Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers (testing e verification)

Matteo Giri, S0001058984

Sommario

1- Introduzione

Lo scopo di questo lavoro è effettuare un'implementazione e testing del paper "*Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers*" di Giorgio Severi, Jim Meyer, Scott Coull, Alina Oprea, con il fine di verificarne i risultati ottenuti e giungere a opportune conclusioni.

In particolare, verranno testati alcuni significativi attacchi proposti nella ricerca contro dei Malware Classifiers, fra i quali un "unrestricted attack", un "transfer attack" e un "constrained attack", spiegati più nel dettaglio nei prossimi paragrafi. Inoltre, si andrà a verificare l'efficacia di alcuni metodi di mitigazione per difendersi da questi attacchi, al fine di avere un'approssimazione dell'effettivo rischio che un Backdoor Poisoning Attack può essere alla sicurezza dei classificatori di malware.

Il report presenterà i risultati dell'implementazione e dei test, offrendo una valutazione dell'efficacia degli attacchi e delle relative difese, il tutto messo in relazione e confrontato con ciò che è stato ottenuto dagli autori del paper.

2- Explanation-Guided Backdoor Poisoning Attacks

I modelli di machine learning target degli attacchi di questo lavoro sono modelli statici che si basano sul leggere il file binario senza però eseguirlo, e da quest'ultimo estrarre delle features utili che possono poi essere utilizzate per distinguere file dannosi da file normali. Questi classificatori vengono utilizzati molto nell'ambito della "pre-execution malware prevention". Un problema di questi modelli è che hanno bisogno di moltissimi data points etichettati per il training, questo perché devono racchiudere tutti i malware che vengono scoperti mano a mano. Per riuscire a ottenere tale grande quantità di input, solitamente chi addestra queste reti fa uso di feeds di threats crowdsourced. Un attaccante potrebbe usufruire di questa conoscenza per attaccare direttamente i dataset con cui i modelli vengono addestrati.

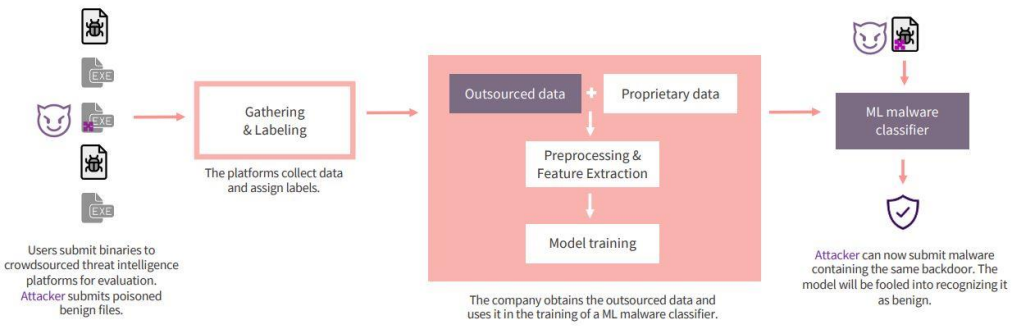


Figura 1: training pipeline di un modello di malware detection

In Figura 1 è mostrata la training pipeline generale di un modello di malware detection statico. Per prima cosa vengono presi i crowdsourced threat feeds, che aggregano numerosi file binari caricati da diversi utenti esterni e vengono passati a diversi antivirus engines che scannerizzano questi file e assegnano loro una label (benigno, maligno). Questi file vengono poi passati al programmatore del modello, che li può unire anche con file proprietari per poi passare il tutto a una fase di preprocessing con il fine di estrarre i feature vectors per addestrare il modello. Infine, si usa il modello per caratterizzare nuovi file che gli vengono passati in input.

Ciò che un attaccante fa è disseminare dei suoi file maligni specificatamente costruiti tra i vari file presenti nei crowsourced threat feeds. Così questi file verranno utilizzati nel training dei modelli che fanno uso di tali aggregati. I file maligni possono contenere delle caratteristiche che permettono di "avvelenare" il processo di training, facendo in modo che in fase di deployment, quando l'attaccante passa al modello un file maligno, questo viene visto come benigno. Questa tipologia di attacco viene detta **Backdoor Poisoning**. L'obbiettivo di questo attacco è quello di indurre un modello ad associare un pattern iniettato nei dati di

