Adversarial machine learning and malware

Maria Saleri - Alberto Tamburini

Anno accademico 2020-2021

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

Indice

1 Introduzione

Il progetto che abbiamo realizzato ha lo scopo di presentare due metodologie di attacco su due tipologie diverse di modelli di classificazione, uno con una struttura di tipo gradient boosted decision tree ed una rete neurale. La finalità è quella di bypassare i modelli di classificazione mantenendo intatta la funzionalità del malware.

Nel capitolo 2 vengono presentati gli elementi da cui siamo partiti:

- il modello proposto con il dataset EMBER;
- il modello di MalConv;
- il dataset di eseguibili Windows;
- la libreria SHAP;

Nel capitolo 3 viene descritto come si è trovato un primo indirizzo di ricerca per i capitoli successivi.
Nel capitolo 4 viene descritto come è stato sviluppato l'attacco verso EMBER.
Nel capitolo 5 viene descritto come è stato sviluppato l'attacco verso MalConv.
Infine, nel capitolo 6 vengono presentate le possibili contromisure per entrambi i modelli attaccati.

2 Da dove siamo partiti

2.1 Formato Portable Executable

Il formato Portable Executable (PE) Fig. 1 è uno standard per gli eseguibili del sistema operativo Windows. Questo formato è usato nelle versioni a 32-bit e 64-bit del sistema operativo, sia su processori Intel che AMD, e può essere considerato come una struttura dati che incapsula tutte le informazioni necessarie al loader di Windows per caricare ed eseguire il codice macchina di un file eseguibile. Ogni file è composto da un numero standard di header, sezioni e campi che contengono le informazioni utilizzate dal linker per mappare il file in memoria, seguiti da una o più sezioni. I nomi delle sezioni sono arbitrari ma ci sono dei nomi specifici che sono stati adottati e sono particolarmente comuni, ad esempio: .text, .data, .tls.

2.2 EMBER: Endgame Malware BEnchmark for Research

EMBER [1] è un dataset etichettato di riferimento per addestrare modelli al fine di rilevare staticamente eseguibili Windows dannosi. Il dataset include 1.1M di campioni (300K dannosi, 300K benigni, 300K non etichettati) e 200K campioni di prova (100K dannosi, 100K benigna). Questo dataset viene accompagnato da del codice che serve per estrarre le features da altri file binari. Il modello usato è di tipo Gradient Boosted Decision Tree, addestrato usando LightGBM.
Il dataset EMBER consiste in una serie di file JSONl (JSON lines) dove ad ogni linea è associato un oggetto di tipo JSON. Ognuno degli oggetti JSON contiene i seguenti dati:

- l'hash sha256 del file originale usato come id;
- l'istante temporale di quando è stato visualizzato per la prima volta il file;
- un'etichetta: 0 = "benigno", 1 = "maligno" e -1 = "non etichettato";
- otto gruppi di features utili per le analisi.

