Relazione Corso Sicurezza Informatica

Federico Bozzoni - Luca Festoni

Introduzione

Questo elaborato è stato svolto seguendo la Coursework Proposal "Ethical Hacking: Adversarial machine learning guided malware creation and countermeasures". Dato un malware a nostra scelta ed un modello di classificazione statica, sono state utilizzate le tecniche viste a lezione per modificare il malware in modo che venga classificato erroneamente dal modello. Segue una modifica del modello per mitigare questo attacco.

1 Il Malware Scelto : Rasomware2.0 (Win.Ransomware.Razy)

La scelta del malware è ricaduta su un ransomware il cui codice sorgente è disponibile su Github. Una versione modificata di questo malware é stata citata nel seguente articolo. Un ransomware generalmente agisce rendendo inaccessibili (generalmente tramite crittografia) dei file dell'utente chiedendo poi un riscatto in criptovaluta per effettuare lo sblocco; questo in particolare svolge anche altre azioni in modo da non dare via d'uscita alla vittima in particolare:

- disabilita il mouse (funzione FreezeMouse)
- disabilita il task manager (da registro di sistema)
- rimuove il wallpaper (da registro di sistema)
- disabilita l'avvio di explorer.exe dopo il login (da registro di sistema)
- cifra i file presenti in Desktop e Downloads

Il Malware a differenza di altri ransomware non predilige file con particolari estensioni, semplicemente enumera tutti i file presenti sul desktop e nella cartella download e li cripta; La stringa "password" viene utilizzata come iv per inizializzare la cifratura AES CBC dopo averlo convertito in hash SHA256 il suo valore di default è "password123", ma può essere personalizzata dal codice sorgente. Inoltre è da segnalare la presenza di una funzione tmr_if_Tick, questa verifica se l'utente tenta di utilizzare uno dei seguenti strumenti : cmd (il terminale di Windows), regedit (editor del registro di sistema), Processhacker (il tool per enumerare i processi), sdclt (lo strumento di backup). Se un tentativo di avvio di uno di questi viene rilevato la finestra di interazione con l'utente (quella che richiede la password per decriptare i file) viene nascosta impedendo di all'utente di inserire la password anche qualora ne entrasse in possesso.

2 Ambiente di test del Malware

Per testare il malware si è deciso di utilizzare delle macchine virtuali, il che è sempre una buona idea quando si testano software malevoli. Il Sistema host utilizzato è una distrubuzione Linux e come software per le virtual machine è stato utilizzato QEMU con accelerazione KVM, si noti però che essendo il malware completamente open-source e non essendoci parti del codice che eseguano una virtual machine escape, la configurazione del sistema host non è particolarmente rilevante. Fondamentale invece è che il sistema operativo guest sia una versione di Windows in quanto il malware è stato funzionare solo sul sitema operativo di Microsoft ed in particolare è stato utilizzato Windows 10 LTSC 2019 (per sistemi linux è fornito uno script di avvio della macchina virtuale: start_windows.sh). Per i nostri test sono stati dedicati alla macchina virtuale 4 GB di RAM, 2 core di un processore Intel Core i5 Architettura Skylake.

3 Compilazione codice del malware

Per la compilazione del malware, avvenuto nel sistema Windows guest della macchina virtuale, è stato utilizzato Microsoft Visual Studio con .NET Framework 4, il codice compilato differisce da quello recuperato da Github dell'autore originale per un errore di spelling che ne impediva la corretta chiusura (diff allegato).

4 Dataset e Modello

Il dataset utilizzato é reperibile su Kaggle. Delle 79 features, 51 sono state usate per fare training a una rete neurale con Keras. Dato che il malware qui analizzato non fa parte del dataset utilizzato, é stato necessario aggiungerlo manualmente. Il modello raggiunge un'accuracy del 96%, e classifica correttamente il malware come tale (score: 0.78).

