UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale in Ingegneria Informatica e dell'Automazione

Rilevazione di malware da traffico DNS passivo con una rete neurale LSTM

Malware detection from passive DNS traffic using a LSTM neural network

Relatore: Chia.mo

Prof. Luca Spalazzi

Correlatori: Chia.mi

Prof. Alessandro Cucchiarelli

Prof. Christian Morbidoni

Tesi Finale di:

David Caprari

Anno Accademico 2019/2020

Abstract

Il seguente lavoro di tesi si focalizza su l'utilizzo di una rete neurale Long Short-Term Memory per l'intercettazione di traffico DNS malevolo da parte di malware che utilizzano Domain Generation Algorithms per la generazione del dominio dei loro server di comando e controllo.

Lo studio si basa su traffico reale ottenuto da richieste indirizzate al server DNS della rete GARR delle università italiane in un orizzonte temporale definito. Tali dati vengono poi confrontati con diverse risorse online per l'effettiva definizione delle query malevole e la suddivisione di queste in base all'appartenenza ad alcune famiglie di DGA. Ottenuto un dataset di controllo si utilizzano più reti LSTM, addestrate su differenti set di dati pre-lavorati, per testarne le performance di riconoscimento. Viene infine effettuato un confronto delle diverse performance evidenziando punti forti e deboli di ogni soluzione.

Sommario

1. Introduzione ai Malware basati su DGA e studi correlati

1.1 DNS e reti

Definizione e scopo

Presentiamo anzi tutto alcuni dei concetti su cui si basa questo lavoro. In particolare, si forniscono alcuni dei concetti chiave legati alle reti informatiche, alla loro struttura ed al loro utilizzo da parte di un utente.

Oggigiorno è possibile suddividere una rete informatica in diversi strati. In particolare, il modello TCP/IP Reference[1] suddivide le reti in 4 layer di protocolli (1):

1. Link
2. Internet
3. Transport
4. Application

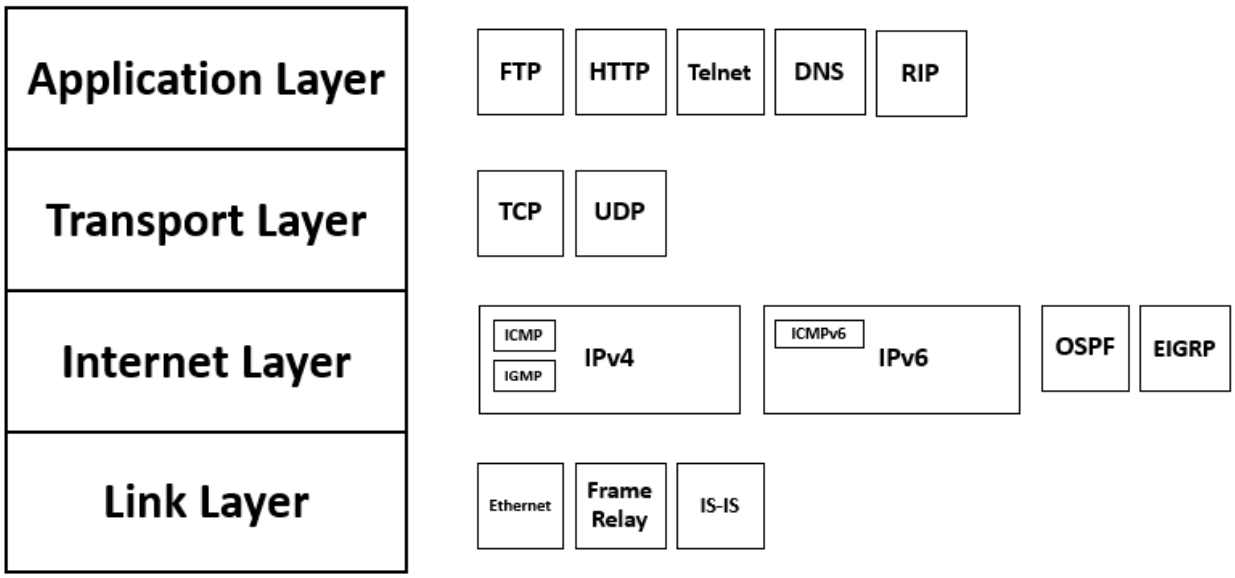

Fig. 1: Modello TCP/IP. Associa ad ogni layer i protocolli che possono comporre una rete.

[1] Modello più compatto dell'OSI Reference Model (26), sottintende alcuni layer del modello originale e utilizza il protocollo IP come strato Network. L'OSI è lo standard di principi per la struttura di una rete più seguito e completo.

L'informazione di uno strato viene suddivisa in parti, ogni parte di queste viene incapsulata nel segmento del protocollo scelto tra quelli del livello inferiore. Il ricevitore decapsulerà i pacchetti per ottenere le parti dell'informazione e ricostruirla.

Un breve esempio: supponiamo di avere una pagina web http da inviare ad un client. La pagina verrà suddivisa in n parti. Ognuna di queste n parti verrà inserita in un pacchetto TCP: il pacchetto TCP che contiene l'n-esima parte della pagina HTTP sarà una stringa di dati che conterrà all'inizio le informazioni relative al protocollo, poi l'informazione della pagina di una lunghezza prefissata ed infine un segmento che indichi il termine della stringa. Questo pacchetto a sua volta sarà incapsulato nel livello internet e così via.

Dallo strato più profondo a quello più astratto, tralasceremo completamente lo strato Link e ci dedicheremo agli altri, indicando, di questo, solamente che un dispositivo su di una rete si serve di diverse tecnologie per l'invio dei dati alla rete stessa, quali Ethernet, Wi-Fi, ADSL, etc.

Nelle reti che ci interessano il protocollo Internet che viene utilizzato principalmente è il protocollo IP, nelle sue versioni IPv4 e IPv6, rispettivamente Protocollo IP versione 4 e Protocollo IP versione 6. Questo protocollo, in breve, consente di assegnare un indirizzo ad ogni dispositivo connesso alla nostra rete. Un po' come se il nostro indirizzo IP fosse l'indirizzo della nostra abitazione. Ogni componente della rete che è interessato a comunicare con noi sa che dovrà inviare le informazioni a questo indirizzo.

Un indirizzo IPv4 generico viene indicato comunemente da 4 numeri intervallati da tre punti. A livello macchina è codificato da 32 bit, quindi 4 byte. Ogni byte è separato da un punto, nella notazione *dotted,* e varia da 0 a 256, codificando quindi in totale un numero limitato di indirizzi. Un indirizzo IPv6, in modo similare, consente un numero maggiore di indirizzi totali, essendo composto da una stringa di 16 byte.

A questo punto, anche se è meno rilevante per questo lavoro e seguendo la metafora precedente, il layer di trasporto si occupa di far in modo che le nostre informazioni arrivino a casa nostra, in modo simile ai vettori delle poste e le aziende di spedizioni.

I protocolli utilizzati per questo fine sono i protocolli TCP e UDP. Entrambi si occupano della trasmissione delle informazioni tra client e client o tra client e server, la loro differenza riguarda il formato leggermente

diverso. Banalmente il TCP è molto simile alla posta raccomandata con ricevuta di ritorno, l'UDP più simile alla posta ordinaria. Più precisamente e in breve, il TCP supporta l'ordinamento dei pacchetti e consente una trasmissione intera senza perdite rimettendoci in velocità di trasmissione, l'UDP invia i pacchetti senza ordinamento, anche per nodi diversi, per ottenere più velocità e poi semmai recuperare i pacchetti persi (2) (3). Ovviamente tutti i nodi delle nostre reti supportano entrambi i protocolli che non sono esclusivi.

Lo strato Application è quello formato dai protocolli che consentono la gestione dei pacchetti inviati o ricevuti da parte di applicazioni in grado di decodificare le informazioni che contengono. Ad esempio, HTTP e HTTPS sono i due protocolli di livello applicazione che utilizziamo quando navighiamo online su un browser; quando mandiamo un file in stampa alla stampante, viene utilizzato generalmente il protocollo SMTP o meno frequentemente il protocollo SMB[2]. In questi esempi, HTTP e SMTP forniscono le informazioni disimballate dagli strati precedenti alle applicazioni browser o al decoder della stampante che sono in grado di decodificare l'informazione che voleva essere inizialmente inviata.

Il protocollo che ci interessa maggiormente di questo layer è il protocollo DNS (4). Tornando alla nostra metafora, il DNS non è altro che l'ufficio postale che raccoglie le lettere che inviamo e ci fornisce l'indirizzo alle quali vogliamo inviarle.

Molto più nel dettaglio, un DNS è un server online che contiene un grande database nei quali associa ai domini web i loro indirizzi IP. Ogni volta che un dispositivo desidera connettersi ad un server utilizzando il nome di dominio del server (ad es. google.com), il DNS risponde al dispositivo comunicandogli l'indirizzo IP, IPv4 o IPv6, che deve contattare (per google.com ad esempio 8.8.8.8).

Ovviamente all'interno del codice del dispositivo preso in esame, l'indirizzo del DNS è codificato in indirizzo IP e non in nome di dominio, altrimenti non sarebbe possibile risolvere il nome del DNS e quindi risolvere in cascata tutti i nomi di dominio.

[2] SMTP e SMB sono degli esempi di protocolli applicazione che si utilizzano in genere per la trasmissione di informazioni all'interno di una rete LAN.

Il server DNS quindi funge da snodo principale ed indirizzatore di quasi tutte le nostre comunicazioni ed ogni rete o sottorete ne possiede uno (se non lo possiede letteralmente, si punta ad uno esterno). Solitamente è uno strumento gratuito che i gestori delle infrastrutture web mettono a disposizione per rendere possibile la navigazione sulla rete.

Inoltre, il processo è completamente trasparente all'utente, ma molti dispositivi supportano la modifica del server DNS preimpostato. Alcune motivazioni che potrebbero spingere un utente alla modifica di questo parametro sono la necessità di una risposta DNS più veloce oppure il bypass di alcune restrizioni sui contenuti online che alcune compagnie telefoniche dispongono sui propri DNS.

Oscurare un sito in un DNS impedisce l'accesso al sito da chiunque utilizzi quel DNS come proprio risolutore di indirizzi. L'attività di oscuramento del sito viene detta Blacklisting (da *"inserire il dominio in una lista nera"*) e viene utilizzata principalmente per restringere l'accesso a siti che violano le normative statali sui contenuti da apporre nel proprio sito. L'esempio più diffuso è il blocco della pirateria informatica di contenuti audiovisivi: il crimine a volte non è così grave da dover necessariamente chiudere il dominio che, più velocemente e con meno burocrazia, viene oscurato.

A volte è possibile quindi modificare il proprio server DNS per accedere a contenuti che altrimenti sarebbero censurati in quanto essi non sono effettivamente rimossi: viene semplicemente tagliato un metodo di accesso al contenuto.

L'analisi del traffico che viene registrato da un server DNS e la possibilità di utilizzare un sistema di Blacklisting sono due delle pratiche più importanti legate a questo argomento che si renderanno fondamentali per questo lavoro, come vedremo in seguito.

1.2 Malware

Definizione, caratteristiche e scopo

Malware, termine coniato nei primi anni 90 da Yisrael Radai (5), è l'abbreviazione di Malicious Software, che generalmente viene tradotto in italiano come Virus Informatico o Codice Maligno.
Nel campo della sicurezza informatica viene definito come malware qualsiasi programma o parte di codice che interferisce con le operazioni svolte da un utente per mezzo di un computer (6).

La definizione appena data è abbastanza lasca ed il termine contiene in realtà diverse tipologie di codici maligni con diverse caratteristiche e diverse potenzialità malevole. È possibile classificare i malware seguendo alcune loro proprietà, di solito il loro comportamento o il metodo di diffusione. Di questi, una gran parte dei malware più elaborati e pericolosi, è nascosta intelligentemente in codice difficile da rilevare o utilizza vulnerabilità di cui al momento della proliferazione non si è a conoscenza.

La loro grande diffusione ed una sufficiente molteplicità rende molto difficile individuare caratteristiche comuni a tutti i tipi. Ciò rende arduo il compito di trovare una unica "cura miracolosa" da parte di chi si occupa di sicurezza informatica su larga scala.

La propagazione del malware avviene principalmente attraverso frammenti di codice che vengono inseriti in codici eseguibili già esistenti e la quasi totalità di essi, con rarissime eccezioni, è utilizzata a scopo di lucro, direttamente o indirettamente. Si ha, ad esempio, un guadagno diretto quando il codice richiede un pagamento da parte di un utente, che sia un inserzionista pubblicitario od una vittima di furto di informazioni, o si ha un guadagno indiretto quando, sempre ad esempio, si effettua un sabotaggio ai sistemi informatici di una azienda rivale.
Come facilmente immaginabile, in base alle loro caratteristiche e al loro comportamento, generano valuta in modi tanto diversi quanto ne sono varie le diverse tipologie.