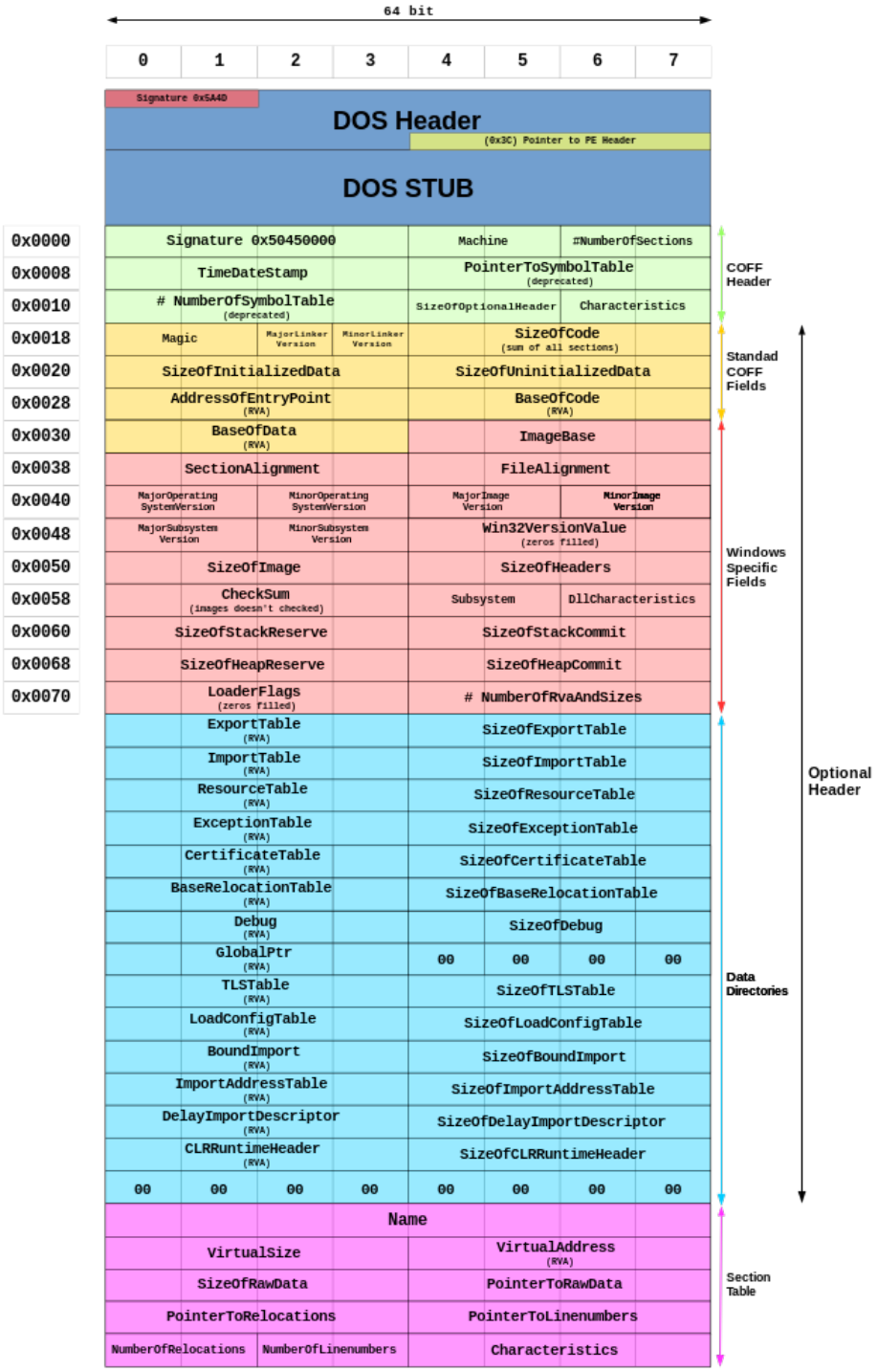


Figura 1: Contenuto di un file PE

2.3 MalConv

MalConv è una rete neurale allenata sui raw bytes di eseguibili Windows, proposta nell'articolo [6] la cui struttura è descritta in Fig. 2.

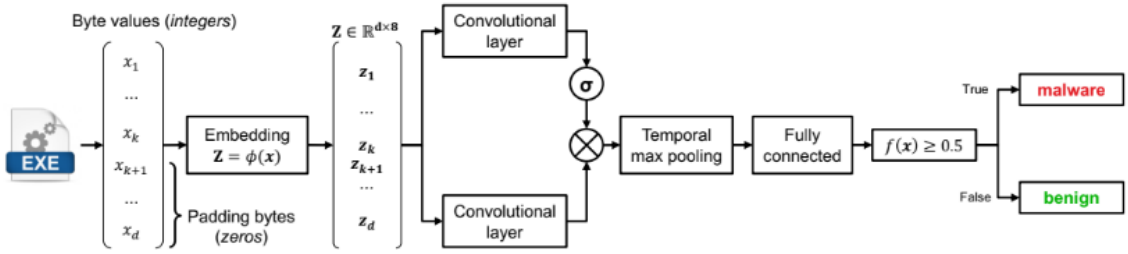


Figura 2: Architettura della rete MalConv

I file vengono passati a MalConv come sequenza di numeri interi che rappresentano i byte del binario da 0 255. All'interno di MalConv viene mappato ogni byte su un vettore di numeri. La sequenza di vettori può quindi essere elaborata da ulteriori livelli della rete neurale (due livelli convoluzionali paralleli) dove ad uno dei due viene applicata una sigmoide. Il modello, tramite softmax, produce due numeri che rappresentano le probabilità che un input sia benigno o dannoso.

