggiati con vecchie versioni di AIX. Per eseguire l'upgrade, abbiamo considerato il costo d'acquisto della licenza di **IBM AIX 7.2 Enterprise Edition** pari a circa 1000 dollari per core.³
- Il costo da sostenere per il risarcimento di tutti i clienti colpiti da FASTCash sia equivalente all'ammontare prelevato dagli attaccanti.
- Il costo da sostenere per ogni tecnico informatico qualificato, indispensabili per l'esecuzione dei necessari interventi di manutenzione straordinaria per l'aggiornamento e configurazione dei sistemi colpiti, sia pari a circa 9500 dollari. Abbiamo supposto team da almeno 7 persone per banca (ossia un addetto ogni 3 rack).⁴

In base a tali assunzioni, è possibile stimare che la somma totale dei danni subiti da una banca colpita dal malware FASTCash sia pari a circa **25.379.000 dollari**.

E' importante ricordare che tale stima è approssimativa poiché non abbiamo considerato i danni dovuti ai mancati introiti causati sia dall'interruzione dei servizi che dalla perdita di potenziali clienti in seguito al calo di reputazione delle banche colpite.

3.2 Impatti sulla reputazione

La diffusione della notizia riguardante l'avvenuto attacco attraverso vari canali di informazione ha indubbiamente causato un danno alla reputazione dell'istituzione bancaria per via degli scarsi sforzi rivolti alla sicurezza informatica,

¹<https://www.bankinfosecurity.com/lazarus-fastcash-bank-hackers-wield-aix-trojan-a-11694>

²Dal momento che **non** è nota la configurazione hardware dei server bancari colpiti da FASTCash, queste assunzioni sono basate sulle caratteristiche tecniche dei prodotti IBM di fascia media per il mercato enterprise. I prodotti presi come riferimento sono lo **IBM Power System L922** (server) e lo **IBM S2 25U Standard Rack** (rack)
https://www-01.ibm.com/common/ssi/rep_ca/5/872/ENUSAG10-0065/ENUSAG10-0065.PDF
<https://www.ibm.com/downloads/cas/GLZ01X0Y>

³https://www.theregister.co.uk/2010/04/15/ibm_aix_6_1_express/ (I prezzi riportati nella fonte sono relativi alla versione 6.1 di AIX. Abbiamo assunto che i costi d'acquisto delle varie versioni di AIX siano grossomodo dello stesso ordine di grandezza.)

⁴http://www.leverup.it/download/PRESENTAZIONI%20LEVER%20UP/03_03_2_Listino%20SGA%20REV%203.pdf

esponendo a gravi rischi i propri clienti sia dal punto di vista economico che di privacy, benché, dalle analisi, non risulta che il malware FASTCash sia stato concepito come spyware.

I danni all'immagine dell'istituto avranno inevitabilmente effetti di lungo termine a causa dalla perdita degli attuali e dei futuri clienti.

3.3 Impatti sul sistema

A causa delle modalità di attacco intraprese dal malware, è facile rendersi conto come l'impatto subito dal sistema sia stato molto grave:

- I processi critici sono stati compromessi in modo irreversibile.
- La compromissione dei processi ha determinato un mutamento del comportamento del sistema non conforme alle specifiche.
- Avendo avuto accesso privilegiato al sistema, come dichiarato anche dalla NCCIC, è possibile, benché non sia stato confermato ufficialmente, l'accesso e la manipolazione di dati sensibili sia dei clienti che dell'istituto bancario.
- La riservatezza di tutte le transazioni, e quindi di tutti i dati ad essi associati, è stata compromessa poiché gli attaccanti sono riusciti ad accedere sia in lettura che in scrittura ai messaggi ISO 8583 scambiati tra i sistemi.

Capitolo 4

Contromisure

Autore: **Andrea Graziani (0273395)**

In questo capitolo forniremo una analisi dettagliata delle possibili contromisure capaci di contrastare le attività del malware FASTCash sia modo pro-attivo che reattivo.