La caratteristica di alcuni malware sulla quale grandissima parte di questo lavoro si basa è il fatto che per il controllo di alcune funzioni del codice malevolo, per lo stesso aggiornamento del codice o proprio per pura costruzione, hanno bisogno di comunicare via rete con un server esterno. Tale server esterno viene chiamato "Command and Control Server" (server di comando e controllo, abbreviato in C&C Server).

Riprenderemo questo concetto più avanti, avendo la necessità di esplicarlo completamente.

Proliferazione

Ad oggi, il canale di diffusione più utilizzato per veicolare le infezioni da malware è Internet, attraverso il download volontario o involontario di codice dannoso nel world-wide-web o attraverso l'utilizzo improprio di allegati e-mail (7). La macchina dell'utente, oltre a poter ricevere tale codice da reti esterne, può essere vulnerabile ad attacchi che avvengono su reti LAN se altre macchine che ne fanno parte sono infette.

È questo il caso del ransomware Cryptolocker, che tra il tardo 2013 e tutto il 2014 fu così diffuso da far notizia in gran parte del mondo occidentale: infettava la prima macchina di una rete all'apertura di un allegato ricevuto via e-mail e poi si diffondeva attraverso la rete LAN nelle altre; alcune delle macchine infette della rete ritrasmettevano l'allegato infetto dando di nuovo il via al ciclo.

Un altro canale di diffusione comunque rilevante ancora oggi è quello dei supporti rimovibili. Alcuni malware infatti sono in grado di depositarsi nei supporti rimovibili che vengono occasionalmente collegati ad una macchina infetta, per poi replicarsi in nuove macchine al momento dell'occasionale collegamento del supporto. Tale metodologia era molto diffusa a metà degli anni '90 e per gran parte degli anni 2000, prima di una successiva crescita esponenziale dell'utilizzo del web (5).

Catalogazione

È possibile classificare i malware in diversi modi. Ad esempio, alcuni li classificano basandosi sul loro target, sul loro comportamento o sulla loro direttiva di attacco. Riportiamo un paio di classificazioni che meritano di essere citate.

Una generica classificazione che si basa sul target dell'infezione è la seguente (5):

- File infectors: infettori di file, possono essere infettori diretti, quando infettano altri file immediatamente dopo la loro esecuzione o infettori memory-resident, quando sono in grado di attendere in memoria l'avvio di un programma o l'esecuzione di un file che possono infettare.
- Boot-sector infectors: infettano il settore di boot di un disco per ottenere il controllo del flusso di esecuzione del sistema ancor prima del sistema operativo. Generalmente corrompono la prima istruzione del settore di boot affinchè questa punti al codice del malware e lo esegua per prima.
- Virus Multipartiti: quando sono in grado sia di infettare file che di corrompere il boot di un sistema.

A seguire, una classificazione notevolmente efficiente per suddividere i malware in base al loro comportamento (6). Oltre ad essere più larga e a classificare le tipologie in modo più specifico, li classifica anche seguendo termini più comuni all'utente medio.

Segue:

- Virus: in modo simile al virus biologico da cui prendono il nome, si replicano copiandosi all'interno di altri file, programmi o settori di supporti di memorizzazione in modo da essere eseguiti ogni volta che si ha accesso a quella risorsa software o hardware (8). Chiamati anche Infectors (seguendo una notazione più desueta), possono corrompere anche il settore di boot di un sistema. Tra questi abbiamo banalmente virus eseguibili, quando compongono un intero programma malevolo, script, quando sono solo una parte o funzione del programma, macro-virus, quando eseguono un set di istruzioni automaticamente.
- Trojan Horse: dal Cavallo di Troia citato nell'Odissea, è del codice malevolo inserito in un programma che ha funzionalità lecite. Generalmente non ha capacità di replicazione interne ma segue la diffusione

del programma principale che lo ospita. Caratteristica comune al Cavallo di cui sopra, tenta di essere eseguito all'insaputa dell'utente. Esistono Trojan Horse che focalizzano i loro attacchi sull'ottenere il controllo della macchina da parte di un utente malevolo che ne acquisisce il controllo remoto.

- Spyware: codice malevolo utilizzato per carpire informazioni rilevanti da un utente. Ha tra gli obbiettivi principali i metodi di pagamento dell'utente generico o i segreti industriali nel caso degli utenti aziendali.
- Backdoorware (o semplicemente Backdoor): programmi che sfruttano le backdoor di un sistema. Si intendono per Backdoor (porta sul retro) tutti i metodi di accesso non autorizzati ad un sistema. Possono essere Backdoor conosciute, come ad esempio il meccanismo di recupero password di un account, o sconosciute, nel caso di alcune vulnerabilità software. Possono essere classificati come backdoorware anche software che utilizzino vulnerabilità dell'hardware come ad esempio la vulnerabilità Meltdown dei processori Intel x86 o la vulnerabilità Spectre dei processori Intel, AMD e ARM.
- Adware: malware meno nocivi dei precedenti che fanno in modo di presentare all'utente un maggior numero di contenuti pubblicitari rispetto al normale, magari attraverso lo spam di ad o il controllo di alcune vulnerabilità dei browser. Causano comunque danni ai sistemi, quali il rallentamento o il rischio per la privacy dell'utente, tendendo a non avere regolazione sull'advertisement pubblicitario presentato.
- Ransomware: o anche cryptoware, sono malware che una volta infettato un dispositivo hanno come obbiettivo quello di criptarne i dati contenuti nei supporti di memorizzazione con una chiave a cifratura complessa (tra le più comuni SHA128 o SHA256). Viene poi chiesto un riscatto all'utente per la decrittazione dei dati. Di solito sfruttano meccanismi di propagazione attraverso rete LAN, avendo come target principale i sistemi informatici diffusi di grandi aziende ed organizzazioni.

  Stando ad alcune statistiche (7), nell'anno 2019 gli attacchi ransomware sono aumentati del 119%.
- Hijacker: virus che prendono possesso delle funzioni principali di un applicativo e ne distorcono l'utilizzo, ad esempio prendendo possesso dei motori di navigazione dei browser web per aprire pagine o indirizzi indesiderati.

- Rootkit: più che veri e propri malware sono strumenti costruiti su repliche di driver o applicativi di sistema già presenti nel sistema operativo, con la funzione aggiuntiva di nascondere la presenza illecita o non voluta di file o impostazioni. Vengono generalmente affiancati a malware del tipo spyware o trojan per nasconderne le tracce e si tende ad individuare l'unione dei due come malware unico.
- Worm: questi malware non hanno l'obiettivo di infettare altri file per la loro diffusione, ma utilizzano il sistema operativo e lo modificano per tentare di replicarsi attraverso diversi tipi di reti. Per essere eseguiti utilizzano diversi stratagemmi per indurre l'utente a consentirne l'esecuzione.

  Tendono a rallentare il sistema con operazioni indesiderate o dannose e possono essere associati ad altri malware per veicolarli.
- Keylogger: molto simili agli spyware, sono strumenti software o hardware che registrano l'input fornito dall'utente alla tastiera di un computer per poi estrapolarne informazioni personali con l'intento di venderle o utilizzarle. Vengono associati a dei rootkit di oscuramento per non essere individuati da utenti o sistemi antivirus. Molto simili ai keylogger ma meno diffusi esistono anche malware Desktop Recorder che registrano in formato video o immagine ciò che la macchina proietta a schermo e malware Memory Scraper che attingono a dati memorizzati in memoria volatile. Gli ultimi due rimangono molto meno diffusi dei primi in quanto i nuovi sistemi operativi implementano misure più restrittive nella gestione dei permessi di accesso in memoria e alle periferiche principali.

Vulnerabilità

Essendo la definizione di malware piuttosto generica, lo è anche quella di vulnerabilità che essi sfruttano.
Si parla di sistemi sotto attacco malware in diversi contesti e così come non è semplice catalogare i diversi malware, non è semplice catalogare le vulnerabilità. Si possono però fornire delle linee guida dettate dal buon senso per prevenire la loro diffusione e per proteggere i dati sensibili degli utenti (5).

La prima e più consigliata, rivolta soprattutto all'utente semplice, è quella di aggiornare costantemente il software utilizzato. Molti malware infatti sfruttano debolezze nei software generici o inaccuratezze nei software di protezione. Effettuando costanti aggiornamenti ci si assicura di sfruttare la correzione di eventuali bug per diminuire la vulnerabilità generale di un sistema.

Un'altra, banale ad ogni programmatore, è quella di cercare di progettare software adeguato con le opportune protezioni logiche e strutturali. In questo modo, si cerca di rendere il codice più robusto e resistente a possibili utilizzi illeciti partendo direttamente dalla sorgente.

Si consiglia alle aziende e ai gestori di grandi reti distribuite di utilizzare software o sistemi operativi diversi, facendo in modo che una eventuale vulnerabilità di un determinato software o OS non sia presente in un'altra macchina, così da preservare parte delle informazioni della rete.

L'ultima, a volte completamente trasparente all'utente soprattutto in architetture Microsoft, è quella della gestione opportuna dei privilegi di root. Si consiglia infatti di seguire il principio del privilegio minimo, fornendo ad ogni singolo utente o file l'opportunità di accedere al numero minimo di risorse che richiede. Così facendo, oltre ad avere un'esecuzione più efficiente e ordinata, è difficile per un eventuale programma infetto accedere ad una risorsa per la quale non ha autorizzazione.

Greyware

Come in molti altri ambiti nella vita, non sempre esistono gli assoluti. È sempre possibile trovare qualcosa nel mezzo che possa essere sia positiva o negativa in base all'uso che se ne fa o alle circostanze che la circondano. Lo stesso vale per il software e come possono esistere programmi "buoni o cattivi" esistono anche programmi che si trovano nella sottile linea che li separa. Vengono chiamati Greyware (o Riskware) (5).

Greyware, più nello specifico sono programmi che possiedono feature che hanno il potenziale per essere pericolose. Assumiamo il breve esempio del software di gestione dell'attuatore di un passaggio a livello ferroviario: utilizzato in modo positivo consente la gestione del traffico automobilistico e ferroviario, utilizzato senza criteri potrebbe causare danni a cose o persone.

Nella classificazione già data dei malware, si possono considerare greyware tutti gli spyware e gli adware non utilizzati per nuocere alla privacy degli utenti. Possono essere considerati greyware anche gli strumenti di amministrazione software finiti in mani sbagliate o quei malware detti Joke (o Joker) non precedentemente classificati in quanto il loro scopo unico è quello di prendersi gioco dell'utente.

L'esistenza stessa dei greyware pone il problema etico del controllo delle operazioni di difesa degli antivirus o, più in generale, di tutte quelle azioni preventive che possano essere eseguite a priori nei confronti di codice che non sempre viene utilizzato per nuocere o danneggiare un utente. Il problema appena citato ha risvolti anche nelle possibili considerazioni posteriori a questo lavoro.

1.3 Botnet

Definizione, caratteristiche e scopo

Una botnet, o rete di bot, o robot network (5), è una rete composta da un numero di macchine o bot. Il termine bot tende a indicare genericamente un programma automatizzato che esegua interventi senza l'intervento di un utente. Il termine in senso stretto non indica nulla di malvagio.

Con gli anni (90 in poi) il termine ha assunto una connotazione peggiore, finendo ad indicare macchine infette da malware con caratteristiche specifiche. Questi malware, che tendono ad usufruire di rootkit specifici per mantenersi silenti e nascosti, prendono possesso solo di alcune funzionalità del sistema.

Consentono inoltre il controllo di queste funzioni da un utente terzo, un hacker a questo punto definito botmaster.

Assumendo il controllo di centinaia o migliaia di computer, le botnet vengono generalmente utilizzate per inviare spam o virus, rubare dati personali o lanciare attacchi DDoS.

Per attacco DDoS (Distributed Denial of Service) si intende un attacco hacker che, utilizzando botnet numerose, cerca di impedire la fruizione di un servizio banalmente intasandolo.

Le botnet sono molto legate al concetto di malware essendo uno degli esempi più insidiosi della loro pericolosità su larga scala. Queste reti vengono considerate la minaccia più grave e impellente che affligge Internet oggigiorno (5) (9).

Per ottenerne una generica classificazione vengono associate al loro malware, anche se condividono tra le varie classi molte caratteristiche, quali la presenza di un rootkit, di malware che hanno come target servizi e periferiche di rete, di meccanismi di collegamento online.

Seguono la proliferazione dei malware che le generano, con un aumento costante del loro numero negli ultimi anni. Gran parte degli attacchi DDoS giunti a favor di stampa negli ultimi anni sono causati da attacchi effettuati da botnet estremamente numerose.

Meccanismi interni

La caratteristica delle botnet che le rende molto rilevanti per questo lavoro è il fatto che necessitino di connessione con un C&C Server. È ovvio che per essere gestibili da un hacker (o una organizzazione di) hanno bisogno di poter essere gestite in remoto. Questo viene reso possibile da un agente software malevolo che abilita le componenti host della macchina infetta, il quale può avere forma di un file libreria a collegamento dinamico (DLL) o parte di parte di codice legato al codice del malware. Come risultato dell'avvenuta connessione l'agente del bot è in grado di ricevere ed interpretare comandi ricevuti dal botmaster, eseguire attacchi o inviare dati sensibili al botmaster.