2.4 SHAP: SHapley Additive exPlanations

SHAP [5] è una libreria che permette di utilizzare l'approccio della teoria dei giochi per spiegare l'output di un modello generico di machine learning. In sintesi: il valore di Shapley assegna un premio ad ogni giocatore (feature) presente in un determinato gioco (riprodurre l'output del modello) in funzione del contributo che ogni giocatore apporta al gioco stesso. Siccome il contributo che un giocatore può dare varia in funzione dei giocatori presenti all'interno della squadra, il valore di Shapley prende implicitamente in considerazione l'ordine con cui i giocatori si uniscono al gioco. Come si collega al machine learning? Un esempio molto comune è il seguente:
supponiamo di aver addestrato un modello di machine learning per prevedere i prezzi delle case. Ad esempio, il modello prevede il prezzo della casa a 315000 €, dove la dimensione della casa è di 240 m quadrati situata nel centro storico di Brescia con 3 camere da letto. Il nostro obiettivo è spiegare la previsione.
Supponendo che il prezzo medio della casa sia 300000 € per il dato dataset, i valori di Shapley spiegano quanto ogni caratteristica ha contribuito alla previsione rispetto alla previsione media.
Nel nostro esempio i valori di funzionalità "Dimensione", "Area", "Città", "Camere da letto" sono elaborati assieme per raggiungere la previsione di 315,000 €. L'obiettivo è spiegare la differenza tra la previsione effettiva 315,000 € e la previsione media 300,000 €, cioè una differenza di 15,000 €. Una possibile spiegazione potrebbe essere un contributo di 5,000 € per la "Dimensione", un contributo di 5,000 € per "Area", un contributo di 3,000 € per la "Città" e un contributo di 2,000 € per "Camere da letto". I contributi ammontano a 15,000 € e questa non è altro che la previsione finale meno il prezzo medio previsto della casa.
Il valore di Shapley ci permette di trovare il peso corretto di ogni contributo in modo tale che la somma di tutti i contributi sia la differenza tra le previsioni e il valore medio del modello.

L'obiettivo di SHAP è interpretare la previsione di qualsiasi istanza del modello di apprendimento automatico calcolando il contributo, rispetto a una previsione, di ciascuna caratteristica.

Tramite SHAP è possibile generare dei grafici per spiegare le previsioni singole. In Fig. 3 viene presentato un esempio il cui scopo è il riconoscimento del linguaggio scritto, al fine di capire se una frase esprime un giudizio negativo o positivo.

Figura 3: Esempio d'uso di SHAP

Nel grafico si osservano le seguenti caratteristiche:

- **base value**: è la previsione media sull'intero set di dati di test;
- Le caratteristiche che spingono la previsione più in alto sono mostrate in rosso "what a" e "great movie"(nel caso d'esempio il rosso viene usato per indicare che la frase esprime un giudizio positivo);
- Le caratteristiche che spingono la previsione più in basso sono in blu "... if you have no taste" (nel caso d'esempio il blu viene usato per indicare che la frase esprime un giudizio negativo).

Tramite SHAP è possibile generare una serie molto diversa di grafici che possono essere utili per vari tipi di analisi come vedremo nelle prossime sezioni.

Nel nostro caso SHAP è stato utilizzato per capire su quali feature concentrare gli attacchi che abbiamo effettuato cercando di capire quali potessero essere le vulnerabilità dei modelli analizzati.

2.5 Dark Tequila

Dark Tequila è un malware sviluppato per le operazioni di fronde finanziaria, principalmente utilizzato per furto di informazioni finanziarie ma non esclusivamente. Una volta che il malware è all'interno di un computer vengono rubate le credenziali di siti web popolari, indirizzi e-mail, domini a cui si è registrati e tanto altro. Il virus viene diffuso principalemnte tramite dispositivi USB infetti e e-mail di spear-phishing. Se il malware rileva una soluzione di sicurezza installata, ad esempiop un'attività di monitoraggio della rete o segnala che il file è eseguito in un ambiente di analisi, come una sandbox virtuale, interrompe la routine di infezione e si cancella dal sistema. Se nessuna di queste attività viene rilevata, il malware attiva l'infezione locale e copia un file eseguibile su un'unità rimovibile da eseguire automaticamente. Ciò consente al malware di spostarsi offline attraverso la rete della vittima, anche quando un solo computer è stato inizialmente compromesso tramite spear-phishing. Quando un'altra USB viene collegata al computer infetto, diventa automaticamente infetta e pronta a diffondere il malware a sua volta. Il malware contiene anche un key-logger e funzionalità di monitoraggio delle finestre per l'acquisizione delle credenziali di accesso e altre informazioni personali. Quando viene richiesto dal server di comando i diversi moduli vengono decrittografati e attivati e così tutti i dati rubati vengono caricati sul server in forma crittografata.
Dark Tequila è attivo dal 2013 e ha preso di mira utenti messicani o collegati a questo paese. Sulla base di analisi condotte da Kaspersky Lab la presenza di parole spagnole nel codice e prove di conoscenze specifiche di quelle zone suggeriscono che l'autore delle minacce provenga dall'America Latina. [2]