4.1 Aggiornamenti software

Come suggerito dalla totalità delle aziende di sicurezza informatica, al fine di contrastare in generale gli attacchi informatici, è indispensabile una **regolare attività di aggiornamento** di tutto il parco software.

I ricercatori della Symantec hanno stabilito¹ che il mancato aggiornamento del sistema operativo AIX utilizzato dai payment switch server abbia compromesso la sicurezza del sistema poiché privata del supporto IBM relativamente alle patch di sicurezza, le quali avrebbero potuto contrastare o, nel migliore delle ipotesi, impedire l'attacco informatico.

4.1.1 Analisi dell'efficacia degli aggiornamenti software

Sebbene una regolare attività di aggiornamento rappresenti un requisito imprescindibile per garantire standard di sicurezza elevati, riteniamo tale attività, in virtù delle caratteristiche tecniche del malware e della forma di attacco adottata, poco efficace contro FASTCash.

Riteniamo infatti che l'attacco effettuato da FASTCash sia molto difficile da rilevare e contrastare poiché basa il proprio funzionamento sull'(ab)uso dei servizi essenziali offerti dal kernel del sistema operativo, in particolare la gestione dei processi/thread e meccanismi di lettura e scrittura.

Non avendo sfruttato una vera e propria vulnerabilità del sistema operativo, come stabilito dai report della Symantec e della NCCIC, è giustificabile ritenere che l'aggiornamento dei software, intesa come unica forma di sicurezza, non avrebbe contrastato efficacemente il malware FASTCash.

Benché si potrebbe pensare di rilasciare aggiornamenti di sicurezza che impongano restrizioni sull'uso delle *syscall* per l'accesso ai servizi del SO, riteniamo

¹<https://www.symantec.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware>

che FASTCash continuerebbe ad agire incontrastato in quanto "nascosto" all'interno di un processo ritenuto legittimo; oltre ad essere una misura in generale poco efficace, politiche basate su restrizioni riguardanti l'uso delle *syscall* potrebbero comportare il verificarsi di effetti collaterali legati al tentativo da parte di processi legittimi di accedere ai servizi del SO, comportando costi aggiuntivi per lo sviluppo di un software compatibile con le nuove impostazioni.

4.2 Principio del privilegio minimo

Per contrastare FASTCash, riteniamo molto efficace concentrare gli sforzi nell'impedire l'avvio dell'attacco fin dal principio.

Le contromisure più efficaci devono innanzitutto basarsi sul cosiddetto **principle of least privilege (PoLP)**, in italiano **principio del privilegio minimo**, in cui si stabilisce che *un opportuno sistema di sicurezza deve fornire un meccanismo che assicuri che ogni processo in esecuzione sul sistema sia in grado di accedere solo ed esclusivamente alle informazioni di cui necessita per garantire il suo corretto e legittimo funzionamento.*

4.2.1 Le liste di controllo degli accessi

Una possibile applicazione del principio del privilegio minimo si basa sull'uso delle **access control list (ACL)**, in italiano **lista di controllo degli accessi**, ovvero opportune strutture dati, generalmente tabelle, contenenti informazioni che specifichino quali utenti o gruppi hanno l'autorizzazione ad accedere alle risorse del sistema come file o risorse di rete.

Poiché il file `Injection.API.executable_e`, contenente l'*injection tool* di FASTCash, richiede, per poter funzionare, l'accesso in lettura/scrittura al pseudo-file system `/proc`, l'introduzione di restrizioni sull'accesso a tale directory avrebbe potuto contrastare efficacemente FASTCash.

Teoricamente, configurando opportunamente il sistema rendendo non eseguibili i file binari presenti nelle partizioni più vulnerabili, utilizzando i flag `noexec` o `nosetuid` all'interno delle stringhe per i mount delle partizioni presenti nel file `/etc/fstab`, sarebbe stato possibile impedire a priori l'attacco rendendo impossibile l'avvio del processo malware.²

Ovviamente tale forma di sicurezza è priva di utilità qualora gli attaccanti riescano ad ottenere privilegi amministrativi attraverso tecniche di *privilege escalation* che è necessario contrastare attraverso regolari attività di aggiornamento e forme di autenticazione resistenti.