La componente network della botnet è costituita da qualsiasi risorsa che la rete può utilizzare per realizzare le sue direttive. Alcune componenti che la nuova rete instaurata può possedere ed utilizzare sono le seguenti (5):

- Canale di comando e controllo: Come già brevemente introdotto nel capitolo precedente, un server di comando e controllo è una risorsa online che cambia o influenza il comportamento delle botnet. Il termine deriva da utilizzo militare: il dipartimento della difesa americano identifica come identità di comando, che fa esercizio di autorità e direzione, un propriamente designato comandante e le forze collegate che concorrono al completamento di una missione. In questo modo si ammette anche la possibile pluralità dei server di comando e controllo.
  Il C&C è il componente critico della botnet, in quanto risiede dell'abilità del botmaster di controllare la rete. Rimuovendo il server di comando e controllo si rimuove il suo potere di gestione su tutta la rete, che diviene quindi innocua.
- Server di distribuzione del malware: la risorsa online che esegue l'hosting delle componenti malware della botnet, quali l'agente bot, gli aggiornamenti del codice o altri file rilevanti. Ogni volta che una componente del malware o qualsiasi altro file legato al malware ha bisogno di essere aggiornato, la botnet utilizza il suo server di distribuzione per effettuarne l'aggiornamento sulle macchine infette. Viene utilizzato anche come sorgente principale di download delle componenti malware e degli installatori che agiscono sulla macchina da infettare nella fase di infezione.

- Drop zone: la risorsa in cui il botmaster deposita i dati ottenuti dalle macchine infette. Viene utilizzata soprattutto come raccoglitore attivo di informazioni.

Nelle botnet con architetture più semplici, le risorse appena descritte possono trovarsi sulla stessa macchina o all'interno della stessa rete identificata da un unico indirizzo IP.

Inoltre, i server C&C possono essere costruiti e messi in funzione con diverse strutture caratteristiche. La struttura più diffusa è quella centralizzata, nella quale il C&C occupa un nodo unico e centrale. Questa consente l'esecuzione di comandi in tempo reale, in stile definito Push Style, quando le macchine infette risultano costantemente connesse in attesa di un comando da eseguire. In uno stile differente, con comandi in Pull Style, i bot stabiliscono comunicazione con il nodo C&C periodicamente e attendono la presenza di un nuovo comando. Il botmaster quindi pubblica l'istruzione su un file o una direttiva ai quali i bot hanno accesso. Ma l'esistenza di un tempo di propagazione dei comandi rende ovviamente la loro esecuzione non in tempo reale.

Per quanto possegga alcuni vantaggi la struttura centralizzata è il vero punto debole delle botnet. Per questo esistono botnet con C&C decentralizzati che presentano ridondanza di nodi di controllo. In questa struttura i nodi infettati si comportano sia da server di comando e controllo che da client. Questo elimina l'esistenza di un unico nodo centrale della botnet e stabilisce una struttura Peer To Peer, che ha i suoi vantaggi e i suoi vantaggi. Ad una maggiore resilienza della botnet corrisponde una maggiore complessità dei meccanismi dei malware sugli host, il che rende l'infezione più rilevabile localmente.

Infine, esistono strutture ibride che uniscono i pregi delle due precedenti. Un esempio di queste botnet è l'architettura fornita dalla combinazione ZeusP2P/Murofet che utilizza un sistema P2P come C&C primario e, nel caso in cui la connessione ai suoi peers fallisse, ripiega su un server C&C di backup a struttura centralizzata.

Meccanismi di difesa

L'infrastruttura di comando e controllo di una botnet e i suoi scopi vanno preservati se chi gestisce la rete vuole continuare a nuocere e a mantenerla funzionante. Diventa quindi imperativo per il botmaster instaurare dei meccanismi di difesa a protezione del canale di comunicazione delle direttive di controllo. Alcuni meccanismi sono i seguenti (5):

- Bulletproof Hosting: esistono dei servizi di hosting di siti web e domini che consentono di caricare sulle proprie piattaforme del codice malevolo, dati piratati o pornografia. La compagnia che gestisce l'hosting diventa quindi lo scudo dell'hacker dalla legislazione dello stato in cui lui carica il suo materiale online. Sempre più stati istituiscono normative stringenti per un utilizzo pulito di Internet e per questo il Bulletproof Hosting sta diventando sempre più raro.
- Dynamic DNS (DDNS): è un servizio molto utilizzato per associare un solo nome di dominio allo stesso host, nonostante questo cambi spesso il suo indirizzo IP. Molto comodo per le istituzioni o azienda che intendano avere server pubblici sempre raggiungibili, non è molto attraente per i cybercriminali. I costi del servizio, le policy molto stringenti e l'introduzione nelle macchine host e client di un software proprietario sono troppi ostacoli affinché una botnet funzioni indisturbata.
- Fast Fluxing: Attraverso un sistema di redirect, viene associato ad un singolo nome di dominio molti indirizzi IP utilizzando DNS round-robin che registrano record con un TTL (time-to-live) molto piccolo. In questo modo il DNS può rispondere ad una singola richiesta con molti indirizzi IP. Anche questa soluzione presenta le sue problematiche in quanto basta che il gestore del DNS agisca sul sistema per ripristinare i TTL e bloccare il canale di comunicazione.

A questi, appena descritti, va aggiunto il meccanismo di protezione più importante per questo lavoro: il Domain Fluxing.

Domain Fluxing

Di solito l'indirizzo web del C&C viene codificato all'interno del malware o su una risorsa che il malware può utilizzare in locale: abbiamo appena visto con quali meccanismi. Come abbiamo già detto, ci si è resi conto che indipendentemente da quanto è ben nascosto il server di comando e controllo e da quanto è complicata la sua architettura, è sufficiente oscurarlo per rendere la botnet innocua e liberare le macchine infette dal controllo dell'hacker (10). La macchina infetta, dopo che sia stato oscurato il dominio del server che la comanda, è impossibilitata oltre che a comunicare, anche ad effettuare aggiornamenti del software o delle sue risorse. Quindi, persa la capacità di contattare il server C&C, perde la capacità anche di comunicare con i server di distribuzione per effettuare aggiornamenti del codice e quindi riottenere un nuovo canale di comunicazione.

Per ovviare a questo problema, si è data la capacità all'infezione di recuperare le comunicazioni attraverso una sua lista di domini legati al suo botmaster. Quindi, l'abilità dei malware di generare periodicamente una loro lista di domini da contattare è detta Domain Fluxing.

Il codice che rende disponibile questa capacità è un algoritmo detto Algoritmo di Generazione di Domini (Domain Generation Algorithm). Nel testo ci riferiremo ai domini generati da tali algoritmi come Domini DGA.

Il malware quindi può contattare i domini forniti dal suo algoritmo DGA sapendo che il botmaster avrà cura di utilizzare uno di questi nello stesso periodo di tempo. Un vantaggio di questo sistema è il fatto che in questo modo il botmaster evita il blacklisting del dominio, semplicemente perché anche se un dominio venisse inserito nelle liste nere di indirizzi web indesiderati, basterà aspettare che gli algoritmi di DGA delle macchine infette si sincronizzino su di uno generato più avanti nel tempo. Ovviamente, l'hacker ha il tempo di registrare in anticipo il dominio che sa verrà utilizzato dai client per poi puntarlo al nuovo C&C. Infine, i domini vengono sempre utilizzati per un piccolo periodo di tempo, in modo da rendere sia più difficile l'individuazione, sia inutili i sistemi che assegnano una reputazione ai domini (5).

Gli algoritmi di DGA quindi sono in grado di generare una lista di domini a partire da diverse condizioni. Chi non è neofita dell'informatica saprà bene che è impossibile ottenere attraverso uno strumento informatico standard un codice od una informazione che sia completamente casuale. Un generatore aleatorio può infatti partire da un seme (o seed, in inglese) per generare dei valori random. Seme che ad esempio può essere uno stato attuale della macchina, il clock di sistema, una previsione meteo online, la luminosità della fotocamera, un numero razionale, etc. Esisterà quindi una funzione matematica che legherà il seme al numero generato: ogni funzione di questo tipo ha un certo grado di reversibilità.

Se il seme varia con variabili che variano nel tempo l'intero algoritmo DGA e i domini generati assumono la caratteristica Time-Dependent (dipendenti dal tempo). Allo stesso modo, se il seed generato non ha a che vedere con lo scorrere del tempo, l'algoritmo e i domini vengono detti Time-Independent (non dipendenti dal tempo). Questa caratteristica, anche se può sembrare piuttosto banale, aumenta la diversità degli algoritmi, aumentando i casi che chi si occupa di sicurezza informatica deve prevedere.

Uno dei più grandi svantaggi di questo sistema è che è possibile effettuare reverse engineering sul malware per ottenere il seme casuale. Ottenuto il seme, è possibile utilizzare il DGA per prevedere i domini che saranno utilizzati dalla botnet ed effettuare azioni preventive (quali ad esempio il Sinkholing[3] del dominio). Altri svantaggi risiedono nel fatto che vengono generate molte richieste che puntano a domini non esistenti (NXDomain[4]) oppure, con piccola probabilità, possono essere generate richieste a domini già esistenti, registrati da entità legittime. Il rumore generato dal gran numero di richieste a domini non esistenti rende più facile l'individuazione del malware da software antivirus o da chi gestisce la rete (5) (10).

[3] Per Sinkhole si intende lo strumento, di solito un server, che dà la possibilità ai ricercatori di sostituirsi al botmaster e prendere il controllo della botnet per analisi successive.

[4] NXDomain, che sta per Non-eXistent-domain, è il codice di risposta che viene fornito dai server DNS quando si richiede loro un dominio non registrato o non esistente.

1.4 Reti Neurali

Definizione e scopo

Una rete neurale artificiale (abbreviata in ANN o NN dall'inglese Artificial Neural Network) è una schematizzazione formale del funzionamento del cervello umano, o del cervello biologico più in generale (11) (12).

Com'è ben risaputo l'organo cerebrale è formato da un elevatissimo numero, circa dieci bilioni, di neuroni collegati tra di loro da un altrettanto elevato numero di sinapsi. Il numero di interconnessioni è stimato ad un milione di miliardi. Tali neuroni sono cellule biologiche che scambiano tra di loro informazioni attraverso segnali elettrochimici. Quando un neurone viene attivato, i neuroni che gli sono collegati si attivano a vicenda ed in un tempo relativamente breve vengono interessate dal processo intere aree del cervello.

Attraverso questo processo, il cervello riesce a discernere la realtà e ad eseguire azioni complesse, nonché processare informazioni ed apprendere.

Senza addentrarci ulteriormente nella biologia del caso, una delle differenze principali che va riportata tra un sistema cervello biologico ed un elaboratore informatico è la gestione delle informazioni. Con un certo grado di approssimazione, un computer fa eseguire alla sua unità di calcolo principale, la CPU, delle istruzioni in modo sequenziale mentre agisce su dati memorizzati e concentrati su altri supporti, solitamente sulla memoria RAM. Possiamo dire che anche i programmi software seguono questa struttura, eseguendo compiti in modo sequenziale e separando i dati dagli algoritmi che li processano. Nel cervello umano, invece, l'informazione è distribuita su unità di calcolo elementari, i neuroni appunto, semplici ma che operano in parallelo. Inoltre, l'informazione viene conservata nelle stesse unità, senza separazione. Nei neuroni però, la frequenza di trasmissione delle informazioni, e quindi di elaborazione, è sufficientemente bassa, si parla di circa 100Hz. Un calcolatore elettronico raggiunge frequenze di calcolo sequenziali del processore diversi ordini di grandezza più grandi, si parla per i computer più comuni di qualche GHz[5]. Da qui nasce l'idea di replicare il sistema biologico cercando di utilizzare la capacità di calcolo degli elaboratori elettronici.

[5] GHz che sta per GigaHertz, 10^9 Hz

Quello che i due sistemi hanno di simile è la risposta ad uno stimolo, quindi la possibilità di fornire un output quando fosse fornito un input. Tale capacità viene utilizzata come base della schematizzazione e si cerca di studiare il sistema neurone a scatola chiusa e non come processo biologico. Questo permette di astrarsi dai meccanismi biologici ed elettrochimici che lo contraddistinguono e replicarne la struttura e il funzionamento attraverso un calcolatore informatico e a un'adeguata rappresentazione formale dello stesso.

I primi studi sull'argomento risalgono al 1949 quando Donald Hebb ipotizzò l'apprendimento biologico come processo sinaptico e fornì una formula procedurale, nota come regola di Hebb:

"Se un neurone A è abbastanza vicino ad un neurone B da contribuire ripetutamente e in maniera duratura alla sua eccitazione, allora ha luogo in entrambi i neuroni un processo di crescita o di cambiamento metabolico tale per cui l'efficacia di A nell'eccitare B viene accresciuta" (13).