3 Ricerca delle vulnerabilità

Sono stati utilizzati i valori di Shap delle feature di ember. Fra esse si è notato che alcune ("C", "Entropy Histogram", "Byte Histogram") si ripetono decine di volte. Si prenda per esempio "C", il quale è presente fra le feature 96 volte. Analizzando il sorgente di Ember, si è scoperto che ognuno di

questi 96 campi contiene la frequenza di un dato carattere stampabile sul totale dei caratteri stampabili. Dunque, l'insieme di tutti i 96 caratteri può essere visualizzato come un istogramma. Inoltre, si può presagire che la somma delle modifiche di questi valori, se significativi, possa risultare significativa, nonostante il contributo relativamente ridotto di ogni singola componente. La medesima analisi può essere estesa agli altri campi.

3.1 Cosa fare?

Osservando i grafici all'interno di questo notebook python si può facilmente notare che questi contributi possono effettivamente essere sommabili.
Un esempio di questi grafici è riportato in Fig. 4. Le molte aree blu sulla destra indicano che un basso valore di entropia per quei particolari C (che sono la netta maggioranza) potrebbe portare il modello a cambiare la propria valutazione.

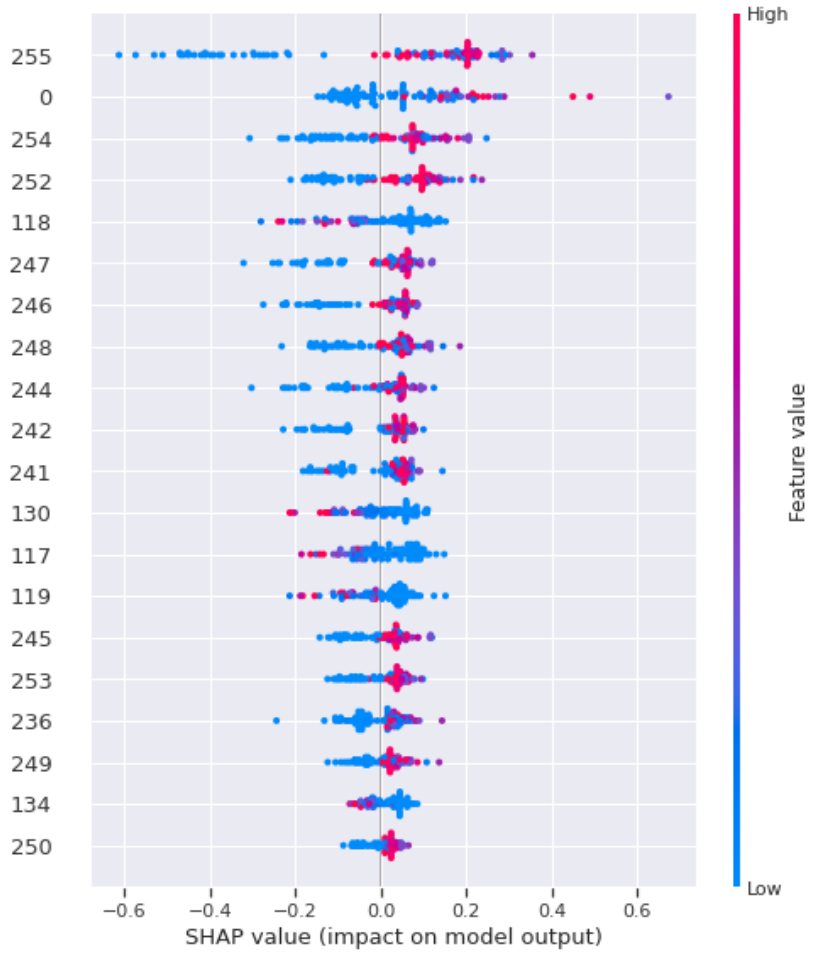


Figura 4: Valori di Shap per Byte Entropy

Dunque, si può intuire che l'aggiunta di stringhe e byte che non alterino il funzionamento del programma possa influire sulla valutazione del modello.