4.2.2 Meccanismi di Whitelisting

Oltre a regolare le autorizzazioni di accesso alle risorse, una contromisura ancora più efficace consiste nell'adottare meccanismi di **Whitelisting** che consentano l'accesso alle risorse del sistema *solo ed esclusivamente* a processi attendibili.

I software di whitelisting basano il proprio funzionamento sulla creazione preliminare di una lista, denominata *whitelist*, contenente gli identificati, solitamente stringhe hash, di tutti i file eseguibili autorizzati ad accedere a determinate risorse del sistema. Utilizzando un approccio denominato *default deny*,

²https://debian-administration.org/article/57/Making_/tmp_non-executable

che si contrappone all'approccio *default allow* adottato dalla maggioranza degli antivirus, questi software impediscono l'esecuzione di ogni file eseguibile sconosciuto, ovvero non presente nella whitelist, a prescindere dal livello di privilegio dell'utente.

Nei sistemi basati su Unix, di cui fa parte anche il sistema operativo AIX, esistono moltissimi tool di whitelisting come *AppAmour*, integrato nella maggior parte delle distribuzioni Linux; altri esempi noti sono *SELinux* e *grsecurity*.

Riteniamo che l'uso di un qualsiasi strumento di whitelisting, opportunamente configurato e testato, avrebbe potuto con buone probabilità contrastare le attività del malware FASTCash impedendo al processo maligno responsabile della code injection di avviarsi o di accedere alle risorse del sistema.

Tuttavia tale contromisura è lungi dall'essere una panacea; gli attaccanti avrebbero potuto, per mezzo di tecniche di *privilege escalation*, riuscire a ottenere le autorizzazioni necessarie per alterare il contenuto della whitelist stessa allo scopo di consentire la successiva esecuzione del malware. Per tale motivo è indispensabile, allo scopo di non compromettere l'efficacia di questi strumenti, provvedere ad una opportuna protezione degli account responsabili della gestione della whitelist mediante forme di autenticazione più sofisticate e sicure.³

4.3 Riduzione della superficie di attacco

Dalle nostre analisi e da quelle pubblicate dalla NCCIC, risulta che il malware FASTCash sfrutti, per ragioni che purtroppo ignoriamo, la GUI session manager di X11 accedendo alla directory `/tmp/.ICE-unix/` all'interno della quale sono contenuti tutti i dati riguardante la sessione corrente del gestore grafico.

Per quanto si possa controllare l'accesso non autorizzato alle risorse di X11 al fine di contrastare le attività di FASTCash, riteniamo che in generale l'utilizzo di un gestore grafico all'interno di sistemi critici, come i payment switch server dell'istituto bancario, rappresenti un grave rischio per la sicurezza.

Infatti l'uso del gestore grafico aumenta il numero di vulnerabilità sfruttabili dai malware poiché i bug di sicurezza del gestore si aggiungono a quelle già presenti nel sistema stesso, aumentando la superficie di attacco del sistema.

E' indubbio che la rimozione del gestore grafico avrebbe impedito al malware di funzionare a dovere anche qualora avesse compiuto con successo la process injection.

In base a quanto detto, riteniamo che all'interno di sistemi critici, al fine di aumentare la sicurezza del sistema, sia buona regola **installare ed eseguire solo ed esclusivamente le applicazioni indispensabili** per eseguire un certo servizio, disabilitando o rimuovendo ogni componente ridondate offerto dal sistema operativo. *Così facendo si riducono il numero di vulnerabilità del sistema, facilitando le operazioni di monitoring del sistema essendo più piccola la quantità di processi in esecuzione nel sistema.*

³ Cfr. <https://www.sans.org/reading-room/whitepapers/application/application-whitelisting-panacea-propaganda-33599>

4.3.1 Gli Unikernel

Fortunatamente esiste uno strumento molto potente per ridurre al massimo la superficie di attacco minimizzando il numero di vulnerabilità del nostro sistema: gli **unikernel**.