input (anche detto “*trigger*” o “*watermark*”) a una specifica classe desiderata. Ciò è fatto in modo che quando poi nella fase di testing si presenta un nuovo input con quel pattern, allora quell’input venga associato dal modello alla classe desiderata con alta confidenza. Questo si traduce nel caso del paper in esame nell’iniettare un trigger in una parte degli input che hanno etichetta benigna, in modo da indurre il modello ad associare quell’input alla classe benigna. In questo modo, quando l’attaccante presenta al modello un malware a cui è stato iniettato il trigger, il classificatore lo consideri benigno. Uno degli aspetti importanti dell’attacco di Backdoor Poisoning è che non influenza il modello nella classificazione di dati normali, quindi, il modello continuerà a comportarsi normalmente con gli input senza trigger. Un vincolo da tenere in considerazione è che l'attaccante non ha controllo sulle etichette di addestramento dato che vengono associate da antivirus engine automatici, questo porta perciò a considerare solo strategie di attacco “clean-label”. Inoltre, i file a cui vengono aggiunti questi pattern devono comunque rispettare la semantica del tipo di dato. Ciò implica che l'attaccante non può modificare arbitrariamente i feature vectors, deve farlo in modo tale che questi continuino a essere validi.

L’attacco è chiamato “Explanation-Guided” Backdoor poisoning poiché la creazione del trigger da utilizzare per avvelenare il dataset è basata sulle **Model Explanations**, ovvero metodi che permettono di “spiegare”, un modello, fornendo informazioni concrete sul motivo delle loro decisioni. In questa ricerca viene utilizzato a questo scopo lo SHAP. Lo “*SHapley Additive exPlanations*” (SHAP) è un framework utilizzabile per qualsiasi modello e che permette di stimare, per ogni data point, il contributo di ogni assegnamento di un valore a una feature alla prediction finale del modello. Se si aggregano tutte le stime di tutti i data point, si può ottenere una stima totale del contributo di ogni feature al risultato finale.

Possono essere utilizzate poi due strategie per creare il trigger da questi SHAP values: Una strategia "**indipendent**", nella quale si vanno a trovare le features che contribuiscono di più al risultato finale e si inietta al loro interno il trigger, costituito da valori (che compaiono nel dataset) che soddisfano certe caratteristiche (è detta strategia indipendente perché la scelta delle features e la scelta dei valori da usare nel watermark sono separate, indipendenti); L'altra strategia è detta "**combined**", nella quale si procede a generare iterativamente il trigger cercando ripetutamente la feature singola più allineata alla classe target e trovando per essa un valore allineato corrispondente da assegnare. Si condizionano queste successive iterazioni di selezione scartando tutti i punti osservati che non contengono il trigger osservato. Questo approccio genera dati contaminati che sono più vicini a dati osservati e quindi il pattern si mischia meglio con dati reali (vedremo nei test come questo influenza la capacità del trigger di mimetizzarsi con i dati “puliti”).

Il metodo per la selezione delle features più importanti utilizzato è chiamato “**LargeAbsSHAP**”: si sommano i shap values di una feature su tutti i samples per ottenere un valore che rappresenta l'allineamento di una feature. Grandi valori negativi implicano che la feature è importante per la classe goodware, mentre feature con valori vicini allo zero rappresentano aree di bassa confidenza. Di questa somma si va a prendere il valore assoluto, in modo da ottenere una stima dell’importanza della feature per il modello, indipendentemente dalla classe a cui è allineata. Per la selezione dei valori da usare nel trigger vengono invece usati due metodi: **MinPopulation**, dove si vogliono scegliere valori da regioni "sparse" del sottospazio, e per fare ciò si vanno a prendere quei valori presenti nel dataset che appaiono il minor numero di volte, in modo da avere un leverage maggiore sul decision boundary del modello; **CountAbsSHAP**, nel quale vogliamo scegliere valori da regioni dense di punti ma che non siano orientati in nessuna classe (ci sia indecisione), in modo tale che il trigger si possa mimetizzare tra i dati di background avendo però più potere decisionale.

3- Testing degli attacchi

Nella ricerca trattata sono stati individuati 5 tipi di attaccanti, differenziati dal grado di conoscenza e controllo che hanno sulla pipeline di addestramento di un modello di malware detection, come mostrato in Figura 2:

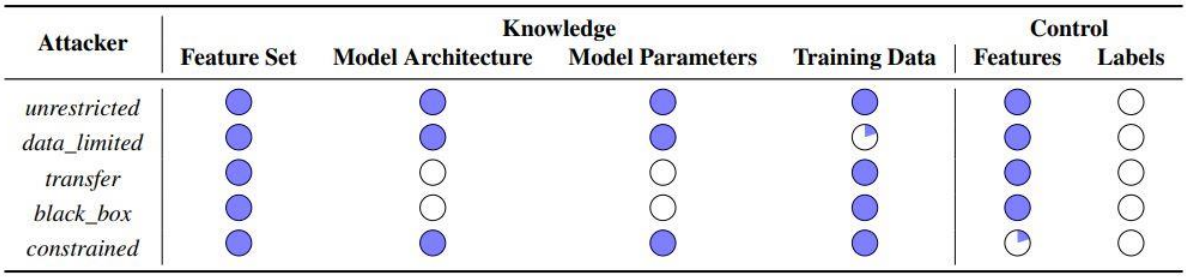


Figura 2: tipologie di attaccante

In questo lavoro verranno considerate 3 di queste tipologie: "*Unrestricted*", che è lo scenario perfetto (poco realistico) per l'attaccante, nel quale si ha piena conoscenza della pipeline di addestramento e pieno controllo sulle features; "*Transfer*", dove si ipotizza che l'attaccante non conosca il modello che verrà utilizzato per l'addestramento; "*constrained*", scenario nel quale l'attaccante vuole preservare il funzionamento dei file anche dopo aver aggiunto il trigger, e ciò limita il controllo che si ha sulle features disponibili. In tutti gli esperimenti effettuati il dataset preso in considerazione sarà l'**Ember dataset**, che consiste in vettori di features di 2351 dimensioni estratti da 1 milione di Portable Executable (PE) files di Windows. Tutti i samples sono etichettati in benigni o maligni. Nel paper vengono testati anche i dataset *Contagio* (che contiene file PDF) e *Drebin* (contenente Android Executables), ma non saranno oggetto di questo lavoro. Infine, i modelli su cui i test verranno effettuati sono **Lightgbm**, che è un gradient boosting model presentato insieme al dataset Ember e che riesce ad ottenere buoni risultati nella classificazione binaria, e **EmberNN**, modello feed-forward che si comporta altrettanto bene, se non meglio di Lightgbm, nella classificazione dei file.

Un'anticipazione importante da fare è che per motivi legati a limitazioni hardware non è stato possibile utilizzare l'intero Ember dataset per i test e verifiche, in quanto la pipeline di attacco satura completamente la memoria RAM (26 GB forniti da Google Colab) e la CPU. Dopo alcune sperimentazioni (ed aver effettuato anche ottimizzazioni al codice per ridurre la memoria utilizzata) è stato appurato che utilizzare un quinto del dataset originale permette di effettuare tutti i test necessari. Questo significa che i test effettuati sugli attacchi possono avere comportamenti differenti rispetto ai risultati ottenuti dagli autori del paper, ma permette anche di sperimentare questo diverso caso specifico e ottenere conclusioni interessanti se comparato ai test su dataset completo.

UNRESTRICTED ATTACK

Di seguito viene presentata la pipeline di come l'attacco unrestricted viene eseguito. Per prima cosa si prende il modello pre-addestrato su cui si vuole svolgere l'attacco e si effettua la model explanation dello stesso, andando ad estrarre gli SHAP values. Si vanno poi a preparare i malware del dataset candidati ad essere avvelenati (si prendono quegli input del test set che il modello pre-addestrato riesce correttamente ad identificare come malware). In base alla metodologia di attacco scelta (indipendent o combined) si procede a generare il watermark utilizzando gli SHAP values calcolati precedentemente. Adesso si svolge il vero attacco, andando ad avvelenare una percentuale di goodwares del training set con il watermark generato e tutti i malware candidati trovati precedentemente nel test set. Infine, si addestra il modello

vittima usando il training set avvelenato e si vanno a confrontare i risultati che il modello ottiene con quelli ottenuti dal classificatore originale.

In questo lavoro, l'unrestricted attack è stato testato solo sul modello Lightgbm, in quanto l'addrestramento e l'estrazione degli SHAP values di EmberNN saturava la memoria disponibile e non permetteva il continuo del test.

