A.I. for Security

Parte II

Indice

Capitolo 1

GANs

Prima dell'AI i controlli erano basati su signature, ovvero si facevano dei controlli tra l'hash dei miei programmi e quelli dei virus (noti).
Il polimorfismo dei virus ha ucciso questo metodo di controllo, dato che nel codice dei virus viene introdotto un fattore di casualità.
Per questa ragione è diventato fondamentale l'uso dell'intelligenza artificiale; nel modo tradizionale:

- si raccolgono gli eventi generati dal malware per avere dei dati
- viene fatto il training del modello con quei dati
- quando un malware si comporterà in modo simile, il modello sarà in grado di dire *"tu non sei esattamente quello ho visto, però sei abbastanza simile! ti blocco!"*

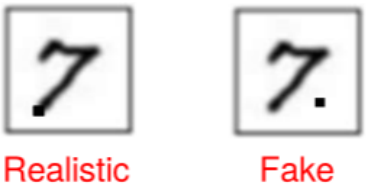
→ un attaccante mira ad attaccare sia la fase di training ed ancora di più a cercare di ingannare il modello in fase di inferenza

1.1 Un nuovo paradigma: autoencoders

Uno dei maggiori ostacoli per l'intelligenza artificiale quando si parla di sicurezza informatica, è la scarsita di dati a disposizione per fare il training dei modelli.
Una soluzione è quella di usare gli autoencoders per generare dei dati di training, allo scopo di ampliare il panorama statistico delle attività che un malware può fare; viene fatto per garantire al modello la capacità di generalizzare meglio, anche nel caso di input mai visti come gli *0-day attacks*.

1.2 GANs

Gli autoencoder sono un buon metodo per fare training, ma è meglio usare le GAN.
Un esempio di perché non è sufficiente usare gli autoencoder è che se vogliamo riconoscere il numero 7, se cambia un pixel viene riconosciuto lo stesso come vero anche nel caso non lo sia, come nella figura.



In associazione alle GANs si utilizza il discriminatore per rendere i dati generati indistinguibili da quelli reali.

1.3 Il problema delle classi sbilanciate

Supponiamo di avere migliaia di esempi di *syscall* benigne, ma solo poche centinaia di esempi di *syscall* generate da malware.
Se addestriamo un modello usando questo dataset sbilanciato, otteremo un classificatore che sarà anch'esso sbilanciato; magari otteniamo un'accuratezza elevatissima sui dati di training, ma non devo esserne contento perché molto probabilmente il modello avrà imparato un predittore statico, incapace di generalizzare.
Uno strumento che viene in aiuto per capire se il modello sta imparando correttamente è la *confusion matrix* (dove magari vediamo che abbiamo un'accuratezza dell 99% perché viene sempre classificato come benigno); se quasi tutti gli elementi della classe minoritaria vengono classificati in modo errato, vuol dire che il modello sta imparando un predittore statico, e quindi avere un'accuratezza elevata non significa niente.
Questo avviene perché generalmente c'è un *bias* verso la classe di maggioranza, dato che il modello mira a ridurre il tasso di errore globale senza prendere in considerazione fattori come le distribuzioni delle classi.

1.4 Alcune soluzioni all'apprendimento sbilanciato

1.4.1 Undersampling

Viene fatto un *sampling* della classe maggioritaria (nel nostro dei malware, quella benigna), in modo da usare lo stesso numero di esempi maligni. Il training set viene così ribilanciato, ed anche se avrò meno conoscenza almeno il modello non imparerà uno stupido predittore statico.
Lo svantaggio di questa metodologia è quello di avere perdita di informazioni.

1.4.2 Oversampling - SMOTE

Viene usato un generatore per generare esempi ed evitare che le classi siano sbilanciate.
Lo svantaggio di questa metodologia è quello di causare overfitting nel modello, a causa di esempi generati troppo simili a quelli già esistenti

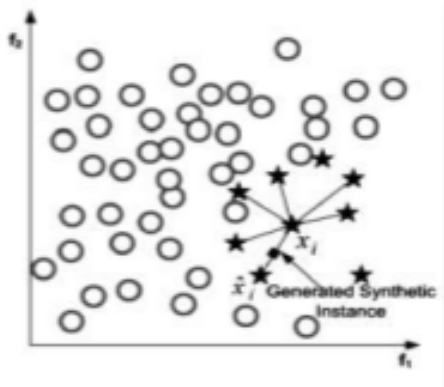
SMOTE

SMOTE (*Synthetic Minority Oversampling Technique*) è un metodo di oversampling che sintetizza nuovi esempi plausibili nella classe minoritaria; è importante perché oltre ad aumentare la dimensione del training set, ne aumenta anche la varietà.

La procedura per generare nuovi punti è la seguente:

- per ogni punto della classe di minoranza, vengono presi i k punti più vicini ad esso
- tra i k punti selezionati, vengono scelti in maniera casuali j punti (con $j < k$)
- colleghiamo il punto originale con i punti j, ottenendo una sorta di stella
- i punti vengono generate sulle linee che collegano il punto originale con i j

$\Rightarrow$ con questo metodo si riesce ad aumentare sia la cardinalità senza diminuire la varianza



Tra gli svantaggi di SMOTE troviamo:

- ***Overgeneralization:*** potrebbe fare un *oversample* di esempi rumorosi o che non sono informativi
- ***Mancanza di flessibilità:*** il numero di punti sintetici generati deve essere fissato a priori, non permettendo flessibilità in caso sia necessario un ri-bilanciamento