Basandosi su un processo di apprendimento che si basa su una schematizzazione matematica della regola di Hebb e su una necessità di minimizzazione dell'errore, nel 1958 Frank Rosenblatt propose il *Perceptron* (o Percettrone) (14), primo esempio di sistema cibernetico in grado di riconoscere immagini. Il primo percettrone possedeva uno strato di neuroni artificiali per gestire l'ingresso delle informazioni ed uno strato per gestire l'uscita e la risposta data al riconoscimento. Questi neuroni agivano come funzioni matematiche di stato per mezzo di un calcolatore elettromeccanico. I pesi delle sinapsi tra gli strati dei neuroni determinavano l'attivazione dei neuroni dello strato successivo. Tale struttura di alimentazione di strati prograda viene detta *feedforward*.

Il processo di apprendimento utilizzato da Rosenblatt, uno dei fondamentali utilizzati ancora oggi, è la retropropagazione dell'errore. In breve, in base all'errore sull'output fornito dal sistema rispetto a ciò che si attenderebbe, l'algoritmo di addestramento altera il peso delle sinapsi che collegano i neuroni in modo da fornire un output molto più vicino a quello atteso. L'attribuzione del corretto peso ad ogni sinapsi avviene al seguito di un processo di addestramento in cui si ripete più volte la retropropagazione dell'errore: si "insegna" alla macchina la capacità di riconoscere un'informazione fornendole molte informazioni di cui si conosce già

il valore atteso di output. Particolarità fondamentale delle reti neurali è che sono poi in grado di riconoscere anche elementi sui quali non sono state addestrate, fornendo un output attendibile (11).

Dopo alcune critiche che dimostravano l'incapacità del percettrone di Rosenblatt di riconoscere funzioni matematiche che non sono linearmente separabili, si abbandonò per qualche anno l'idea. Nel decennio successivo, grazie ad un aumento della capacità di calcolo degli elaboratori che permise l'inserimento di più strati di neuroni, si fu in grado di risolvere il problema del riconoscimento delle funzioni non linearmente separabili e si riprese a lavorare su questa tecnologia.

Ad oggi le reti neurali possono essere strutture software o hardware in grado di discernere e distinguere automaticamente alcuni concetti.

Vengono generalmente usati più strati di neuroni (vedi *Fig. 2*) e possono essere alimentati gli uni gli altri anche con verso inverso, le informazioni non passano quindi solamente dallo strato precedente al successivo, ma possono esistere strati alimentati a ritroso. Le reti con strutture di alimentazione a feedback vengono dette *Reti Neurali Ricorrenti* (RNN) (15).

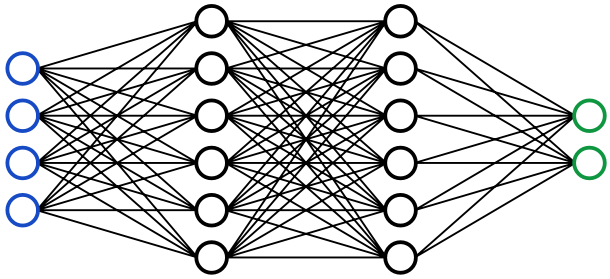

Fig. 2: Rappresentazione grafica di una rete neurale. In blu i neuroni artificiali che costituiscono il layer di ingresso. Le informazioni si muovono attraverso le sinapsi sui segmenti: ogni segmento ha un suo valore di peso della connessione. In nero i neuroni degli strati nascosti intermedi tra input e output. In verde i neuroni dello strato di output.

Una famiglia di reti neurali molto importanti per questo lavoro sono i classificatori, i quali grazie ad un input multiplo su più segnali e un opportuno addestramento supervisionato sono in grado di rispondere in output con un segnale unico che identifichi l'intero sistema di ingresso. Vengono quindi utilizzati per il riconoscimento di diverse strutture di dati, tra cui anche le stringhe, sequenze di parlato, immagini e filmati.

Reti Neurali LSTM

Le *Long Short-Term Memory* sono reti neurali artificiali ricorrenti (da non confondere con le reti neurali ricorsive) utilizzate per l'apprendimento profondo (*Deep Learning*, dall'inglese) (12) (16). Una rete LSTM possiede connessioni di tipo feedback e può processare intere sequenze di dati, ad esempio lingua parlata, video e scrittura a mano. Funzionano inoltre molto bene nel campo della rilevazione e predizione delle anomalie delle reti di traffico e nel campo dell'Intrusion Detection System.

L'unità fondamentale della rete, il neurone, è composto da una cella con memoria dati (cell), un attivatore di ingresso (input gate), un attivatore di uscita (output gate) e un attivatore di memoria (forget gate). La cella mantiene il dato che contiene per un tempo arbitrario mentre i tre attivatori regolano il flusso di informazioni dentro e fuori dal neurone (12).

La rete neurale LSTM previene il problema della scomparsa del gradiente di addestramento. In breve, questa famiglia di classificatori permette, attraverso accorgimenti progettuali, di effettuare addestramenti con set di dati più grandi ed in modo più consistente. La prima rete neurale LSTM è stata proposta da Sepp Hochreiter e Jürgen Schmidhuber nel 1997 (12).

Interpretare le performance di una rete neurale

Per valutare la robustezza di un modello predittivo si utilizzano in genere diversi parametri. Ne forniamo una veloce introduzione in modo da poter comprendere maggiormente i valori derivati dagli esperimenti che verranno eseguiti

I possibili risultati delle predizioni di una rete neurale si dividono in 4 possibili casi: un valore predetto può essere un *true positive* (o anche TP) se la rete ha predetto un valore positivo per un valore che è realmente positivo, *false positive* (o FP) se la rete ha prodotto un valore positivo per un elemento che non è realmente positivo. Allo stesso modo si ottengono anche *true negative* e *false negative*.

Il valore *precision* si ottiene da:

$$Precision = \frac{TP}{TP + FP}$$

Il valore è infatti riferito alla precisione del modello nel predire una caratteristica e individua il rapporto dei predetti positivi tra tutti i positivi.

Il valore *recall* si ottiene da:

$$Recall = \frac{TP}{TP + FN}$$

Riporta il rapporto di quanti tra tutti i positivi la rete ne predica positivi.

Il valore *F1-Score* si ottiene da:

$$F1 = 2\left(\frac{Precision * Recall}{Precision + Recall}\right)$$

Ed è utilizzato come valore di bilanciamento tra *precision* e *recall*. Inoltre, è un'ottima misura per le prestazioni di una rete quando le sue classi non sono bilanciate.

I valori che abbiamo appena descritto rappresentano caratteristiche interne alle varie classi, i valori di *Micro Average* e quello di *Macro Average* riportano invece dati esterni alle varie classi.

Il valore di *Micro Average* aggrega i contributi di tutte le classi per calcolare il valore medio della misura. Il valore *Macro Average* invece calcola il valore della misura indipendentemente e poi ne esegue la media, trattando tutte le classi con lo stesso peso.

Il valore di *Accuracy* di un modello ne riporta invece l'accuratezza prendendo i valori predetti in modo corretto (TP) e dividendo per tutti i valori calcolati.

Un altro strumento fondamentale per l'analisi delle performance di un classificatore neurale è la matrice di confusione. Questa matrice riporta sulle righe gli elementi delle classi reali e sulle colonne le predizioni di questi elementi. Lo strumento è importante perché consente di vedere oggettivamente cosa viene predetto per ogni elemento e permette di capire se c'è sbilanciamento nell'apprendimento di alcune classi rispetto ad altre. Possono esistere classi che tendono ad attrarre elementi di altre o possono esistere classi che vengono completamente confuse per altre: i parametri introdotti prima danno risultati delle performance più statistici, mentre la matrice di confusione può mostrare chiaramente questi fenomeni.

1.5 Meccanismi di contrasto a Malware e Botnet

Per l'utente generico

Negli anni, contemporaneamente allo sviluppo dei malware, sono stati sviluppati meccanismi di difesa con i quali un utente possa difendersi dalle minacce informatiche. Passeremo quindi attraverso un breve excursus di quali strumenti può utilizzare un utente.

Lo strumento più conosciuto per difendersi dagli attacchi informatici e dall'infezione di malware è l'antivirus (7). Un antivirus è infatti un software progettato per prevenire, rilevare e rendere innocui codici dannosi.
Il motore di rilevazione malware di un antivirus utilizza diverse tecniche per rilevare un'infezione. Una di queste è la rilevazione della firma (signature o fingerprint) del malware. Per firma si intende una sequenza di byte comune ad alcuni modelli di malware. L'azienda che produce il software antivirus analizza tutte le minacce che le pervengono, associa loro tale firma che poi carica su un database online. Il software nella scarica questo database periodicamente e lo utilizza per confrontare la firma di ogni nuovo file nella macchina con le firme del database. Si dà per scontato che i file non nocivi non contengano firme presenti nel database. Questo metodo è un'ottima protezione per i malware più comuni, in quanto una maggiore diffusione aumenta la probabilità che il codice dannoso sia entrato in possesso dei ricercatori delle aziende di cybersecurity, che quindi possono generare il fingerprint del malware.

Ovviamente tale metodo non consente una protezione totale poiché non copre le minacce non note all'antivirus. Entra quindi in gioco la seguente tecnica detta di rilevazione delle anomalie. Questo frammento del software si occupa dell'individuazione delle minacce non presenti nei database dei malware e sfrutta diverse tecnologie, ad esempio la scansione della memoria alla ricerca di pattern anomali, il controllo su nomi sospetti o di file con grandezze dell'header irregolari. Più in generale ogni meccanismo che consenta la rilevazione di qualcosa di potenzialmente insolito può essere definita come appartenente a questa categoria. Mentre l'analisi delle signature se ben realizzata implica un numero di falsi positivi pressoché nullo, l'analisi euristica può generare sia falsi positivi che falsi negativi. È quindi molto importante predisporre una precisa sensibilità dell'algoritmo

che, se settato su parametri troppo restrittivi può generare un grande numero di falsi positivi, mentre se viene settato su parametri troppo laschi risulta inutile e non performante.

La grande diffusione dei maggiori software antivirus rende possibile l'utilizzo di alcune analisi telemetriche per giudicare il grado di nocività del software. Infatti, le compagnie possono registrare alcuni dati in tempo reale della macchina per ottenere informazioni sul comportamento del software. Per telemetria si intendono dati di utilizzo dinamici, quali: il consumo delle risorse della CPU, consumo di risorse della memoria, utilizzo di quali file, etc. Il confronto di diverse telemetrie all'utilizzo di file simili permette di comprendere con bassa precisione se un file è infetto o meno.

Nell'ultimo periodo vengono sempre più utilizzate tecniche di Data Mining per estrarre feature specifiche (tradotto spesso in "caratteristiche") dal codice in modo sistemico e valutarne il grado di malignità.
Infine, gli antivirus più complessi, prima dell'esecuzione del codice sulla macchina, possono eseguirlo in una Sandbox. Una Sandbox è un ambiente chiuso e controllato in cui l'antivirus esegue il software per studiarne il comportamento. Con dei parametri ben disposti, la tecnica risulta molto efficace e permette, da parte delle case che producono software antivirus, di testare i malware che vengono loro forniti in modalità automatiche per generarne poi le signature.
Gli antivirus, e più in particolare gli algoritmi di rilevazione, non proteggono l'utente da tutte le possibili minacce. Per questo motivo le case produttrici affiancano la vendita degli antivirus a diversi servizi volti a completare l'esperienza dell'utente.

Non tutti sanno inoltre che gran parte delle minacce informatiche vengono impedite da una buona costruzione del sistema operativo che gestisce la macchina. Il kernel, il nucleo del sistema operativo che si occupa di associare i processi ai core dei processori, e gli algoritmi del sistema operativo che si occupano di gestire le autorizzazioni di accesso a file e processi sono fondamentali in questo. Una parte del problema sta infatti nelle autorizzazioni che vengono concesse al codice che viene eseguito.

Microsoft con Windows, Apple con macOS e la comunità Open-Source[6] di Linux forniscono sistemi operativi con kernel ed implementazioni differenti. Tali kernel, insieme al resto del codice del sistema operativo che gestisce le autorizzazioni, possono prevedere un eventuale utilizzo inappropriato del sistema di autorizzazioni e prevenirlo, risolvendo parte del problema e rendendo l'intero sistema meno vulnerabile (17).

Una diversa implementazione, e quindi una diversa gestione dei permessi, fa sì che alcuni malware sfruttino vulnerabilità che possono essere specifiche di un sistema operativo e non degli altri (7). Per questo, le case o comunità suddette, al momento della distribuzione di un nuovo sistema operativo o di un suo aggiornamento rendono molto spesso il sistema più sicuro fornendo aggiornamenti sia al kernel che ai diversi moduli del sistema operativo, magari chiudendo le falle rilevate dall'aggiornamento precedente. È per questo molto importante che gli utenti si dotino di sistema operativi il più aggiornati possibile.