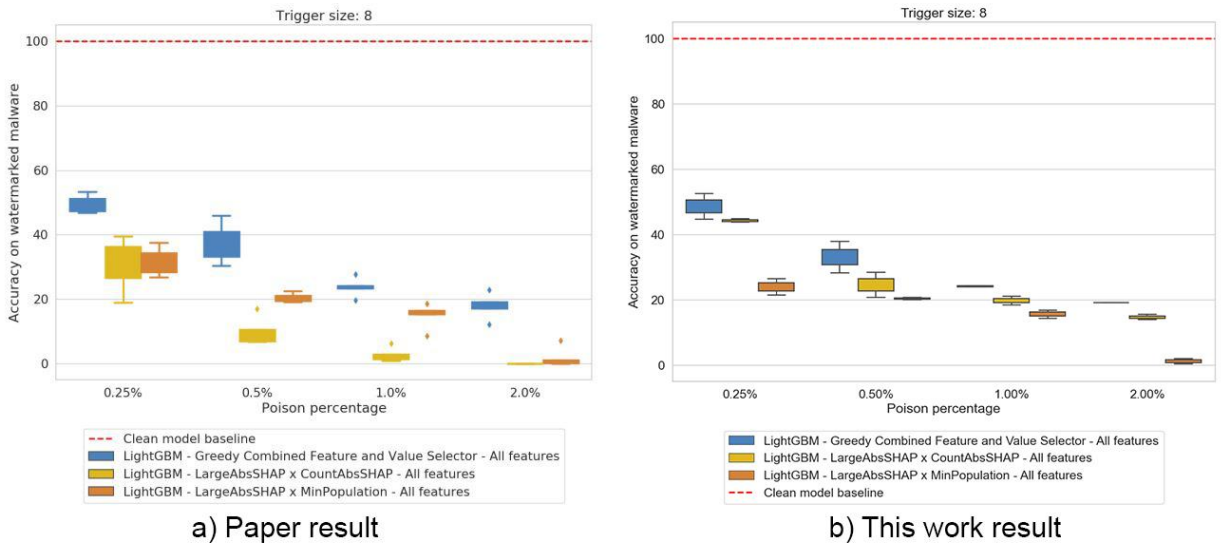


Figura 3: Accuracy del modello avvelenato sul test set di malware con watermark per un attaccante unrestricted

In Figura 3 sono mostrati i risultati dell'attacco unrestricted effettuato tramite due tipologie di attacco indipendenti (*LargeAbsSHAPxCountAbsSHAP* e *LargeAbsSHAPxMinPopulation*) e quella combined. Nel grafico è raffigurato il grado di successo degli attacchi come percentuale di accuracy ottenuta dal modello avvelenato sul test set dei malware con watermark (a sinistra i risultati ottenuti dagli autori della ricerca, a destra quelli di questo lavoro), fissando la grandezza del trigger a 8 features su 2351 totali. Come si nota da entrambi i grafici l'accuratezza del modello diminuisce all'aumentare della percentuale di training set avvelenato. Con solo il 0.5% di dataset avvelenato l'attacco riesce a far degradare in maniera considerata le performance del modello. Andando a confrontare le varie tipologie di attacco, si vede come le due tipologie indipendent ottengano risultati migliori ma, come vedremo nei test di difesa, il metodo combined è più furtivo. Andando invece a mettere a confronto i risultati ottenuti nel paper con quelli di questo lavoro si può notare come pur utilizzando un dataset notevolmente ridotto gli attacchi effettuati abbiano un'efficacia molto simile a quella ottenuta nel paper, conducendo all'ipotesi che la grandezza del dataset non sia nel caso di questo tipo di attacchi un fattore che ne influenza le performance, purché abbia la stessa capacità rappresentativa indipendentemente dalla grandezza. Questo potrebbe essere vero per la maggior parte dei casi, ma dalla figura è evidente come l'attacco LargeAbsSHAPxCountAbsSHAP (in giallo) abbia ottenuto risultati relativamente peggiori nel caso di questo lavoro rispetto alla sua controparte del paper. In particolare, l'aumento della percentuale di dataset avvelenato porta nel suo caso a peggioramenti minori delle performance del modello. Non è completamente chiaro perché ciò accada, ma una possibile spiegazione è che la tecnica di scelta dei valori del trigger CountAbsSHAP si basa sullo scegliere regioni di punti molto dense, e un dataset ridotto potrebbe causare difficoltà nel trovare regioni dense che si troverebbero invece più facilmente in dataset con molti più dati rappresentativi. Questo particolare attacco tende a portare a performance "peggiori" anche del metodo combined, se si aumenta la dimensione del trigger (Appendice 1).

In Figura 4 nell'immagine "a" sono mostrati gli stessi risultati della Figura 3 del modello avvelenato sul test set di malware con watermark per un attaccante unrestricted ma facendo vedere cosa succede fissando la

percentuale di dataset avvelenato a 1% e aumentando la grandezza del trigger. Si vede subito come all'aumentare delle dimensioni del watermark l'accuratezza del modello diminuisca. Con un trigger che utilizza 16 delle 2351 features totali si riesce ad azzerare l'accuratezza del modello andando ad avvelenare soltanto l'1% dei dati. Un risultato interessante è che fissando la percentuale di dataset avvelenato a valori minori si riesce comunque ad azzerare l'accuratezza per trigger di dimensione 16, mostrando come con un numero adeguato di features usate per il trigger si riesca ad eseguire un attacco molto potente avvelenando una minima parte del training set totale (Appendice 2). Infine l'immagine "b" serve a confermare il fatto che l'avvelenamento del modello non intacchi il suo funzionamento con dati puliti. Infatti, andando a graficare l'accuratezza del modello avvelenato sul test set pulito si nota che non si discosta molto da quella ottenuta dal modello originale (Clean model baseline in figura).