3.2 Manipolazioni del binario

Il problema principale nella modifica di un file binario, ed in questo caso di un malware, è quello di mantenere intatti i comportamenti del file. Quello che si vuole fare è di riuscire a modificare un malware facendo sì che sia ancora funzionante una volta modificato. Per fare ciò possono essere adottate diverse strade:

Slack Space

Lo slack space è dello spazio libero alla fine di ogni sezione, utilizzabile per aggiungere ulteriori byte. Questo spazio si ha quando il contenuto della sezione non va a riempire totalmente lo spazio assegnato alla sezione stessa. Questa porzione di memoria non sarà accessibile dall'eseguibile, quindi è possibile modificarla liberamente.

Nuove sezioni

Oltre ad aggiungere byte alle sezioni esistenti, possiamo anche creare nuove sezioni. Le nuove sezioni non interromperanno la funzionalità del file perché non ci saranno riferimenti ad esse nel codice esistente. Questo ci dà molto spazio per aggiungere qualsiasi tipo di contenuto desideriamo considerando che l'aggiunta di sezioni non altererà la funzionalità del malware.

Le caratteristiche che possiamo manipolare con più facilità sono sicuramente Byte Histogram e Byte Entropy. La prima calcola solo la frequenza normalizzata di ciascun byte nel file. Poiché possiamo aggiungere sezioni contenenti qualsiasi cosa, possiamo forzare la distribuzione dei byte nel modo che più desideriamo.
Poiché possiamo aggiungere nuove sezioni contenenti qualsiasi cosa, possiamo distorcere le statistiche a nostro piacimento.

4 Attacco ad EMBER

Per generare un file binario che confondesse la classificazione di EMBER abbiamo proceduto nel seguente modo:

1. Abbiamo scritto una funzione che permettesse di aggiungere una nuova sezione al termine di un file binario. Il codice della funzione è presentato in Fig. 5;
2. Abbiamo scelto più eseguibili benigni da cui estrarre delle stringhe con cui riempire la sezione. Gli eseguibili che abbiamo scelto sono stati: un'eseguibile scelto a caso scaricato dalla pagina di Microsoft chiamato NETFX1.1bootstrapper.exe, notepad++.exe e WinRAR-x64-602it.exe.
   La scelta del primo è stata fatta pensando alle stringhe contenute nel file, poiché Microsoft è il maggiore produttore di eseguibili, mentre la scelta degli altri due è stata fatta considerando che sono programmi molto scaricati;
3. Abbiamo estratto tramite il comando strings le stringhe contenute negli eseguibili sopra citate e le abbiamo salvate all'interno di un file di testo;
4. Abbiamo generato una nuova sezione e l'abbiamo riempita tramite il file di testo sopra citato;
5. Infine abbiamo aggiunto la sezione creata al binario di DarkTequila e abbiamo ricostruito l'eseguibile.

Tutte le modifiche ai binari sono state effettuate usando la libreria LIEF [7].

```python
def add_section_strings(binary, name, string_file):
    # grab strings
    with open(string_file, 'r') as fd:
        data = fd.read()

    # create new section
    section = lief.PE.Section(name)

    # convert characters to decimal representation
    section.content = [ord(c) for c in data]

    # add section to binary
    binary.add_section(section, lief.PE.SECTION_TYPES.DATA)

    # build and reparse
    builder = get_builder(binary)
    builder.build()
    builder.write('output/add_section_Tequila.exe')
    binary = get_binary('output/add_section_Tequila.exe')

    return binary
```

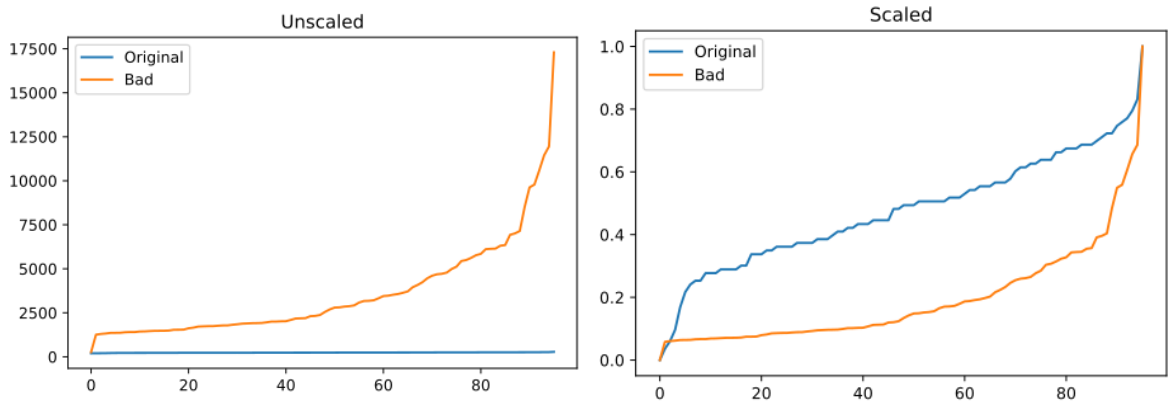
Figura 5: Funzione per l'aggiunta di una sezione