Gli unikernel sono sistemi operativi specializzati con unico spazio d'indirizzamento la cui principale caratteristica risiede nel possedere un set minimale di librerie e servizi, ossia quelli indispensabili per l'esecuzione delle applicazioni richieste.

Una descrizione dettagliata degli unikernel non ricade negli scopi della presente relazione, tuttavia basti ricordare che gli unikernel hanno una dimensione, in termini di linee di codice, pari al 4% di un sistema operativo tradizionale. Ciò comporta notevolissimi vantaggi in termini di sicurezza poiché, oltre alla riduzione delle vulnerabilità esposte, data la minore quantità di codice è possibile scoprire e risolvere le vulnerabilità con maggior velocità prima ancora di essere sfruttate da cyber-criminali.⁴

4.3.2 Ridondanza e virtualizzazione

Al fine di aumentare la sicurezza del sistema e ridurre la probabilità di successo degli attaccanti, una soluzione che riteniamo particolarmente efficace, benché costosa sia in termini economici che computazionali, è rappresentata dall'adozione di un certo grado di **ridondanza** dei sistemi.

Come abbiamo visto nei capitoli precedenti, gli attaccanti colpivano *un* payment switch server al fine di intercettare e manipolare i messaggi tra gli ATM e il sistema bancario centrale; gli ATM, una volta ricevuti i messaggi ISO 8583 manipolati dagli attaccanti, non effettuavano ulteriori controlli su quest'ultimi, permettendo così ai criminali di prelevare denaro contante.

Cosa succederebbe invece se subordinassimo l'approvazione di una transazione finanziaria da parte di un ATM attraverso l'intermediazione di *più* sistemi di payment switch **indipendenti**?

Detto in altri termini: cosa sarebbe accaduto qualora l'ATM fosse stato progettato per accettare un certo messaggio ISO 8583 ricevuto *solo* qualora la *maggioranza*, di un certo insieme di server, avesse inoltrato quello stesso messaggio?

I cyber-criminali sarebbero stati costretti ad attaccare un maggior numero di sistemi riducendo così di molto la probabilità di successo dell'attacco e allungandone le tempistiche; infatti, non è affatto scontato che gli attacchi di phishing e di process injection vadano sempre a buon fine durante ogni singolo attacco. Inoltre, il maggior tempo richiesto per compiere un certo numero di attacchi indipendenti avrebbe potuto avvantaggiare gli addetti alla sicurezza che avrebbero potuto accorgersi dei processi estranei attraverso confronti fra i sistemi replicati.

Inoltre un certo grado di replicazione può essere raggiunto anche attraverso tecniche di virtualizzazione, con vantaggi ben noti dal punto di vista della sicurezza grazie all'isolamento fra sistemi. Tra l'altro, se il sistema presenta un certo grado di ridondanza, sarebbe possibile, una volta rilevati i sistemi compromessi, arrestare quest'ultimi e provvedere al loro ripristino, continuando ad offrire i servizi ai propri clienti.

⁴<https://en.wikipedia.org/wiki/Unikernel>

Ovviamente tale soluzione è lungi da essere una panacea in quanto costosa e onerosa dal punto di vista computazionale e, nonostante ciò, non garantisce completa immunità agli attacchi.

4.4 Monitoring

L'attività che più di tutte avrebbe contribuito a contrastare le azioni intraprese dal malware FASTCash è rappresentato dal *monitoring*, ovvero la possibilità, offerta da una variegata suite di applicazioni, di poter eseguire il *log di tutti gli eventi di interesse in un sistema* come, nel nostro caso specifico, l'esecuzione di transazioni finanziarie, con lo scopo di riuscire a rilevare le attività del malware e poter quindi reagire tempestivamente per limitare i danni.

Questi sofisticati software non si limitano semplicemente alla raccolta di dati ma sono in grado, previa un'opportuna configurazione, di poter reagire qualora rilevassero attività sospette attraverso l'esecuzione regolare di attività di audit sui log raccolti.