Concetto fondamentale inoltre si ritiene essere la necessaria capacità da parte dell'utente da discernere la pericolosità delle eventuali minacce. Soprattutto per quando riguarda i malware che si basano su ingegneria sociale, una buona educazione al web è fondamentale per la prevenzione delle infezioni informatiche. Avere un'idea grossolana di ciò che si scarica dal web o della opportuna grandezza di un allegato e-mail possono consentire all'utente di proteggersi a priori e rendere meno richieste le altre soluzioni.

Per gli analisti e i ricercatori

Coloro che lavorano nel campo della cybersecurity utilizzano diverse tecniche per effettuare ricerche sui malware e implementare soluzioni negli antivirus o nel codice operativo di un sistema. La più degna di nota, come già accennato, è la tecnica del Reverse Engineering. Tramite analisi del software e algoritmi opportuni è possibile risalire al codice sorgente di un software a partire dal suo comportamento ed in particolare a partire dal suo codice eseguibile. L'analisi attraverso questa tecnica è oggettivamente complicata in quanto gli algoritmi che ricostruiscono il codice sorgente non sono in grado di ricostruirne anche la leggibilità linguistica. Tutta la parte lessicale associata alle variabili ed ai costrutti viene persa: vengono persi i nomi dati alle variabili

[6] Sorgente aperta: il codice è visualizzabile, modificabile ed utilizzabile sotto licenza libera da chiunque lo richieda.

e tutti e commenti che potrebbero essere utili per decifrarne la complessità. Per quanto sia complicata, questa analisi è però fondamentale. I ricercatori possono capire e studiare più attentamente le funzionalità di un malware, ricostruirne le caratteristiche e le funzioni, nonché tentare di individuarne i possibili creatori.

Il RE, tra molte cose, ci permette di risalire a copie del codice sorgente degli algoritmi DGA (5). Risalire al codice permette di ottenere il seed di generazione aleatoria dei domini, e questo, come illustrato nel paragrafo sul domain fluxing, permette di bloccare le azioni delle botnet che utilizzano domini DGA per i server di comando e controllo.

Negli ultimi anni si fa sempre più utilizzo dell'intelligenza artificiale (o Artificial Intelligence, AI) per il riconoscimento e il tracciamento dei malware (18). Nel caso degli algoritmi per la generazione di domini, è possibile ottenere buoni risultati utilizzando algoritmi di classificazione basati sull'AI.
La ricerca redatta da Raaghavi Sivaguru et al. del 2018 (18) riporta i risultati delle migliori tecnologie basate sull'intelligenza artificiale adoperate nella rilevazione di malware a partire da log del traffico DNS. Gli strumenti più interessanti tra quelli analizzati sono reti neurali con diverse conformazioni adattate all'analisi di lunghe liste di domini web. Le AI della ricerca vengono addestrate su dei set di addestramento al fine di riconoscere i domini DGA di diverse famiglie di malware. L'idea è che poi, dopo aver trovato una soluzione che implementi suddetti algoritmi con bassi tassi di falsi negativi e di falsi positivi, sia possibile renderne efficace una implementazione in un algoritmo installabile su un server DNS. Per quanto già detto in precedenza, così facendo sarebbe possibile bloccare le attività nocive sia delle botnet che dei malware che utilizzino Domain Fluxing per nascondere il proprio server C&C.

Altri studi correlati, come quello di Kamal Alieyan et al. del 2017 (9), riporta come sia possibile effettuare analisi statistiche del traffico DNS oltre all'utilizzo di veri e propri classificatori. Alcuni strumenti, come quelli che studiano l'entropia delle varie richieste DNS hanno evidenziato che quelle legate ai malware tendono a disporsi temporalmente in modo diverso rispetto alle benigne: in ogni zona con lo stesso fuso orario si hanno ragionevolmente più richieste DNS durante il giorno, essendo queste legate all'attività dell'utenza umana. Le richieste malevole invece si muovono ad orari diversi e tendono a sovrapporsi a diverse zone orarie.

Un altro strumento citato nella ricerca di Alieyan, presentato da Stalmans e Irwin nel 2011 (10), evidenzia come sia possibile costruire una struttura di alberi decisionali per la rilevazione di domini generati da DGA analizzando l'intera richiesta fatta al server DNS, quindi non solo il dominio da risolvere, ma anche i parametri forniti in risposta dal DNS.

L'ultimo lavoro di ricerca che citeremo in questa parte, di Duc Tran et al. del 2017 (16), fornisce importanti informazioni su come costruire una rete neurale di tipo LSTM (Long Short-Term Memory) per sviluppare un sistema di rilevazione malware basati su DGA agendo su traffico DNS analizzato passivamente. Tran e associati forniscono inoltre su una repository pubblica lo strumento utilizzato per generare la rete ed addestrarla.

In particolare, il problema che Tran e i suoi colleghi si pongono di risolvere è dovuto ad un possibile sbilanciamento dei set di addestramento quando si parla di domini generati da algoritmi DGA (16). Esistono infatti alcuni algoritmi che generano un numero minore di domini rispetto ad altri. Il numero di domini generati non è però proporzionale alla ingenuità del codice, e nessuno vieta il fatto che un algoritmo che generi pochi domini sia associato ad un malware molto pericoloso. Per questo, la rete neurale durante la fase di addestramento avrà conosciuto verosimilmente meno domini di una famiglia di quanti possa averne conosciuti di un'altra. I ricercatori citati hanno introdotto un meccanismo di Multiclass-Imbalance che permette di risolvere il problema assegnando un peso maggiore ai domini meno numerosi nel set di addestramento. Viene modificato l'algoritmo di retropropagazione dell'errore agendo sulla magnitudine del costo che viene associato agli errori di predizione più grandi fatti in fase di addestramento dalla rete. Gli errori più consistenti, esclusi i primissimi cicli, vengono commessi sul riconoscimento degli elementi mai visti dalla rete: un minor numero di elementi di una classe fa in modo che siano più grandi gli errori che compaiono quando la rete ne vede gli elementi per le prime volte. L'adeguamento che viene fatto sulla rete in seguito ad un errore (definito in inglese *retrofit*) viene maggiorato dall'algoritmo di Multiclass-Imbalance nel caso in cui l'errore fatto in fase di predizione è sufficientemente grande. In questo si aumenta il focus dell'addestramento della rete sulle classi con meno elementi e questo consente di ottenere un maggior bilanciamento dei risultati delle performance totali di riconoscimento delle varie classi.

Lo sviluppo di questo lavoro si basa proprio sulle ricerche appena citate. Vedremo nei prossimi capitoli come ottenere dati di traffico DNS passivo, formattarli adeguatamente ed analizzarli. Successivamente, assieme a questi assoceremo informazioni su domini sicuramente malevoli per istituire un ottimo set di addestramento nonché un efficace parametro di confronto. Verrà poi generata una rete neurale LSTM.MI basandosi sull'algoritmo fornito da Tran et al.. Rete che poi verrà testata su dati reali ottenuti facendo una selezione dei dati non utilizzati per l'addestramento. Confronteremo poi le performance di questa rete con i dati contenuti nelle ricerche riportate sopra per valutarne l'attuale performance.

2. Raccolta dei dati

2.1 Risorse

Log DNS rete GARR

Come già brevemente introdotto, lo scopo del DNS è quello di indirizzare ogni nodo al dominio che sta cercando online. Ogni rete, piccola o grande che sia, se non ne possiede uno, ha un nodo che punta ad un DNS realmente esistente.

La prima istituzione della rete GARR risale al 1991 ad opera del Consorzio per l'Armonizzazione della Rete della Ricerca (19). L'obiettivo dell'infrastruttura nazionale è quella di collegare gli istituti di ricerca universitaria con una rete a banda ultra-larga. Questo per rendere possibile un canale di comunicazione all'avanguardia tra le università e i centri di ricerca nonché tra questi e le principali infrastrutture europee. La rete ad oggi connette con banda ultra-larga molte delle università italiane, compresa l'Università Politecnica delle Marche.

Una rete così grande e con intenti così specifici può avere la necessità di disporre di un server DNS interno. In questo caso il DNS della rete, essendo questa accessibile solamente dai dispositivi connessi agli access-point delle università italiane, risolve solamente le richieste fatte da macchine che si trovano connesse alla sua rete. Ad oggi è gestita dal Consortium GARR, fondato nel 2001 da CNR, ENEA, INFN e Fondazione CRUI con il patrocinio del Ministero dell'Istruzione.

Grazie alla collaborazione degli amministratori della rete, il Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche ha accesso ai log DNS dell'infrastruttura. Per log si intende un file di testo che elenchi in forma leggibile ad un programmatore tutti gli eventi che la macchina registra in un determinato settore di competenza. I log delle query[7] DNS conterranno quindi informazioni riguardo ogni

[7] Query DNS, tradotto come richiesta DNS, rappresenta l'atto e la richiesta di informazioni che una macchina fa ad un DNS per l'eventuale risoluzione in indirizzo IP di un dominio.

singola richiesta di risoluzione di dominio con, dato fondamentale, la stringa di dominio che la macchina che si connette al DNS ha necessità di risolvere.
Questo ha reso possibile uno studio dettagliato della mole di dati generata dall'intera rete ed una analisi passiva (non in fase di risoluzione ma successiva) di tutto il traffico internet che vi passa attraverso.

La macchina a cui il Dipartimento ha accesso genera ogni giorno 24 log di traffico DNS. Ogni log corrisponde al traffico passato per il nodo in una determinata ora del giorno. Se si prende come campione l'insieme dei log generati nella settimana che va dal 12 al 18 Settembre 2019, per ogni ora di registrazione la mole di dati è pari a $k10^6$ query DNS dove k varia da circa 3 per le ore notturne del finesettimana a circa 6 dei giorni feriali in orario lavorativo. Arrotondando per difetto sul risultato di un breve calcolo basato sulla dimensione approssimata dei dati in Megabyte e del numero di stringhe che ogni dato contiene, si ottengono circa 8.5 milioni di query totali per i giorni feriali e circa 4.5 milioni di query per i festivi. Nei giorni successivi alla settimana presa in esame la dimensione dei dati aumenta compatibilmente alle riaperture agli studenti delle università italiane e all'inizio dei corsi. Si ha infatti per alcuni giorni di Ottobre 2019 un k relativo a orari lavorativi nei giorni feriali uguale a 12. Inoltre, per ogni fine settimana preso in esame la quantità di query diminuisce, compatibilmente al numero delle attività umane eseguite nei plessi universitari durante i giorni festivi.

L'utilizzo della rete GARR per questo studio rispetto ad una rete generica o più comune lo rende un caso particolare: per i DNS delle più grandi reti diffuse a livello mondiale non ci si aspettano fluttuazioni delle query così ampie. Un DNS imponente come quello fornito da Google.com all'indirizzo IP 8.8.8.8 è utilizzato da utenti in tutto il mondo e questo attenua notevolmente le fluttuazioni dovute agli orari notturni: ad alcune zone notturne corrisponderanno zone del globo in cui si ha il giorno a bilanciare i numeri. Per questo ragionamento si è supposta una disposizione più o meno uniforme della popolazione sul globo e si è fatta l'assunzione che la macchina DNS all'indirizzo indicato sia un'unica macchina o rete di macchine che non reindirizzi a DNS locali per accelerare la fase di trasmissione della query e del dominio risolto. Il discorso non vale però per le macchine fornite dagli operatori telefonici nazionali. In questo caso, coprendo un'unica zona

ci si aspetta una fluttuazione più simile alla nostra rete, ma con notevole differenza rispetto alla tipologia dell'utente.

Il lavoro si basa su un orizzonte temporale che va indicativamente dal 1° settembre al 31 Ottobre 2019. Si considera quindi un periodo che dal punto di vista delle quantità di dati può essere mutevole, ma non eccessivamente. Inoltre, è un periodo in cui le università che appartengono alla rete GARR redigono corsi e attività. Questo fa anche in modo che le query DNS registrate, supponendo un'indagine piuttosto generica, siano effettuate da utenti che siano professori ed impiegati ma anche ed in gran numero studenti. Va fatto notare quindi che il campione analizzato non è statisticamente omogeneo. Si presuppone che una rete universitaria colleghi utenti con consapevolezza delle proprie azioni online superiore alla media nazionale e ci si aspetta quindi di trovare meno infezioni per numero di query rispetto ad una rete generica. Non avendo un parametro di confronto non ci addentreremo ulteriormente nel discorso che sarebbe interessante approfondire.

Supponendo quindi un numero di query giornaliere di 8 milioni e moltiplicando per i 61 giorni dello studio si ha indicativamente un numero di richieste DNS compatibile con i 500 milioni di query totali. Vedremo in seguito come una ottimizzazione del codice degli script che gestiscono l'analisi dei dati non renderà questo grande numero un problema dal punto di vista dell'esecuzione. Anche per questo è mantenuta una vaga precisione nel dimensionamento del numero che non è necessariamente così interessante.