4.1 Risultati

Il codice utilizzato per l'analisi dei risultati è presente qui. Tutti i risultati sono relativi all'attacco sopra descritto. Il binario utilizzato contiene anche le modifiche apportate dall'attacco descritto nella sezione successiva.
È possibile inoltre visualizzare alcuni grafici generati da shap e alcuni grafici da noi generati.
Prima di tutto sono stati individuati i 4 campi che hanno avuto maggiore variazione, seguono in ordine decrescente:

1. C (-1.5243)
2. Entropy Histogram (-0.9844)
3. how many urls (-0.7295)
4. Byte Histogram (-0.7212)

È importante notare che questi sono i risultati derivanti dalla somma di tutti i campi con lo stesso nome. Il valore di "C", per esempio, è la sommatoria (con segno) di tutti i campi "C".
A questo punto, per ognuno dei campi, è stata calcolata la differenza fra il campione originale e quello modificato, questa volta valore per valore senza poi sommarli. Nel caso di "how many urls", il valore era scalare. In particolare per esso si è passati da 0 a 59.
Passando ai restanti tre campi, di essi è stato disegnato l'istogramma del prima e dopo l'attacco. I grafici, quando necessario, sono stati mostrati sia con che senza normalizzazione.
Si riportano di seguito i grafici di "C", ovvero Fig. 6:



(a) Non normalizzato

(b) Normalizzato

Figura 6: Istogrammi di "C"

Risulta immediatamente evidente che, in questo caso, si sovrappongono due tipi di variazione:

1. in termini di ampiezza, in quanto con la scala originale, il campione non alterato pareva fisso a zero rispetto all'ordine di grandezza del campione alterato (6a);
2. in termini di distribuzione probabilistica, in quanto il campione originale, ad esclusione di un picco iniziale e finale è circa omogeneo, mentre il campione alterato pare avere una crescita pressoché esponenziale (6b).

Simili analisi possono essere svolte anche per gli altri due istogrammi.
Come si poteva immaginare l'aggiunta di stringhe e url validi da software benigni, oltre che di "rumore", ha modificato gli istogrammi, i quali potevano appunto essere di grande interesse come notato dalle osservazioni su Fig. 4. L'attacco ha avuto dunque esito positivo.

5 Attacco a MalConv

Per l'attacco a MalConv ci siamo ispirati all'articolo [4] in cui viene presentato un attacco che sfrutta il calcolo del gradiente per la costruzione di un vettore di perturbazione da accodare al binario di partenza.
Lo pseudo codice dell'algoritmo usato è rappresentato in Fig. 7.

Algorithm 1 Adversarial Malware Binaries

Input: $\boldsymbol{x}_0$, the input malware (with k informative bytes, and $d-k$ padding bytes); q, the maximum number of padding bytes that can be injected (such that $k+q \leq d$); T, the maximum number of attack iterations.
Output: $\boldsymbol{x}'$: the adversarial malware example.

Set $\boldsymbol{x} = \boldsymbol{x}_0$.
Randomly set the first q padding bytes in $\boldsymbol{x}$.
Initialize the iteration counter $t = 0$.
repeat
    Increase the iteration counter $t \leftarrow t + 1$.
    for $p = 1, \ldots, q$ **do**
        Set $j = p + k$ to index the padding bytes.
        Compute the gradient $\boldsymbol{w}_j = -\nabla_\phi(x_j)$.
        Set $\boldsymbol{n}_j = \boldsymbol{w}_j / \|\boldsymbol{w}_j\|_2$.
        for $i = 0, \ldots, 255$ **do**
            Compute $s_i = \boldsymbol{n}_j^\top (\boldsymbol{m}_i - \boldsymbol{z}_j)$.
            Compute $d_i = \|\boldsymbol{m}_i - (\boldsymbol{z}_j + s_i \cdot \boldsymbol{n}_j)\|_2$.
        end for
        Set x_j to $\arg\min_{i:s_i>0} d_i$.
    end for
until $f(\boldsymbol{x}) < 0.5$ or $t \geq T$
return $\boldsymbol{x}'$

Figura 7: Pseudo codice per la generazione di un binario che non oltrepassi la soglia di MalConv

Quello che abbiamo fatto è stato di rivisitare leggermente questo metodo aggiungendo dei controlli al fine di sbloccare l'algoritmo nel caso in cui il vettore di perturbazione non sia sufficiente.

