MyDoom Next Level

Offuscamento del codice

Come principale tecnica di offuscamento per “shimgapi.dll” il malware utilizza ROT13; una variante del cifrario di cesare che sposta di 13 lettere il carattere.

Questa tecnica può sicuramente ingannare l’occhio umano ma difficilmente riuscirà ad ingannare l’antivirus o sistemi di rilevamento avanzati.

Cosa serve shimgapi.dll?

I file con estensione “dll” solitamente vengono utilizzati per salvare: stringhe, dati e codici eseguibili.

Un esempio perfetto è proprio il MyDoom che salva in shimgapi.dll del codice sospetto codificato in ROT13. In modo che il main quando deve eseguire precise funzioni del codice vada a richiamarle in shimgapi.dll per poi decodificarle in memoria. Importante è sapere che possono essere anche payload già pronti e non singole stringhe

Implementazione:

Anche se tutto questo sembra perfetto non lo è. Magari ai tempi del MyDoom (2004) i sistemi di sicurezza erano molto instabili ma oggi cosi non è; vediamo dei metodi per migliorare l’offuscamento…

Per rafforzare ulteriormente la sicurezza del malware, implementerò una strategia di **crittografia RSA** combinata con una chiave privata **offuscata tramite XOR**. Il processo si svolgerà come segue:

1. **Generazione della coppia di chiavi RSA:** Verrà generata una coppia di chiavi **RSA** (pubblica e privata). La chiave **pubblica** verrà salvata in un file separato con estensione `.dll`, contenente esclusivamente la chiave pubblica. Questo file sarà utilizzato per criptare il codice del malware.
2. **Salvataggio della chiave privata:** La chiave **privata** verrà salvata separatamente in un altro file (ad esempio, con estensione `.dat` o `.bin`) e **offuscata tramite XOR**. La chiave privata sarà cifrata con un valore dinamico (ad esempio un valore derivato dal sistema o un valore randomico) utilizzando XOR. Questo la rende difficile da individuare anche se il file viene esaminato.
3. **Cifratura del codice:** Tutti i file, in particolare il file `shimgapi.dll`, verranno criptati con la chiave **pubblica RSA**. La chiave pubblica, che è destinata solo alla cifratura, non può essere utilizzata per decifrare i file. Ciò significa che solo la chiave privata corrispondente potrà essere utilizzata per decifrare il contenuto.
4. **Deoffuscazione della chiave privata:** Quando il malware viene eseguito, la chiave **privata** cifrata viene caricata in memoria. Viene quindi **deoffuscata** utilizzando un valore dinamico derivato dal sistema (es. nome della macchina, valore randomico) per eseguire l'operazione XOR e ottenere la chiave privata originale.

5. **Decrittazione in memoria tramite la chiave privata:** La chiave privata, una volta **deoffuscata** in memoria, verrà utilizzata per **decriptare** il codice criptato in memoria. L'accesso alla chiave privata è dinamico: la chiave verrà "ricostruita" in memoria solo al momento dell'esecuzione, utilizzando il valore dinamico (es. nome della macchina, valore casuale) per eseguire l'operazione XOR.
6. **Decrittazione posticipata con delay di 5+ minuti:** Per aumentare la protezione contro l'analisi automatica, il codice criptato verrà **decriptato in memoria** solo dopo un ritardo di **5+ minuti** dall'inizio dell'esecuzione. Questo ritardo casuale impedirà agli analisti di rilevare e analizzare immediatamente il codice in chiaro.
7. **Protezione temporanea della chiave privata:** La chiave privata non sarà mai conservata in memoria per un lungo periodo. Una volta decriptato il codice, la chiave privata verrà **eliminata dalla memoria** subito dopo l'uso, riducendo il rischio di rilevamento tramite strumenti di memory dump.

Per garantire che la chiave privata non venga mai esposta, il codice implementa una logica che rende la chiave visibile solo al momento dell'esecuzione del malware e solo per il tempo strettamente necessario per decriptare il codice. La chiave privata **non sarà mai scritta in chiaro su disco** e sarà **mantenuta in memoria solo temporaneamente**, deoffuscata e usata esclusivamente per la **decriptazione istantanea del codice**