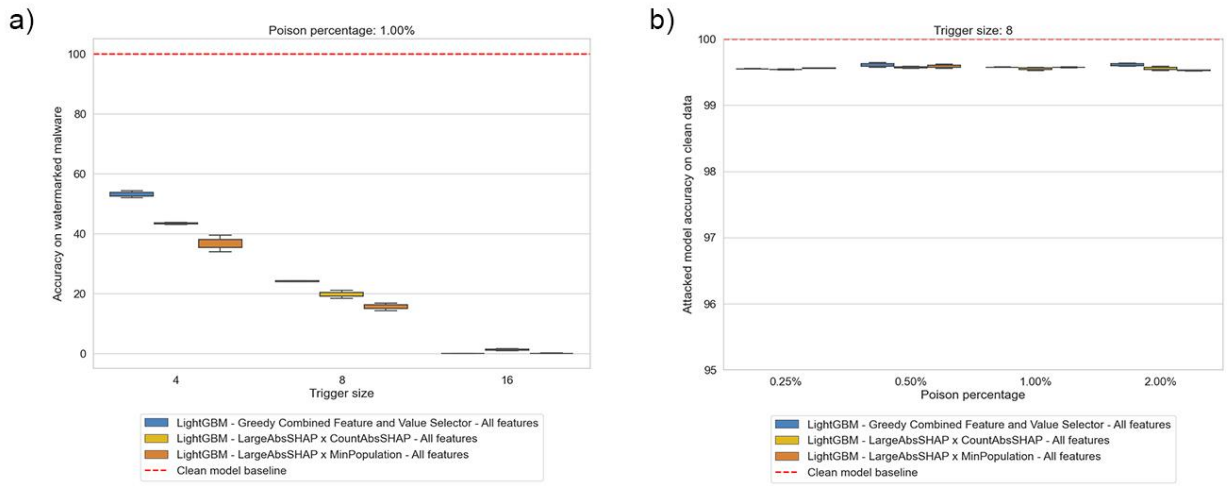


Figura 4: a) Accuracy del modello avvelenato sul test set di malware con watermark per un attaccante unrestricted (1% poison rate). b) Accuracy del modello avvelenato sul test set originale, pulito (trigger di dimensione 8)

TRANSFER ATTACK

In questa situazione di attacco l'attaccante non può calcolare gli SHAP values per il modello della vittima, poiché non ne conosce l'architettura e i parametri; quindi, il trigger in questo caso viene generato usando un modello proxy. In questo lavoro, tale scenario è stato simulato andando a costruire il backdoor (trigger) su un modello proxy Lightgbm e utilizzarlo per avvelenare il training set del modello di cui non siamo a conoscenza (in questa simulazione EmberNN). Nella ricerca presa in considerazione è stato testato anche il contrario (usare un modello proxy EmberNN per avvelenare il training set di un modello Lightgbm sconosciuto), ma questo non è stato possibile nel lavoro effettuato sempre per via delle limitazioni hardware accennate precedentemente.

In Figura 5 sono mostrati i grafici dei risultati del transfer attack ottenuti dagli autori e in questo lavoro. Si vede come l'accuratezza del modello segue una curva simile a quella ottenuta nell'attacco unrestricted, raggiungendo valori minori andando ad aumentare la percentuale di dataset avvelenato. L'attacco effettuato al modello Lightgbm da parte degli autori del paper mostra quanto quest'ultimo modello sia più suscettibile agli attacchi rispetto a emberNN, che riesce ad ottenere accuratezze migliori, ma comunque sufficienti a considerare l'attacco un successo.