La visita sistematica degli archivi del server DNS del GARR permette al Dipartimento di Ingegneria dell'Informazione di poter salvare su un server locale i file contenenti le informazioni sulle query in formato *.log*. L'estensione è relativa ad un file di testo e questi vengono poi compressi per salvare spazio in memoria della macchina (20). Per ottenere accesso alle risorse si è effettuata una copia delle informazioni sulle macchine sulle quali è stato eseguito lo studio. Gli algoritmi utilizzati per l'analisi scompattano i file dall'archivio se ne hanno necessità o lavorano direttamente sui file compressi per ottenere maggiore efficienza spaziale.

Per un esempio di qualche riga di un log DNS si rimanda all'appendice.

Feed Bambenek Consulting

Con l'aumento dell'utilizzo da parte dei malware degli algoritmi di generazione di domini è anche aumentato il numero di organizzazioni che effettuano ricerca su questi ultimi. Tra queste, il Bambenek Consulting[8] è un consultorio di investigazioni e intelligence di cybersecurity che si focalizza sul contrasto delle grandi minacce criminali online. Questa organizzazione mette a disposizione una delle più importanti risorse utili agli agenti del settore.

Il Bambenek Consulting Feed (abbreviato da qui in avanti in Bambenek Feed o BF) è un feed online fornito dal Bambenek Consulting che contiene diverse informazioni. Tra queste, i dati nel feed degli Algoritmi di Generazione di Dominio rappresentano una blacklist di possibili domini malevoli associati a diverse famiglie di DGA (21).

Il feed contiene una lista aggiornata ogni 24 ore dei possibili domini che tali famiglie di malware possono contattare in un intervallo di tempo centrato nel giorno in cui il feed viene emesso. In altre parole, poiché alcune famiglie di DGA Time-Dependent generano possibili domini dei C&C server che questi server possono assumere solamente in un determinato intervallo di tempo (ad esempio, 5 giorni per Crytpolocker, 3 giorni per altre diverse famiglie minori), il feed giornaliero viene centrato sul giorno in cui viene emesso. Significa che se sto prendendo in esame una famiglia di DGA che utilizza i propri domini per 5 giorni e so che il dominio verrà generato il giorno 16, troverò il dominio all'interno del feed già dal giorno 14 che rimarrà in ogni edizione giornaliera del feed fino al giorno 19 non compreso. Allo stesso modo, se la famiglia di malware utilizza i propri domini per un tempo di 3 giorni, il dominio generato il 16 sarà presente dal 15 al 17. Fuori dal loro breve periodo di validità i domini vengono infatti rimossi. Ovviamente sta ai ricercatori che forniscono informazioni sulle diverse famiglie a quantificare opportunamente la finestra di utilità dei domini generati. Come già indicato, ogni famiglia di DGA-TD possiede la sua finestra a scorrimento.

I domini delle famiglie DGA Time-Independent hanno per definizione validità molto più estese e vengono aggiunti e rimossi in base all'occorrenza.

[8] http://www.bambenekconsulting.com/

Il feed contiene diverse informazioni per ogni nome di dominio:

- Nome di dominio
- Famiglia del DGA
- Giorno di inserimento nel feed (vedi sopra)
- Link al manuale del Bambenek Consulting per quella famiglia

Per valutare la quantità di informazioni contenute in ogni feed giornaliero prendiamo in considerazione quello generato il 16 Settembre 2019.

La dimensione del set è di circa $8*10^6$ nomi di dominio malevoli forniti ogni giorno. Rispetto ai dati GARR ha meno senso moltiplicare il dato per i 61 giorni del campione generico in quanto i domini all'interno del feed vengono costantemente ripetuti (vedi finestre a scorrimento dei TD o i domini TI).

Esattamente come per i dati del GARR, anche la dimensione in MB del feed di Bambenek Consulting e la complessità computazionale delle operazioni eseguite sul set di dati non risultano significative a patto che in fase di progettazione degli script che li utilizzano siano previsti meccanismi di gestione efficienti.

Bambenek inoltre fornisce un feed ulteriore denominato High-Confidence DGA Feed. L'unica differenza con il feed preso in esame risiede nell'assenza delle famiglie giudicate Medium o High FP-Risk, cioè nell'assenza di famiglie che generano un coefficiente di falsi positivi più alto. Non è stato possibile approfondire la questione poiché non vengono forniti i valori numerici del FP-Risk. Utilizzeremo comunque il feed standard in quanto Bambenek Consulting indica pochi domini ad alto rischio di falso positivo e non c'è una differenza talmente significativa da far preferire l'High-Confidence a quello standard.

Il feed viene fornito come risorsa online disponibile al download[9] sotto forma di file in formato di testo *.txt*. Per un utilizzo più efficiente e per non oberare la memoria della macchina su cui ne è stato eseguito il

[9] Download al link: https://osint.bambenekconsulting.com/feeds/dga-feed.txt

download, lo script redatto per scaricare la risorsa ne implementava la compressione in formato *.xz*[10]. Gli algoritmi che vedremo successivamente scompattano la risorsa se ne hanno la necessità o lavorano direttamente sul file compresso quando ne hanno la possibilità per ottenere maggiore efficienza spaziale.

Come per i log del DNS della rete GARR, per un esempio di qualche riga del feed del Bambenek Consulting si rimanda all'appendice.

Majestic Million

Se con il confronto dei domini che compaiono nei log DNS della rete GARR con quelli che compaiono nel feed Bambenek è possibile ottenere informazioni su domini realmente richiesti e sicuramente malevoli, è importante anche poter assegnare informazioni a domini richiesti che siano però sicuramente benevoli.

Nelle ricerche citate si utilizzano i domini forniti da AWS Alexa[11] (strumento indicato anche come Amazon Alexa, essendo AWS una sussidiaria del grande marchio di e-commerce) che è un sistema di reputazione e analisi del traffico web. In particolare, della suite Alexa, il servizio top-sites permette a pagamento di ottenere tramite API la lista dei domini con più connessioni in un determinato periodo.

Per questo lavoro si è però scelto un servizio analogo, completamente gratuito, che consente di ottenere risultati praticamente identici.

Lo strumento offerto dalla compagnia inglese Majestic[12], detto Majestic Million (22), fornisce un elenco composto dal primo milione di domini con il maggior numero di sottoreti di provenienza ottenuto attraverso vari strumenti di mappatura del web di proprietà dell'organizzazione. Il feed dei domini viene aggiornato ogni giorno attraverso un indice che la società ottiene analizzando circa 500'000 milioni di pagine web e le relative URL sulle pagine.

---

[10] Simile alla compressione in *.zip*, è un formato di compressione senza perdite operato dall'algoritmo Open-Source (licenza GNU-LGPL) LZMA2 supportata dai sistemi Linux che è nel caso dei file di testo più efficiente dei più diffusi *.zip* o *.rar* o *.7z*.

[11] https://www.alexa.com/topsites

[12] https://it.majestic.com/

Lo strumento, molto similmente a come viene utilizzato AWS Alexa nelle ricerche citate, fornisce un sistema di reputazione dei domini che contiene in quanto, se questi domini sono molto visitati e con molte sottoreti e in più vengono analizzati da una azienda leader nel settore della mappatura del web, si presuppone che siano domini leciti e non maligni.

A prova di ciò, prima di effettuare ulteriori analisi e per valutare l'utilità dello strumento, si è deciso di prendere un feed campione del Majestic Million (abbreviato anche in MM) e confrontarlo con il feed DGA emesso dal Bambenek Consulting dello stesso giorno. Sono stati presi entrambi i feed del giorno 30 settembre e con un semplice script scritto in linguaggio Python3 si è andati alla ricerca di eventuali stringhe di domini che comparissero in entrambi i feed. Grazie al risultato nullo dell'intersezione ed alle assunzioni fatte prima sulla necessaria benignità dei domini, si è potuto considerare lo strumento utilizzabile.

Come viene chiaramente indicato dal nome della risorsa, essa contiene esattamente un milione di domini definibili positivi. Anche in questo caso, con accorgimenti di progettazione è possibile considerare la dimensione del set irrilevante ai fini dello svolgimento del lavoro.

Diversamente da come si fa ovviamente con i log del DNS della rete GARR e da come si fa con il feed dei DGA del Bambenek Consulting, per quanto riguarda il feed del Majestic Million non si è ritenuto necessario il costante aggiornamento e download giornaliero della risorsa. Una breve analisi effettuata su 5 feed successivi ha indicato che la differenza tra i domini che appaiono nei 5 giorni è minima. Si ha infatti una variazione minima di pochi punti percentuali dei domini presenti nei feed emessi nei giorni che vanno dal 28 Settembre 2019 al 2 Ottobre 2019. Inoltre, il feed è rivolto ad un pubblico internazionale ed è realizzato non tenendo conto della provenienza nazionale dei domini. Per questo, oltre alla minima variazione giornaliera, si suppone che la variazione giornaliera dei domini nazionali italiani sia anch'essa minima. L'analisi dei domini nazionali viene fatta poiché essendo la rete GARR una rete nazionale circoscritta al territorio italiano, ci si immagina che tra i domini che il suo DNS debba risolvere sia presente una grande fetta di domini localizzati in Italia, quindi meno frequentati da utenti nel resto del globo e sicuramente in proporzione meno frequentati rispetto ai domini a fruizione internazionale.

Il feed viene fornito attraverso download di un file *.csv* dal sito della risorsa[13]. Come per gli altri feed, per comodità si è deciso di comprimere il file mediante funzione specifica della routine che esegue il download della risorsa quando necessario. Per il formato di compressione, sempre quando necessario, si è scelti il formato *.xz* come negli altri casi.

[13] Al link https://it.majestic.com/reports/majestic-million

2.2 Script

Python

La totalità degli script scritti per i fini di questo lavoro è progettata in linguaggio Python.

Python è un linguaggio di programmazione software dinamico e di alto livello, orientato agli oggetti ed utilizzabile per molti tipi di sviluppo software. Offre un forte supporto all'integrazione con altri linguaggi e programmi, è fornito di una estesa libreria standard ed è relativamente semplice da apprendere.

Anche se orientato agli oggetti, non è eccessivamente ferreo a riguardo e consente la scrittura efficiente di codice procedurale. Questo lo rende un linguaggio molto pratico e adatto, tra i tanti usi, alla programmazione distribuita, allo scripting, alla computazione numerica ed al system testing. Il fatto che non venga direttamente interpretato in linguaggio macchina, ma che passi per la generazione di un byte code, lo rende molto adatto alla programmazione multipiattaforma pur restando sufficientemente rapido.

La totalità degli script creati da zero è scritta in Python3 e più nel dettaglio nella versione 3.7 del linguaggio. Le librerie esterne rispetto a quelle fornite dalla piattaforma del linguaggio vengono utilizzate per lo più per ottenere una maggiore comodità nella gestione dell'esecuzione e vengono prelevate tra quelle ufficialmente supportate dal linguaggio, quindi si può considerarne sicuro l'impiego.

Per quanto riguarda invece gli script forniti da altri, essi sono redatti in Python2.8 e ne è stato necessario l'aggiornamento alle librerie più recenti.

Principali problematiche ed accorgimenti

Questo lavoro si basa su set di informazioni molto grandi. Abbiamo già definito grossolanamente qualche grandezza. Come ben definito della teoria sulla complessità degli algoritmi, un semplice confronto tra vettori, quando si hanno in mano vettori con dimensioni di milioni di elementi, può risultare troppo complesso a livello computazionale e può richiedere molto tempo per essere eseguito. In questo caso si parla di elevata complessità computazionale temporale. Nel caso di set di dati così grandi si parla anche di elevata complessità spaziale, in quanto il salvataggio di tutte le risorse in memoria può generare problemi di necessità di spazio.

Trattandosi di un lavoro così delicato, con numero sufficiente di responsabilità nell'eventuale caso in cui venisse realizzata una implementazione, è anche richiesto un certo grado di confidenza riguardo i risultati ed i dati prodotti.

Queste due problematiche hanno portato alla necessità di prendere alcuni accorgimenti per la risoluzione puntuale degli esperimenti e per consentire un comodo utilizzo degli strumenti software prodotti.

Per quanto riguarda la complessità degli algoritmi utilizzati si è scelto di ricorrere a soluzioni performanti di alto livello unendo ciò alla necessità di utilizzare strumenti semplici ma efficaci. Quindi si è voluto utilizzare strutture e tipi di dato già presenti in Python. Per risolvere sia il problema del confronto sia il problema della dimensione dei set si è scelto di utilizzare il tipo di dato built-in di Python *dict*, che sta per Dizionario.