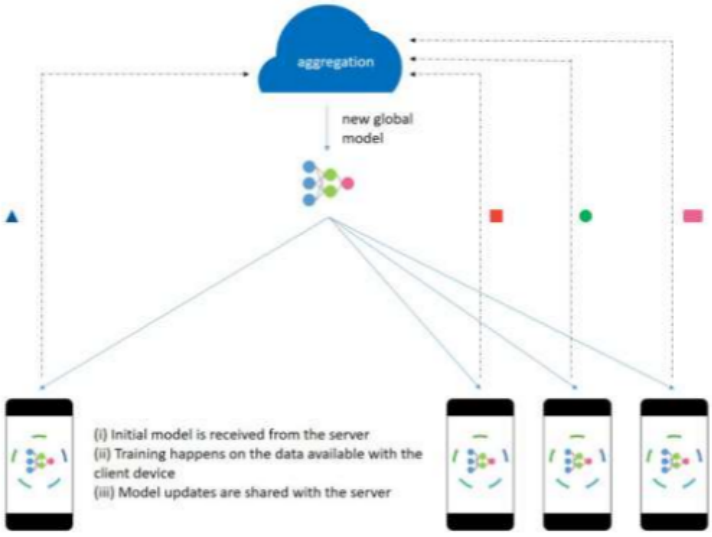
1.4.3 Cost-Sensitive approach

Viene assegnato un peso a ciascun tipo di errore, in relazione a quale classe appartiene (maggioritaria o minoritaria); in questo posso ad esempio dare più peso alla classe che ha meno esempi ribilanciando il mio dataset a livello algoritmico (invece che a livello di dati).

Capitolo 2

Federated Learning

L'idea è quella di consentire il training collaborativo (nel senso di distribuito) di grandi modelli. Viene addestrato ciascun nodo della rete e poi ciò che viene mandato periodicamente ad un punto comune sono i parametri.
È un paradigma di estremo successo in tutti i quei casi dove è rilevante la privacy dell'utente, dato che non è necessario mandare i dati primari ma possono essere condivisi solamente i parametri.



Bisogna fare attenzione a dopo quanto tempo di training locale cominciare a condividere i parametri con il modello centrale.

Possibile attacco

Un attaccante potrebbe fare un attacco ai dati training, nel caso in risucisse ad aggiungersi ad un gruppo di nodi che condividono i loro parametri ad un modello federato.
Si può attaccare mandando dati casuali (**vandalismo**) o ancora peggio, si può attaccare mandando valori di pesi che cercano di far sì che poi il modello si comporti *come voglio io* nel caso che mi interessa (**trojan horse**); è più facile fare un attacco rispetto al training classico centralizzato dove c'è un controllo dell'accesso.

Considerazioni

- da un lato ho una garanzia di privacy, dato che i dati primari non vengono condivisi
- dall'altro è vulnerabile ad attacchi nella fase di training del modello

Inoltre, ci possono essere problemi anche quando i partecipanti sono tutti onesti:

- un partecipante smette di contribuire
- un partecipante finisce i dati di addestramento
- si fa fatica a fare un bilanciamento del modello centrale, dato che ogni nodo locale fa il suo bilanciamento locale
- non so niente dei parametri che ricevo (magari è un predittore statico...)

2.1 Sicurezza nel Federated Learning

Quando si vuole fare un modello di minaccia la prima cosa da fare è un modello degli asset: per ogni asset si devono individuare i **failure mode**, ovvero modalità note con cui un asset può malfunionare.
Per sapere i failure mode di un modello di intelligenza artificiale, bisogna prendere delle proprietà generali desiderate (anche indipendenti dallo scenario, come ad esempio l'accuratezza); un failure mode è la negazione di una di queste proprietà.
Un esempio di due proprietà la cui negazione rappresenta un failure mode sono:

- **Privacy:** avviene ad esempio quando qualcuno può ricostruire i valori del training set in sede di inferenza; nel caso del federated learning, si potrebbero avere problemi anche in sede di training, se un nodo perifico può dedurre i valori di training degli altri nodi
- **Robustezza:** intesa come il numero di errori introdotti per ogni variazione del training set

Viene definito **threat** l'azione che porta ad un failure mode.
Un esempio di threat ad un modello di federated learning sono:

- ***Model Poisoning***, iniettando dei pesi fasulli (vandalismo o trojan horse)
  - potrei difendermi considerando come *outlier* i valori che sono tre volte la deviazione standard (metodo del *three-sigma*), e facendo un filtraggio statistico (supponendo che siano una minoranza); ovviamente funziona solo su attacchi che mirano a modificare i valori del training set per diminuire l'accuratezza complessiva del modello (vandalismo) e non attacchi avversariali, ovvero mirati a modificare un un punto specifico del training set
- ***Privacy Attack***, cercando di trarre informazioni sugli altri modelli della rete; si dividono tra:
  - ***Reconstruction Attacks:*** mirano a ricostruire il training set originale e i dati degli utenti
  - ***Membership Attacks:*** mirano a capire se un determinato dato specifico è presente nel training set
  - $\Rightarrow$ potrei far viaggiare i pesi criptati, anche se comporta tutti i costi di avere un sistema di cifratura (criptare, decriptare, gestione delle chiavi...); un'alternativa è quella di usare sistemi crittografici omomorifici, anche se hanno dei *footprint* molto grossi rendendoli difficile da applicare in casi reali

Una volta individuati i failure mode e i rispettivi threat, si può passare a fare un'analisi del rischio per cercare di limitare i possibili danni. Nel caso del federated learning abbiamo un rischio dovuto al fatto di avere un sistema distribuito invece che centralizzato.