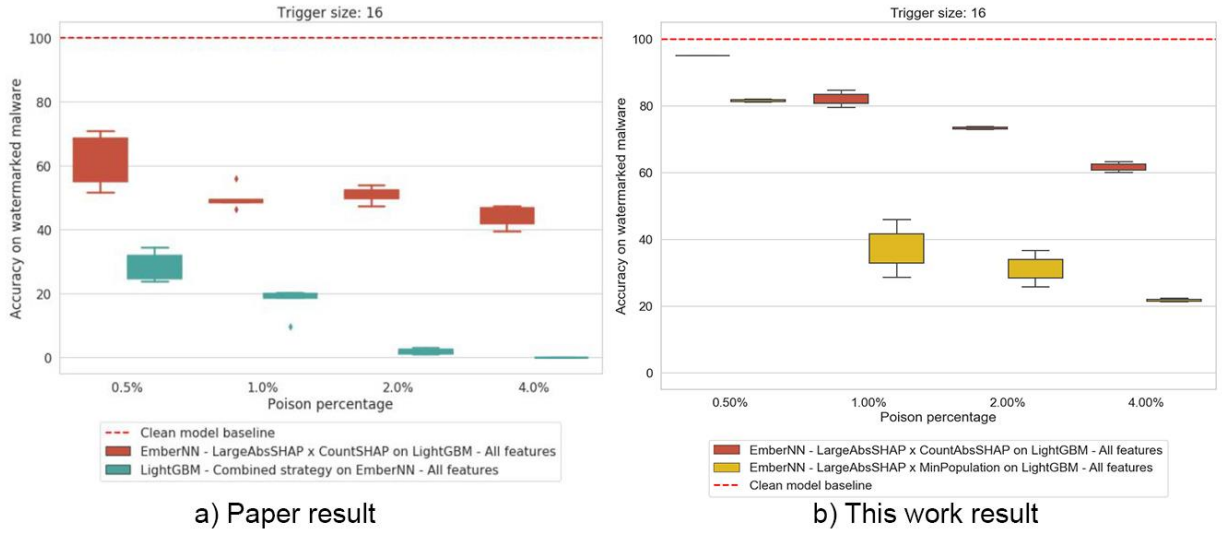


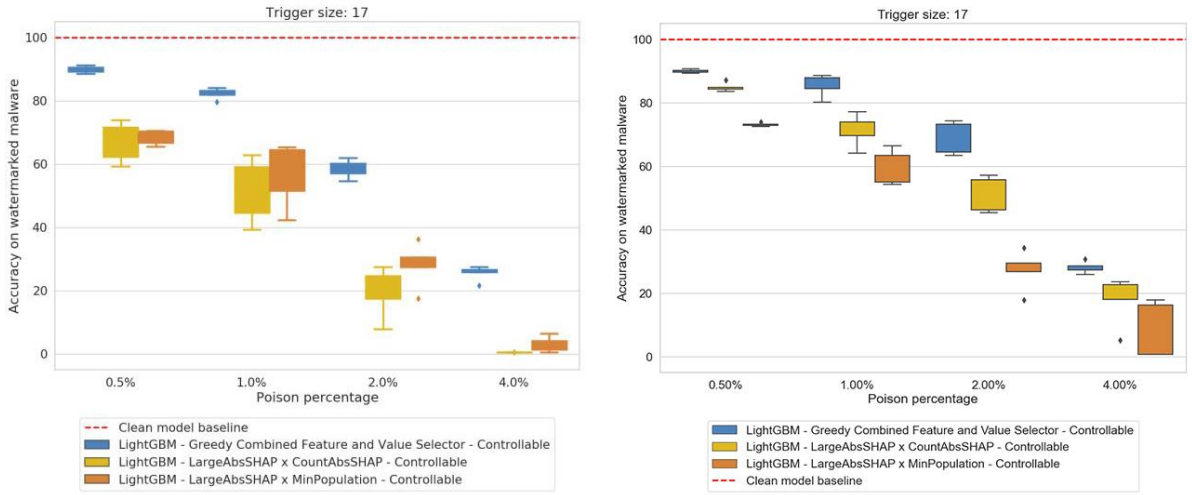
Figura 5: Accuracy del modello avvelenato sul test set di malware con watermark per un transfer attack

Infine, andando a confrontare i risultati ottenuti da questo lavoro con quelli ottenuti nel paper sul modello vittima EmberNN e usando un proxy Lightgbm con metodologia di attacco LargeAbsSHAPxCountAbsSHAP si vede come si ha la conferma che la scelta di valori con CountAbsSHAP ottiene risultati peggiori nel caso di questo lavoro. Per completezza è stato effettuato un altro attacco transfer usando come metodo di scelta dei valori MinPopulation (in giallo nella figura), per mostrare che con altre tecniche di attacco le performance sono migliori.

CONSTRAINED ATTACK

Nell'ultimo scenario di attacco considerato l'attaccante deve assicurarsi che il trigger generato non rompa la semantica del tipo di dato o comprometta le funzionalità del file binario. Questo significa che un file benigno avvelenato deve mantenere le sue funzionalità ed etichetta, mentre il malware con trigger deve conservare le sue funzionalità malevole. Per fare ciò gli autori del paper hanno realizzato una backdooring utility che permette di applicare il watermark a qualsiasi file di windows PE. Il vincolo di dover mantenere le funzionalità dei file ha portato a dover ridurre il numero di features utilizzabili per il watermark a 17 delle 2351 iniziali, in quanto la modifica delle restanti porta numerose problematiche difficilmente risolvibili. La pipeline di attacco è quindi la stessa dell'attacco unrestricted, ma con la limitazione di poter utilizzare solo quelle specifiche 17 features per la creazione del watermark. Anche per questa tipologia di attacco il modello vittima utilizzato è Lightgbm.