Per Python un dizionario è un insieme di associazioni chiave-valore strutturato similmente ad una tabella hash. Una tabella hash è una struttura dati che supporta l'indicizzazione secondo hash e assegna ad un codice numerico, l'hash appunto, un unico elemento. Può avvenire, nel caso in cui la tabella si avvicini all'essere sovraccarica, che ad un hash venga associata una lista e non un unico elemento. Il processo rende la ricerca di un elemento all'interno della tabella molto più veloce. Infatti, quello che viene fatto operativamente è generare l'hash dell'elemento che si vuole cercare all'interno di un dizionario e verificare che nella zona destinata all'hash appena ottenuto sia presente o meno un elemento. Se l'elemento è presente, si scorrerà al massimo una lista sufficientemente breve, nel miglior caso basterà confrontare un unico elemento. Il confronto di due dizionari, prendendo un elemento di uno e cercandolo nell'altro, non richiede quindi un confronto tra due vettori, nel quale si richiederebbe lo scorrere interamente il secondo alla ricerca dei valori contenuti nel primo

per tante volte quanti sono gli elementi nel primo vettore, bensì un'operazione con tempo assimilabile a quello di un unico confronto.

Assumendo che la dimensione dei due vettori sia assimilabile al parametro n, si dice che la complessità computazionale della ricerca degli elementi di un vettore all'interno di un altro nel caso peggiore sia $O(n^2)$ in quanto vengono effettuati n*n confronti. Nel caso della tabella hash, assumendo che sia ottimizzata, la complessità è dell'ordine di O(n), in quanto la ricerca di un elemento della prima nella seconda richiede soltanto un confronto, quindi n*1 totali. Utilizzando la struttura dizionario rispetto ad una più semplice si riesce a risolvere il problema della complessità temporale: il numero di confronti cala di un ordine di grandezza assimilabile a 10^6, non poco quindi.

Una tabella hash ottimizzata inoltre occupa uno spazio leggermente maggiore, ma assimilabile all'ordine di grandezza del numero di elementi che contiene. Si è quindi scelto di trascurare l'ingombro spaziale a favore di una maggiore velocità di esecuzione.

Ogni volta che quindi sarà necessario caricare i dati contenuti nei tre feed presentati nei paragrafi precedenti, si è scelto di utilizzare tali strutture. Nelle associazioni chiave-valore dei dizionari, la chiave che viene utilizzata è la stringa di dominio che si vuole analizzare, mentre al valore viene associato un oggetto che contiene diverse informazioni, differenti per ogni feed.

Un ulteriore accorgimento sta nell'evitare che vengano caricati troppi dati in memoria, per non sovraccaricare l'esecuzione. Nel caso in cui, ad esempio, dovesse essere necessario effettuare il confronto di dati contenuti in diversi giorni con una delle altre risorse, si preferisce caricare in memoria i dati di un giorno alla volta, snellendo l'esecuzione.

Ulteriori approfondimenti sul codice utilizzato verranno presentati più avanti e nell'appendice.

2.3 Analisi Preventive

Prima di eseguire gli esperimenti finali, sono stati eseguiti dei test e dei brevi esperimenti per valutare la fattibilità e le opportunità di ricerca che erano disponibili per mezzo dei dati disponibili esposti nei paragrafi precedenti. Inoltre, sono serviti da introduzione ai concetti base degli strumenti e a comprendere meglio come questi si possono confrontare.

Un primo test iniziale è stato fatto per comprendere le dimensioni delle intersezioni tra domini sicuramenti malevoli appartenenti al feed del Bambenek Consulting e i domini realmente richiesti al server DNS della rete GARR. Per effettuare questo primo breve esperimento è stato scritto ex-novo uno script in linguaggio Python che fosse in grado di caricare in memoria tutti i domini che comparivano in entrambi i set, che sapesse confrontarli e che poi potesse redigere qualche piccola statistica. Si è scelto di effettuare questo primo test confrontando 5 giorni di log con un feed BF generato nel giorno centrale dei 5. I log scelti sono quindi quelli relativi al periodo tra il 14 e il 18 Settembre 2019 mentre il BF scelto è quello del 16 stesso mese.

Data	Query malevole	Query Malevole & NXDOMAIN	Domini malevoli unici	NXDOMAIN totali
2019-09-14	1102	1012	100	2828259
2019-09-15	401	368	100	2103694
2019-09-16	7982	7474	2105	3204748
2019-09-17	6318	5803	2104	3779819
2019-09-18	4610	4303	2089	3979982

Tab. 1: Risultati schematizzati del primo test sui dati.

Come è possibile notare nella *Tab. 1* il numero di query malevole che sono apparse nei log DNS nei giorni del periodo sono consistenti. Il server DNS non ha risposto a tutte le query malevole con NXDOMAIN ad indicare la forte possibilità che, tra tutte le richieste, qualche macchina sia riuscita a mettersi in controllo con almeno un server C&C.

L'aumento notevole dei domini malevoli unici del giorno 16 è stato associato a diverse possibili cause. Si è ipotizzato infatti che potesse essere dovuto ad un fisiologico aumento del traffico (il 16 Settembre 2019 è un lunedì), ad una possibile infezione proprio il 16 di almeno una nuova macchina della rete, al centramento del feed Bambenek in quel giorno o ad un diverso scheduling dei malware.

Proseguendo l'analisi, dal giorno 15 al giorno 16 viene registrato un aumento di un fattore approssimato a circa 20 dei domini malevoli unici, un aumento di un fattore approssimato a 2 dell'intera mole di dati e un aumento di un fattore approssimato 1.5 degli NXDOMAIN.

Viene notato che i domini malevoli unici dei giorni 14 e 15 appartengono interamente alla famiglia di malware Vawtrak. Confrontando gli hash dell'indirizzo IP si conclude che tali richieste sono state effettuate da una sola macchina infetta che ha effettuato 100 tentativi di connessione al server C&C ogni ora per qualche ora al giorno. I restanti domini unici che compaiono a partire dal 16 settembre appartengono alla famiglia Necurs per la maggior parte. Entrambe le famiglie di domini Vawtrak e Necurs sono generate da DGA Time-Independent.

Tali analisi permettono di poter riprendere alcune cause ipotizzate del fenomeno dell'aumento dei domini del 16 Settembre: probabilmente non si tratta di aumento di traffico, visto che gli altri valori legati al traffico aumentano con fattori diversi; non abbiamo informazioni aggiuntive sulla nuova infezione e non possiamo ottenerne; per evitare la centratura dei feed Bambenek, per i prossimi esperimenti verrà adoperato per l'intersezione il feed del giorno in cui sono stati emessi i log.

Il secondo test preliminare viene effettuato al fine di testare ed ottenere un set di domini sicuramente benevoli, in prospettiva di doverci addestrare un classificatore. Similmente a quanto fatto nel primo breve esperimento, viene effettuata un'intersezione tra i domini presenti nel log della rete GARR con quelli presenti nel Majestic Million. L'implementazione è molto simile al caso precedente ed il periodo preso in esame va dal 21 al 30 settembre compresi.

Data	21/09	22/09	23/09	24/09	25/09	26/09	27/09	28/09	29/09	30/09	Totale
Domini Aggiunti	6568	2088	11902	8083	6138	5284	3519	920	621	4259	49382

Tab. 2: Risultati schematizzati del secondo test sui dati.

Nella *Tab. 2* vengono riportati i dati dei domini unici aggiunti giornalmente al set di domini sicuramente benevoli. Il picco del primo giorno è dovuto al fatto che si parte da un file vuoto, mentre il picco del 23 è sicuramente dovuto ad un aumento del traffico (il giorno 23 Settembre 2019 è un lunedì). Compatibilmente

con quanto ci si aspetta, Lunedì 30 vengono aggiunti meno domini unici. E durante la settimana si ha un progressivo calo.

Per verificare la bontà del dato ottenuto, parallelamente a questo viene redatto un set di domini sicuramente malevoli richiesti al DNS nello stesso periodo. Viene poi effettuata l'intersezione tra i due che si presenta vuota. Questo indica che si può con certezza affermare che a partire dalle nostre risorse, i domini identificati come malevoli sono sicuramente malevoli e che quelli identificati come benevoli sono sicuramente benevoli.

Una breve analisi dell'intersezione tra log e feed del Bambenek Consulting mostra che le famiglie di DGA presenti sono solamente 3. Abbiamo esempi di domini: Necurs, Vawtrak e Dromedan.

Si è scelto quindi di effettuare una breve ricerca online per comprendere con cosa abbiamo a che fare e quali sono le potenziali minacce associabili ad ogni famiglia.

Per quanto riguarda Necurs un articolo redatto da Microsoft (23) lega i nomi di dominio che abbiamo trovato ad una Botnet con sede in Russia. La rete, nei giorni dello studio condotto da Microsoft, si occupa soprattutto di spamming e scamming, ma possiede le potenzialità nascoste per effettuare attacchi DDoS e per il furto di dati. Questa rete, negli anni, è stata veicolo di distribuzione di diversi malware e trojan più nocivi, come ad esempio il trojan *Zeus*. Nei circa 60 giorni dello studio la botnet è stata in grado di inviare circa 3.8 milioni di e-mail spam, per un numero potenziale di vittime che si aggira attorno ai 40 milioni.

Vawtrak invece risulta essere un "Trojan Bancario" (24) adibito al furto di informazioni. Nonostante sia definibile più datato, in quanto informazioni online sul conto del malware risalgono soprattutto al periodo che va tra il 2014 ed il 2016, è noto che sia un malware ad alta frequenza di aggiornamento. Questo probabilmente è il motivo della sua presenza affermata anche oggi.

Dromedan invece è l'unione di un meccanismo adware finalizzato al download illecito di contenuti senza supervisione dell'utente unito ad un trojan che ne facilita l'installazione e l'esecuzione. Purtroppo, non si hanno molte notizie affidabili a riguardo, ma se ne trovano informazioni soprattutto nei forum dei siti web degli antivirus in cui gli utenti chiedono a dei tecnici metodi per la rimozione.

2.4 Dataset per gli esperimenti

Per la redazione dei dataset finali con cui poi istruire e testare le reti neurali si sono uniti gli approcci già visti. Si è infatti realizzata sia l'intersezione dei domini sicuramente malevoli del feed del Bambenek Consulting con i log del DNS della rete GARR, sia quella dei log con i domini del Majestic Million. In questo modo è stato possibile ottenere due dataset: nel primo vengono indicate le stringhe di dominio sicuramente benevole con l'etichetta *legit* che ne indica la legittimità, nel secondo alle stringhe di dominio sicuramente malevole viene aggiunta l'etichetta *dga* ad indicarne il fatto che sono generati da malware ed in più viene riportata un'altra stringa ad indicarne la famiglia.

Legit	
Alexa	91080
DGA	
Vawtrak	100
Necurs	30458
Dromedan	6
Ramnit	1
Tinba	1
Totale	**121646**

Tab. 3:La tabella riporta il numero di domini per ogni famiglia ottenuti dalle intersezioni sui giorni scelti.

Legit	
Alexa	30000
DGA	
Vawtrak	100
Necurs	15000
Dromedan	6
Ramnit	1
Tinba	1
Totale	**45108**

Tab. 4: La tabella riporta il sottoinsieme delle intersezioni ottenute che verrà poi utilizzato nelle fasi successive.

Per ottenere un primo e grezzo dataset dei domini sono stati presi 29 giorni di log distribuiti tra Settembre e Ottobre 2019. In particolare, sono stati scelti i giorni 16, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 Settembre, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 17, 25, 26 Ottobre. Tutti i suddetti giorni di log sono stati confrontati con i rispettivi 29 giorni di feed Bambenek Consulting, mentre per il Majestic Million sono stati presi i feed dei giorni 16 e 26 Settembre, 6 e 17 Ottobre. Come già espresso, è necessario prendere i feed giornalieri del Bambenek Consulting per evitare di avere problemi di centratura del feed, mentre non è necessario essere così stringenti per i feed del Majestic Million che vengono aggiornati con minor frequenza.

Di questo dataset grezzo, verrà poi utilizzato un insieme più piccolo. Questo insieme è generato da un numero

minore di elementi del primo dataset, elementi presi in ordine casuale in modo da inficiare il meno possibile con l'imparzialità del test.

La resezione del dataset viene effettuata poiché il numero di elementi del dataset grezzo veniva considerato troppo imponente e si è supposto non fosse necessario considerarne la totalità.

I dataset finali, al termine della loro elaborazione, rappresentano la principale risorsa da dare in pasto ai prossimi esperimenti. Per fare ciò verranno comunque modificati e adattati ad ogni tipo di prova che si sta per fare.

Queste informazioni vengono scritte, dagli script che le ricavano, in formato testuale su di un file *.csv*: questo formato di file rappresenta *Comma Separated Values* (CSV appunto) disposti su matrici di righe e colonne. Sono quindi interpretabili come fogli elettronici e nel nostro caso nella prima colonna contengono la dicitura dell'etichetta relativa al tipo di dominio, se legittimo o DGA, nella seconda colonna contengono l'etichetta relativa alla famiglia, nella terza colonna il dominio web che i successivi esperimenti dovranno analizzare e nella quarta il dominio originale.