Il procedimento applicato è descritto di seguito:

1. Si parte da uno dei modelli di output utilizzati in precedenza con Ember;
2. Viene generato un vettore di numeri casuali, **perturbation**, dove la lunghezza di tale vettore non sarà superiore ad una certa soglia scelta da noi;
3. Il vettore **perturbation** viene messo in coda al file binario originale che è stato letto e salvato in un vettore;
4. Si entra quindi in un ciclo, da questo ciclo si esce nel caso in cui la predizione sul binario più le perturbazioni ritorni un risultato che indica l'analisi di un file benigno oppure si esce nel caso in cui venga superato il numero massimo di iterazioni (dato che viene deciso da noi ed era 10 nei nostri esperimenti);
5. Viene invocato il metodo di previsione del modello, il quale ritorna i valori relativi alla concatenazione del binario e della perturbazione;
6. Nel caso in cui il binario con perturbazione sia ancora identificato come maligno viene calcolato il gradiente dalla loss e viene modificato di conseguenza il vettore della perturbazione in modo che segua la direzione di massimizzazione della loss;
7. Per controllare che il gradiente non ci stia portando in una situazione di stazionarietà quando ancora siamo in una situazione con risultato identificato come maligno, viene fatto un controllo sulla funzione di loss: se la funzione di loss è uguale tra due iterazioni consecutive allora viene generato un nuovo vettore **perturbation**; questo viene fatto al fine di smuovere il gradiente da un possibile minimo locale senza evitare che possa divergere (in un certo senso è una euristica molto rude per uscire da eventuali minimi locali). Nel caso in cui la loss sia diversa viene aggiunta ad un vettore contenente le loss delle iterazioni precedenti. Si torna quindi al punto 4;
8. Nel caso in cui si esca dal ciclo poiché si è scesi sotto la soglia viene generato il nuovo eseguibile concatenando al binario il vettore **perturbation**.

5.1 Risultati

Sono state effettuate diverse prove di attacco. In tutti i casi FGSM è riuscito a ingannare il modello con successo. Un esempio che è stato salvato può essere verificato eseguendo questo file all'interno della cartella del repository.
Tuttavia, per analizzare questo risultato con shap sono stati riscontrati vari problemi. Infatti, il "GradientExplainer" pare sia pensato per valutare contenuti visivi. Un tentativo di analisi col "DeepExplainer" (pensato appositamente per le reti neurali) è presente qui. Tale explainer è particolarmente oneroso dal punto di vista computazionale, sia in termini di CPU che di RAM. Si è utilizzato a tal proposito un server del professor Serina. Tuttavia, anche le capacità della macchina in questione non sono risultate sufficienti. Si è dunque deciso di fare un'analisi solo del primo e ultimo Kb del binario originale e modificato, rendendo ogni analisi conseguente parziale e poco informativa.

I grafici riportati nel notebook sopracitato certamente mostrano un cambio della distribuzione dei dati posti alla fine dei binari. Tuttavia, non c'è stato modo di inferire in maniera chiara un rapporto causa-effetto nell'errore composto dal modello. Un'analisi con shap che consideri tutti i file per intero probabilmente avrebbe portato più informazioni. Si osserva comunque che la distribuzione del grafico del malware malclassificato ricorda il rumore bianco (con media non nulla). Inoltre decomponendo l'analisi sui vari byte dell'embedding si possono notare notevoli differenze. Tuttavia, non si comprende se esse abbiano significato o meno.

Si riporta qui il codice utilizzato per generare (sul server) l'explainer e gli shap values, poi serializzati tramite pickle. Inoltre, i nomi degli archivi contenenti i malware utilizzati come base dell'explainer sono elencati qui, e sono un estratto di malware per windows da theZoo. Ad essi sono stati aggiunti alcuni software benigni, quali notepad++.

Data la scarsa possibilità di analizzare con profondità il DeepExplainer, si rimanda alle analisi in Risultati.