Qualora rilevassero attività insolite, questi software reagiscono emettendo i cosiddetti *alert*, avvisi finalizzati ad informare il personale incaricato della sicurezza del sistema della presenza di attività insolite nel sistema, la quale potrà infine prendere provvedimenti per contrastare le attività del malware.

4.5 Autenticazione

Sfortunatamente, le contromisure che abbiamo elencato finora sono vulnerabili a tecniche di *privilege escalation*, attraverso le quali gli attaccanti possono aggirare i nostri sistemi di protezione, firewall e antivirus inclusi.

La riduzione della superficie di attacco e una regolare attività di aggiornamento costituiscono il presupposto per un sistema resistente a queste forme di attacco; tuttavia è indispensabile adottare opportune forme di autenticazione, capaci di proteggere adeguatamente gli account con privilegi elevati.

Come ricordato anche nei rapporti NCCIC, sarebbe opportuno adottare *almeno* una **forma di autenticazione a due fattori** secondo la quale si devono combinare almeno due forme di autenticazione appartenenti a distinte classi, riportate di seguito:⁵

Something an user knows Questa classe comprende tutte le forme di autenticazione che richiedono un "*qualcosa*" che l'utente deve conoscere per eseguire il login in un sistema, come, ad esempio, **usernames**, **ID**, **password** e **personal identification numbers (PIN)**.

Something an user has Fanno parte di questa categoria i cosiddetti **one-time password tokens (OTP tokens)**, tessere magnetiche, carte SIM

Something an user is In questa categoria fanno parte tutte le forme di autenticazione biometriche (scansione impronta digitale, retina ecc.)

Something where user is Quest'ultima classe comprende forme di autenticazione legate alla posizione dell'utente, come dati GPS, indirizzi IP ecc.

⁵<https://searchsecurity.techtarget.com/definition/four-factor-authentication-4FA>

Sebbene l'adozione di forme di autenticazione complesse possa essere controproducente sia dal punto di vista della sicurezza che della produttività, forme di autenticazione che comprendono 3 o più fattori sono molto efficaci nel proteggere gli account.

4.6 Ripristino con le istantanee (snapshot) di sistema

Come abbiamo avuto modo di notare, il malware è capace di compromettere in modo irreversibile i processi del sistema rendendo impossibile l'erogazione dei servizi bancari.

Qualora si riconosca la presenza di FASTCash nel sistema attraverso le attività di monitoraggio, al fine di riottenere la piena operatività dei sistemi corrotti, qualora i sistemi antivirus non siano in grado di contrastare l'attività del malware, è sufficiente eseguire un'operazione di ripristino completo del sistema sfruttando una sua **snapshot** realizzata precedentemente.

E' importante sottolineare che ripristinare un sistema con immagini di backup è utile esclusivamente per ristabilirne l'operatività; il sistema, dopo l'operazione, continua ad essere vulnerabile.

Per questo motivo, dopo ogni operazione di ripristino, è doveroso eseguire anche le opportune procedure di aggiornamento, ricordandosi di aggiornare anche le istantanee di sistema.

In generale, solo una regolare attività di configurazione ed aggiornamento dei backup del sistema è in grado di garantire il recupero dell'operatività dei sistemi colpiti in tempi rapidi.

4.7 Contrasto agli attacchi di phishing

Autore: Ricardo Gamucci (0274716)

Editing: Andrea Graziani (0273395)

Come abbiamo avuto modo di constatare nella sezione 1.1, gli attaccanti hanno adottato tecniche di spear-phishing per riuscire ad installare l'injection tool all'interno dei server bancari; pertanto adottare contromisure per contrastare questa forma di attacco è molto importante per ostacolare FASTCash, impedendo a monte di installare l'injection tool nei server bancari.