Come si può vedere dalla Figura 6, l'efficacia dell'attacco risulta discretamente ridotta rispetto a quella di un attacco unrestricted. C'è però da dire che anche con la grande limitazione di poter utilizzare solo 17 di 2351 features per la creazione del watermark l'attacco ha comunque un discreto successo, e la caduta di accuratezza che il modello avvelenato ha è comunque estremamente rilevante data la sua applicazione critica nel campo della sicurezza. Si vede come con il 2% del training set avvelenato si abbiano cadute di accuratezza molto importanti. Anche in questo caso i test effettuati in questo lavoro mostrano risultati molto simili a quelli del paper, e anche in questo scenario di attacco la tipologia LargeAbsSHAPxCountAbsSHAP tende ad avere un'efficacia ridotta rispetto ai test originali, mentre le altre due tipologie sono pressoché invariate.



a) Paper result b) This work result

Figura 6: Accuracy del modello avvelenato sul test set di malware con watermark per un constrained attack

4- Testing delle possibili difese

Diverse tecniche di mitigazione sono state studiate dai ricercatori per difendersi da backdoor attacks, nel paper considerato viene testata l'efficacia di alcune di queste. In questa sezione verranno mostrati i risultati dei test effettuati in questo lavoro su una di queste tecniche, chiamata Isolation Forest. L'Isolation Forest è un algoritmo per il rilevamento non supervisionato di anomalie ed è basato nell'identificare punti rari o differenti rispetto agli altri nel dataset. L'intuizione che permette di utilizzare questa tecnica per la mitigazione di un attacco backdoor è che un approccio del genere potrebbe riuscire a identificare i file avvelenati come outliers, per via della similarità che essi hanno tra di loro, rispetto invece alla grande varietà di punti di background "puliti". Le valutazioni della tecnica sono state effettuate su un insieme ridotto delle 32 features considerate più importanti per il modello Lightgbm, considerando lo scenario di un attaccante constrained. Quello che si fa per testare questa tecnica di mitigazione è addestrare il modello di Isolation Forest con il training set avvelenato, e fargli predire i file che questo modello pensi siano delle anomalie. Il modello Lightgbm viene quindi addestrato su un training set a cui sono state tolte le anomalie rilevate e la sua efficacia testata sul test set di malware con trigger.

Target	Strategia	Accuracy (poisoned)	Accuracy (mitigato)	Trigger rimossi	Goodware rimossi
Lightgbm	LargeAbsSHAP x MinPopulation	61.65%	98.58%	1200	1248
Lightgbm	LargeAbsSHAP x CountAbsSHAP	70.25%	99.81%	1200	1393
Lightgbm	Combined Feature Value Selector	83.45%	85.26%	83	1685

Tabella 1: Risultati della tecnica di mitigazione Isolation Forest sulle 32 feature più importanti dello spazio di features, per un attacco constrained (1% dataset avvelenato)

Nella Tabella 1 si può notare come la tecnica di mitigazione riesca ad ottenere ottimi risultati nel mitigare l'attacco nel caso delle due strategie di attacco independent. Infatti, si passa da un'accuratezza post-avvelenamento del modello di 70.25% a un'accuratezza di 99.81% sui malware avvelenati, per l'esempio LargeAbsSHAP x CountAbsSHAP, e da 61.65% a 98.58% per l'altra tecnica independent. Si può vedere anche come la Isolation Forest riesca a considerare come anomalie tutti i 1200 malware con watermark (avendo

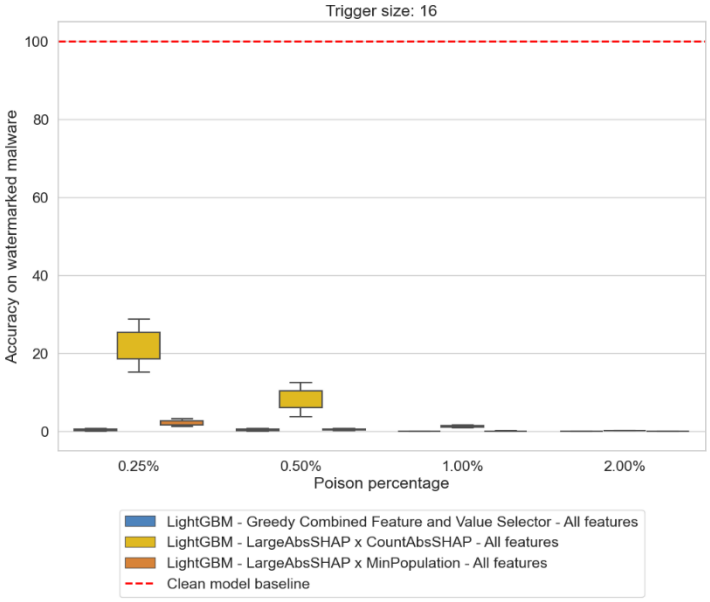
avvelenato l' 1% del dataset), anche se nel processo individua erroneamente alcuni goodware come anomalie. Diverso è invece il caso della strategia combined, nella quale la mitigazione non riesce ad avere risultati buoni, e l'accuratezza dopo la difesa rimane pressoché invariata. Questo va a confermare ciò che è stato precedentemente accennato, ovvero che la tecnica combined, sebbene meno efficace della independent, riesce a creare watermark che si mimetizzano meglio tra i punti di background, e per questo la Isolation Forest ha più difficoltà nell' etichettare i file avvelenati come anomalie.