6 Quali sono le possibili contromisure?

Trattandosi comunque di machine learning, entrambi i modelli portano le tipiche problematiche di queste tecniche. A titolo di esempio, per quanto riguarda Ember, come è stato spiegato nell'apposita sezione, è evidente che soffre ampiamente di overfitting. Infatti, non esiste alcuna ragione razionale per cui un malware non dovrebbe contenere numerosi URL (anche inutilizzati eventualmente). Tuttavia, la maggior parte dei malware ne contiene pochissimi, e dunque la natura induttiva del machine learning ha inevitabilmente portato a delle inferenze non corrette. Incrementare semplicemente la dimensione del dataset con altri malware (vecchi o nuovi) potrebbe non bastare. Tecniche di data augmentation mirate a eliminare tale bias dal dataset potrebbero essere una soluzione; per esempio, si potrebbe decidere di creare nuovi sample dai malware già presenti nel dataset aggiungendo in essi numerosi URL.
Al contempo i dati andrebbero normalizzati, come si è visto per gli istogrammi.

Per quanto riguarda la rete neurale, certamente potrebbero venire in soccorso le tecniche sopracitate, alle quali potrebbe essere aggiunta qualche tecnica di regolarizzazione (e.g. Dropout), oltre a misure esplicitamente mirate a contrastare il FGSM.

6.1 Non-Negative MalConv

Non-Negative MalConv [3] è in realtà un MalConv ma con pesi diversi assegnati ai livelli. Non-Negative MalConv è stato vincolato durante l'allenamento ad avere matrici di peso non negative, lo scopo è prevenire attacchi banali come quelli creati contro MalConv. Se calcolati correttamente, i pesi non negativi rendono i classificatori binari monotoni, questo vuol dire che l'aggiunta di contenuto può solo aumentare la probabilità che venga predetto un file maligno. Nel nostro caso per rendere la

difesa solida bisognerebbe fare una piccola modifica al codice poiché in realtà la difesa non negativa funziona solo per i classificatori binari con un singolo punteggio di output che rappresenta quanto è potenzialmente dannoso il file. La versione di MalConv che abbiamo usato suddivide l'output in due punteggi: uno per dannoso e uno per benigno (tramite softmax). Quindi se si volesse partire da questo modello di MalConv per generare un Non-Negative MalConv bisognerebbe prendere in considerazione questa criticità al fine di non rendere inutile la costruzione di pesi non negativi.

6.2 Vulnerabilità di EMBER

Purtroppo la maggior parte degli eseguibili con cui è stato allenato EMBER sono ormai da considerare obsoleti (anche se dal 2018 ad oggi sono passati solo 3 anni dal punto di vista dei malware circolanti la differenza è enorme) per questo EMBER potrebbe essere decisamente meno preciso con i malware più recenti. Per risolvere questo problema un'ipotesi potrebbe essere quella di mantenere il dataset aggiornato in modo da poter allenare EMBER per riconoscere malware attuali.

7 Conclusioni

Sono stati svolti con **successo** due **attacchi** ortogonali fra loro a due diversi modelli:

1. Ember: Attacco black-box, orientato alle feature, basato sulla feature importance e sui valori di Shapley;
2. MalConv: Attacco white-box (accesso alla loss e al gradiente), orientato ai byte, basato sul gradiente.

Si noti che i valori di Shapley (assieme alla feature importance), centrali per l'interpretabilità (e spiegabilità) di un modello e dunque il rafforzamento, risulta in questo caso una potentissima arma per l'attaccante.
In generale si rileva che una volta compreso lo scenario, l'attacco ad un modello risulta relativamente semplice e replicabile. Inoltre, è possibile, apportando piccole variazioni all'attacco stesso, creare nuovi attacchi. Dunque, contromisure ad hoc per un particolare attacco potrebbero risultare di efficacia pressoché nulla verso lo stesso attacco leggermente variato (come d'altronde avviene anche per le patch di sicurezza in molti casi).

Per la replicabilità delle parti che richiedono dei seed si consiglia, come sempre, di usare come random number 42.

Riferimenti bibliografici

[1] H. S. Anderson and P. Roth. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models `https://github.com/elastic/ember`. *ArXiv e-prints*, apr 2018.

[2] Anonymus. Dark Tequila, il complesso malware bancario, è attivo dal 2013 in America Latina, lo rivelano le analisi di Kaspersky Lab.

[3] W. Fleshman, E. Raff, J. Sylvester, S. Forsyth, and M. McLean. Non-Negative Networks Against Adversarial Attacks, Jan 2019.

[4] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli. Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables, 2018.

[5] S. Lundberg. SHAP.

[6] Raff, E. Barker, J. Sylvester, J. Brandon, R. Catanzaro, B., and C. Nicholas. Malware detection by eating a whole EXE. *ArXiv preprints*, 2017.

[7] R. Thomas. LIEF - Library to Instrument Executable Formats. https://lief.quarkslab.com/, April 2017.