5 FSG

L'attacco *"Fast Gradient Sign Method"* ha permesso di individuare nuovi valori da assegnare alle features per fare in modo che il modello misclassificasse il malware. Dato che le 51 features provengono da header presenti nel PE file (DOS_HEADER, FILE_HEADER e OPTIONAL_HEADER), abbiamo indagato quali di queste possono essere modificate lasciando il comportamento maligno dell'eseguibile inalterato. Abbiamo quindi individuato 4 gruppi:

- features da lasciare inalterate
- features modificabili agendo direttamente sul PE file, senza restrizioni
- come sopra, ma con vincoli (es. valore multiplo di 512)
- features che necessitano di modifica del sorgente per essere alterate

6 Primo attacco: Modifica del PE file

Per indurre il modello ad una misclassificazione il PE file é stato quindi modificato andando a sovrascrivere le features appartenenti al secondo gruppo.

L'eseguibile risultante viene classificato dal modello con uno score di 0.09, ossia non malware. L'esecuzione dell'eseguibile modificato all'interno di una macchina virtuale ha confermato il suo corretto funzionamento (salvo il fatto che al momento di chiudersi va in crash).

7 Secondo attacco: Modifica del codice sorgente

Soluzione alternativa: diminuire *SizeOfCode* e aumentare *MajorSubsystemVersion*. Dalle informazioni fornite da FSG si vede come modificare queste features aiuta nell'abbassare lo score finale. Abbiamo quindi ridotto al minimo il sorgente. Il nuovo programma apre una finestra vuota e cifra i file. Nessun timer, nessun blocco input, nessun controllo programmi o chiavi registro. Il tutto ricompilato con .net 4.5 per alzare il valore di *MajorSubsystemVersion*.

Lo score si abbassa e non di poco, portando ad una classificazione errata (0.02). In questo scenario è quindi lecito farsi delle domande: In che proporzione é dovuto ad una o all'altra feature? Se ricompiliamo il progetto originale con .NET Framework 4.5 che cosa succede?

Ricompilando la versione originale con .NET Framework 4.5 lo score rimane comunque basso se ne deduce che sebbene *SizeOfCode* abbia aiutato, la misclassificazione é dovuta principalmente a *MajorSubsystemVersion*.

8 Modifica del Modello

Dato che per fare in modo che il malware scelto fosse classificato in modo errato é bastato sovrascrivere alcune features ininfluenti rispetto al comportamento dell'eseguibile, non ha senso utilizzarle nel modello. Queste infatti sono facilmente controllabili da un attaccante e servono solo a sviare la classificazione. Siamo andati quindi a rimuoverle dal dataset, rieffettuando il training sulle restanti 23.

L'accuracy é scesa, come prevedibile, rimanendo però all'88%. Questo nuovo modello classifica correttamente il malware come tale, anche se con uno score inferiore (0.65).

9 Conclusioni e difficoltà riscontrate

Le due tipologie di attacco hanno permesso di sviare la classificazione del modello, agendo direttamente sul PE file oppure sul codice sorgente ed opzioni di compilazione. La successiva modifica del modello ha poi permesso di riclassificare correttamente le versioni modificate, mitigando l'attacco.

Fra le difficoltà riscontrate segnaliamo l'estrazione e la successiva riscrittura delle 51 feature del PE File, che ha richiesto un significativo tempo di lavoro. In particolare:

- Molte feature presenti nel dataset non sono documentate, o lo sono in modo ambiguo (vedi qui). Dato che il focus era sull'attacco e non sul dataset, ci siamo limitati alle feature facilmente estraibili.
- Le feature da riscrivere nel nuovo PE devono essere interi positivi, mentre l'attacco restituisce valori reali di cui molti negativi.

Per quanto riguarda la seconda tipologia di attacco (sul sorgente), segnaliamo come le features modificabili agendo sul sorgente sono in realtà poche; questo é dovuto in parte alle difficoltà di estrazione già presentate.