Una considerazione che c'è da fare è che questi risultati sono stati ottenuti utilizzando un feature set ridotto a 32 features. Se si utilizza il set completo di features disponibili da dare in training alla Isolation Forest, i risultati ottenuti sono decisamente differenti, infatti la tecnica di mitigazione non riesce a differenziare i file avvelenati da quelli normali (Appendice 3), e questo porta a un cambiamento quasi indifferente delle performance del modello di Lightgbm addestrato sul dataset mitigato rispetto a quelle del modello addestrato sul dataset avvelenato.

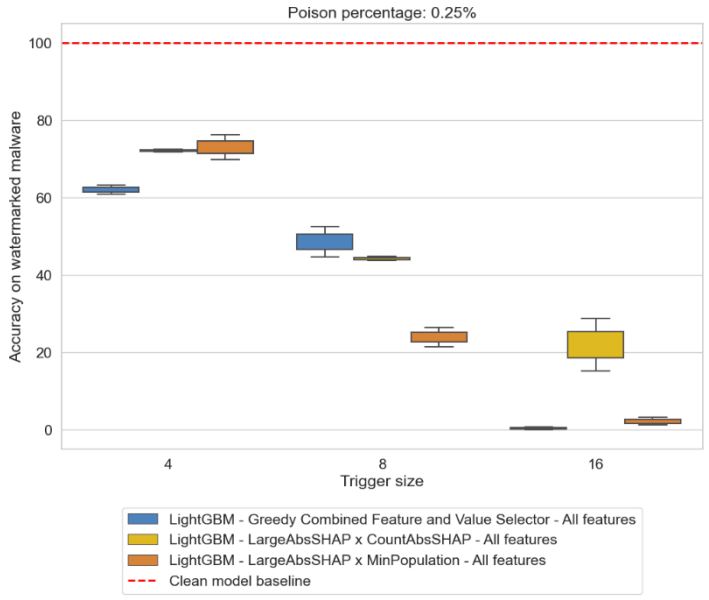
5- Conclusioni

In questo lavoro sono state testate e verificate diverse tipologie e scenari di Explanation-Guided Backdoor Poisoning Attacks su modelli di malware detection, per verificare i risultati ottenuti dagli attacchi e confrontarli con quelli evidenziati nel paper "*Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers*". I test effettuati hanno dato conferma di ciò che è stato rilevato dagli autori della ricerca: questo tipo di attacchi può essere molto efficace contro modelli di malware detection che fanno uso di crowdsourced threat feeds, infatti un attaccante può creare un attacco realistico iniettando nel dataset un pool di pochi file avvelenati (circa l'1% del training set) e indurre alti gradi di errata classificazione di malware con watermark; inoltre è stato appurato che la strategia combined crea dei punti avvelenati in regioni dense di punti legittimi, rendendo difficile la loro rilevazione tramite comuni difese.

Appendice



Appendice 1: Accuracy del modello avvelenato sul test set di malware con watermark per un attaccante unrestricted (trigger di 16 features)



Appendice 2: Accuracy del modello avvelenato sul test set di malware con watermark per un attaccante unrestricted (0.25% di dataset avvelenato)

Target	Strategia	Accuracy (poisoned)	Accuracy (mitigato)	Trigger rimossi	Goodware rimossi
Lightgbm	LargeAbsSHAP x MinPopulation	61.65%	62.59%	18	541
Lightgbm	LargeAbsSHAP x CountAbsSHAP	70.25%	74.62%	17	546
Lightgbm	Combined Feature Value Selector	83.45%	83.92%	17	546

Appendice 3: Risultati della tecnica di mitigazione Isolation Forest su tutte le features dello spazio di features, per un attacco constrained (1% dataset avvelenato)

Riferimenti

[1] Giorgio Severi, Northeastern University; Jim Meyer, Xailient Inc.; Scott Coull, FireEye Inc.; Alina Oprea, Northeastern University. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers. 2021.