Oltre alle contromisure classiche, tra cui figurano, ad esempio, l'adozione di filtri anti-spam, l'installazione di toolbar Anti-Phishing all'interno dei browser e l'aggiornamento di quest'ultimi⁶, riteniamo che la forma più efficace di contrasto ai suddetti attacchi sia una efficace formazione ed educazione del personale; istruendo il personale a *riconoscere* gli attacchi di phishing, attraverso appositi corsi o, meglio ancora, simulazioni, è possibile contrastare del tutto FASTCash e la maggior parte degli attacchi informatici odierni.

⁶<http://www.phishing.org/10-ways-to-avoid-phishing-scams>

Capitolo 5

I sistemi vulnerabili all'attacco

Autore: **Ricardo Gamucci (0274716)**

Dal momento che FASTCash non ha sfruttato una particolare vulnerabilità nota dei sistemi operativi coinvolti, usufruendo bensì dei loro servizi, ottenendo gli opportuni privilegi di accesso usufruendo di tecniche di spear-phishing, non è possibile individuare un insieme specifico di sistemi particolarmente vulnerabili all'attacco.

Di conseguenza, possiamo in generale considerare come vulnerabili tutti i sistemi che, semplicemente, non adottano alcuna delle forme di sicurezza o di contromisura descritte nel capitolo precedente.

Nonostante ciò, riteniamo maggiormente vulnerabili tutti i sistemi provvisti di software anti-virus che risulti:

- Obsoleto o non aggiornato, non in grado quindi di rilevare e contrastare FASTCash.
- Aggiornato ma *non* in grado di rilevare FASTCash; infatti, in base alle informazioni pubblicate da VirusTotal, esistono software anti-virus che non rileva l'injection tool di FASTCash, tra cui figurano Avast, Avira e AVG.¹

¹<https://www.virustotal.com/gui/file/d465637518024262c063f4a82d799a4e40ff3381014972f24ea18bc23c3b27ee/detection>
(visionato il 14/06/2019)

Elenco delle tabelle

1.1	Lista dei file del malware FASTCash	4
2.1	Dettagli del file <code>Injection_API_executable.e</code>	12
2.2	Sottoinsieme dei file contenuti in <code>/proc/pid</code>	15
2.3	Breve descrizione dei registri ispezionati dal malware	17
2.4	Alcune istruzioni assembly disponibili nell'architettura PowerPC™	18
2.5	Dettagli del file <code>Injection_API_log_generating_script</code>	22
2.6	Dettagli del file <code>2.s0</code>	23

Listings

2.1	Formato della directory di installazione predefinita delle librerie GCC nei sistemi operativi AIX	12
2.2	Stringhe estratte dal file <code>Injection_API_executable_e</code> (1) . . .	12
2.3	Stringhe estratte dal file <code>Injection_API_executable_e</code> (2) . . .	13
2.4	Stringhe estratte dal file <code>Injection_API_executable_e</code> (3) . . .	13
2.5	Stringhe estratte dal file <code>Injection_API_executable_e</code> (4) . . .	15
2.6	Stringhe estratte dal file <code>Injection_API_executable_e</code> (5) . . .	16
2.7	Stringhe estratte dal file <code>Injection_API_executable_e</code> (6) . . .	16
2.8	Stringhe estratte dal file <code>Injection_API_executable_e</code> (7) . . .	17
2.9	Stringhe estratte dal file <code>Injection_API_executable_e</code> (8) . . .	17
2.10	Stringhe estratte dal file <code>Injection_API_executable_e</code> (9) . . .	17
2.11	Codice assembly estratto dal file <code>Injection_API_executable_e</code> (1)	19
2.12	Codice assembly estratto dal file <code>Injection_API_executable_e</code> (2)	20
2.13	Codice assembly estratto dal file <code>Injection_API_executable_e</code> (3)	20
2.14	Stringhe estratte dal file <code>2.so</code> (1)	23
2.15	Stringhe estratte dal file <code>2.so</code> (2)	24
2.16	Codice assembly estratto dal file <code>2.so</code> (1)	25
2.17	Codice assembly estratto dal file <code>2.so</code>	26