Di questo set appena illustrato ne viene effettuata una copia con domini elencati in formato leggermente diverso. Per motivi che approfondiremo meglio nel seguente capitolo, vengono effettuati dei test su domini in configurazione detta Testa-Coda (TC o HT dall'inglese Head-Tail). Un dominio in formato normale ad esempio *univpm.it* può essere riscritto in formato testa-coda in *univpmit* semplicemente rimuovendo il punto che separa il dominio di primo livello, *it*, dal dominio di secondo livello, *univpm*. Per domini che possiedono più di due livelli, come ad esempio *ingegneria.univpm.it*, il dominio testa-coda si traduce nella rimozione dei livelli tra il primo e quello di livello massimo e nella rimozione dei punti nel mezzo, *ingegneriait*.

Introdotta questa classificazione dei domini, chiameremo i domini interi, non h-t, in configurazione *full*.

Infine, in entrambi i dataset viene inserito un unico dominio fittizio benevolo molto lungo che consente agli script che verranno utilizzati successivamente di essere eseguiti senza interruzioni. Il bug dipende dal fatto che

le librerie che vengono utilizzate[14] formattano i dati su delle matrici vettoriali. Tali matrici, affinché possano essere manipolate attraverso moltiplicazioni o confronti, richiedono delle dimensioni specifiche che nel nostro caso si adeguano al dominio con lunghezza di caratteri maggiore. Il dominio molto lungo viene inserito per permettere il confronto tra la matrice dei domini di test e quella di addestramento. In ogni caso, visto il numero di elementi contenuti nel dataset, si presuppone che l'aggiunta di un solo dominio fittizio non costituisca un fattore di deterioramento dei risultati ottenuti negli esperimenti che vedremo.

[14]Pandas, al riferimento https://pandas.pydata.org/, e Numpy, al riferimento https://numpy.org/.

3. Esperimenti

3.1 Esperimento 0

Illustrazione e preparazione

Il primo esperimento consiste nel testare un classificatore neurale pre-addestrato dal Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche sui set di dati ottenuti nel capitolo precedente, dalle intersezioni dei log DNS della rete GARR con il feed DGA del Bambenek Consulting e i domini del Majestic Million.

Questo viene fatto utilizzando uno script fornito dal Dipartimento che prenda in input il modello della rete, i dataset generati in questo lavoro e un dataset di riferimento per la strutturazione delle etichette. Lo script, dopo l'analisi delle performance della rete, restituisce a schermo e in output su un file di report i valori numerici relativi alle capacità di riconoscimento.

Il classificatore fornito dal DII è composto da due reti neurali: la prima effettua un'analisi sui domini in configurazione *full*, la seconda sui domini in configurazione *h-t*. Entrambe le reti effettuano sia una classificazione binaria che indichi se il dominio è malevolo o benevolo, sia una classificazione multiclasse che indica la famiglia di appartenenza del dominio. È ovvio che un dominio che viene identificato come benevolo non verrà identificato come appartenente ad una famiglia di DGA.

Questo classificatore è addestrato su un grande set di dati che ingloba alcune famiglie di DGA presenti nel Bambenek Feed oltre ad alcune famiglie non presenti che hanno richiesto approcci di ingegneria inversa per ottenerne i nomi di dominio richiamabili. Inoltre, tale set di addestramento contiene elementi per ogni famiglia in numero opportunamente dimensionato. Si rimanda alla tesi del collega Lorenzo Cicchini per ulteriori informazioni sul lavoro di allestimento del dataset.

Lo script di testing, previsto per un diverso esperimento, ha necessitato diverse modifiche affinché potesse essere funzionante e affinché fornisse dati completi ed affidabili. In particolare, lo script prevedeva che il set di training della rete LSTM e il set di test fossero due sottoinsiemi esclusivi dello stesso insieme di dati più grande e che contenessero entrambi le stesse famiglie di DGA. Le modifiche apportate al codice hanno permesso un allineamento delle etichette, ciò fa in modo che lo script sia in grado di riconoscere con più libertà famiglie di DGA che non conosce così da fornire un risultato dei test più affidabile. Per completezza ed esigenze di utilizzo, lo script è stato aggiornato in modo che implementi le più recenti librerie di Python2.7.

Risultati

Report binario *full*

Fornendo allo script la rete ed il dataset di test relativi alla configurazione dei domini *full*, si ottengono i seguenti risultati per la classificazione binaria:

- Class report:

	Precision	**Recall**	**F1-Score**	**Support**
White	0.9896	0.9835	0.9866	30001
Black	0.9677	0.9795	0.9735	15106
Micro Average	0.9822	0.9822	0.9822	45107
Macro Average	0.9786	0.9815	0.9801	45107
Weighted Average	0.9823	0.9822	0.9822	45107

Tab. 5: Riporta le performance della classificazione binaria della rete neurale del DII con i domini in configurazione full

- Overall Accuracy: 0.9821757155208726

  White: 0.984

  Black: 0.979

- True Positive: 44303

La classificazione binaria della rete neurale addestrata dal DII performa molto bene, in linea con i risultati ottenuti nel lavoro di ricerca di Duc Tran e associati. Si hanno punteggi che arrivano al 98.9% di precisione di riconoscimento dei domini legittimi e un buon 96.7% di precisione di riconoscimento dei domini malevoli.

Report multiclasse *full*

- Class report:

	Precision	Recall	F1-Score	Support
Necurs	0.9977	0.8251	0.9032	15000
Alexa	0.9896	0.9835	0.9866	30001
Dromedan	0	0	0	6
Vawtrak	0.7500	0.8700	0.8056	100
Micro Average	0.9913	0.9305	0.9599	45107
Macro Average	0.6843	0.6697	0.6738	45107
Weighted Average	0.9916	0.9305	0.9583	45107

Tab. 6: Riporta le performance della classificazione multiclasse della rete neurale del DII con i domini in configurazione full

- Overall Accuracy: 0.9821757155208726

  Alexa: 0.984

  Vawtrak: 0.87

  Necurs: 0.825

  Dromedan: 0

- True Positive: 44303

La classificazione Multiclasse funziona con buone performance. I domini delle famiglie Dromedan e Vawtrak sono numericamente inferiori di più di un ordine di grandezza sia nel test che nel training, questo ne spiega in parte le peggiori performance di riconoscimento. La famiglia di DGA Necurs, quella più numerosa nel nostro set di dati di test, viene riconosciuta con altissima precisione ma più bassa recall: questo significa che la rete opera con grande certezza quando indica un dominio appartenente a questa famiglia ma che non è in grado di indicare con assoluta certezza l'intero numero di domini che la famiglia possiede nel set di dati.

I domini benevoli della famiglia definita Alexa vengono riconosciuti con punteggi superiori al 98%: questo sta a significare che i domini definiti come tali hanno il 98,96% di probabilità di appartenere veramente alla famiglia in questione e che il 98,35% dei domini fatti avere al classificatore di questa famiglia vengono riconosciuti nel modo giusto.

Matrici di confusione *full*

Vengono riportate le matrici di confusione relative al set di dati di test con i domini in configurazione *full*.

	White	**Black**
White	29507	494
Black	310	14796

Tab. 7: Matrice di confusione della classificazione binaria della rete neurale del DII con i domini in configurazione full

La matrice di confusione binaria conferma l'analisi sui parametri fatta nel paragrafo precedente.

Come anticipato, questa rete neurale è addestrata anche su famiglie di DGA non contenute nei dati che abbiamo dato in pasto al classificatore. Per questo la matrice di confusione della classificazione multiclasse non sarà riportata completa, ma ne verranno riportate righe e colonne più importanti.

	Necurs	**Alexa**	**Dromedan**	**Vawtrak**	**Altri**
Necurs	12377	302	0	16	2305
Alexa	29	29507	0	12	452
Dromedan	0	4	0	1	1
Vawtrak	0	4	0	87	9

Tab. 8: Matrice di confusione della classificazione multiclasse della rete neurale del DII con i domini in configurazione full. La colonna Altri rappresenta le altre famiglie di DGA su cui i domini proposti vengono diffusi.

La rete non è addestrata per la famiglia Dromedan che non riconosce e assegna principalmente ad Alexa. Una minore accuratezza di tutte le classi rispetto al modello binario è dovuta alla distribuzione dei domini false-negative sulle 31 classi di addestramento. Viene confermato quanto detto prima riguardo la famiglia Necurs nel paragrafo relativo alla classificazione multiclasse: muovendosi sulla prima colonna della *Tab. 8*, i domini predetti come Necurs sono quasi interamente appartenenti alla famiglia di DGA Necurs, esclusi 29 domini che realmente sono benevoli; dei domini appartenenti alla famiglia, 2607 elementi vengono predetti come appartenenti ad altre.

Report binario *h-t*

Fornendo allo script la rete ed il dataset di test relativi alla configurazione dei domini *h-t*, si ottengono i seguenti risultati:

- Class report:

	Precision	Recall	F1-Score	Support
White	0.9846	0.9841	0.9866	30001
Black	0.9684	0.9694	0.9689	15106
Micro Average	0.9792	0.9792	0.9792	45107
Macro Average	0.9765	0.9767	0.9766	45107
Weighted Average	0.9792	0.9792	0.9792	45107

Tab. 9: Riporta le performance della classificazione binaria della rete neurale del DII con i domini in configurazione h-t

- Overall Accuracy: 0.9791606624249007

  White: 0.984

  Black: 0.969

- True Positive: 44167

La classificazione Binaria dei domini *h-t* funziona con performance buone ma leggermente più basse di quella dei domini *full*. La differenza sui True Positive è di circa 140 elementi. Rimane comunque in linea con i risultati ottenuti nelle ricerche di riferimento.

Report multiclasse *h-t*

Passiamo poi alla classificazione multiclasse:

- Class report:

	Precision	Recall	F1-Score	Support
Necurs	0.9991	0.7968	0.8865	15000
Alexa	0.9846	0.9841	0.9843	30001
Dromedan	0	0	0	6
Vawtrak	0.8824	0.9000	0.8911	100
Micro Average	0.9885	0.9215	0.9538	45107
Macro Average	0.7165	0.6702	0.6905	45107
Weighted Average	0.9891	0.9215	0.9515	45107

Tab. 10: Riporta le performance della classificazione multiclasse della rete neurale del DII con i domini in configurazione h-t

- Overall Accuracy: 0.9214756024563815

  Alexa: 0.984

  Vawtrak: 0.9

  Necurs: 0.797

  Dromedan: 0

- True Positive: 41565

Si ottengono risultati molto simili a quelli della classificazione multiclasse in configurazione *full*.

Si ha un migliore riconoscimento della famiglia Vawtrak, probabilmente dovuto alla configurazione che rende la famiglia più vulnerabile. Si hanno performance di riconoscimento della famiglia Necurs però lievemente inferiori. Essendo la famiglia Necurs più consistente nel supporto del set di test, le performance multiclasse totali sono leggermente inferiori. Riguardo la famiglia Necurs vale il discorso fatto nella classificazione multiclasse dei domini in configurazione *full*: alta precisione indica maggiore sicurezza sull'identificazione del dominio, più bassa recall indica che non tutti i domini della famiglia vengono riconosciuti. I domini benevoli della famiglia Alexa vengono riconosciuti con ottime performance.

Matrici di confusione *h-t*

Riportiamo anche le matrici di confusione relative a questa prova:

	White	**Black**
White	29523	478
Black	462	14644

Tab. 11: Matrice di confusione della classificazione binaria della rete neurale del DII con i domini in configurazione h-t

Anche in questo caso la matrice di confusione binaria conferma l'analisi sui parametri fatta nel paragrafo precedente.

Per la classificazione multiclasse:

	Necurs	**Alexa**	**Dromedan**	**Vawtrak**	**Altri**
Necurs	11952	451	0	1	2596
Alexa	10	29523	0	9	459
Dromedan	0	3	0	2	1
Vawtrak	0	8	0	90	2

Tab. 12: Matrice di confusione della classificazione multiclasse della rete neurale del DII con i domini in configurazione h-t. La colonna Altri rappresenta le altre famiglie di DGA su cui i domini proposti vengono diffusi.

Anche qui i false-negative della famiglia Necurs vengono distribuiti tra le varie classi di DGA, come indicato da una recall di circa l'80%. Come anche i domini benevoli della famiglia Alexa, i domini che vengono indicati come Necurs hanno alta probabilità di appartenere realmente a quella famiglia.

Per non ripeterci eccessivamente, i risultati rimangono comunque molto simili a quelli ottenuti con i domini in configurazione *full*, quindi ne valgono le stesse considerazioni con le opportune distinzioni.

3.2 Esperimento 1

Illustrazione e preparazione

Il secondo esperimento viene effettuato in maniera leggermente diversa. Invece che lavorare su un classificatore già addestrato e fornito dal Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche, si è deciso di lavorare su di un nuovo classificatore, addestrato su dati differenti.

Il classificatore viene infatti generato con lo script descritto nel lavoro di ricerca di Duc Tran et al. (16) e messo a disposizione dagli autori stessi su di un set di addestramento preparato ad hoc. Lo script, come già riferito, viene aggiornato alle ultime versione delle librerie da cui dipende e viene reso compatibile con il lavoro, tutto ciò cercando sempre di non corrompere il lavoro fatto da chi lo ha messo a disposizione.

Il nuovo set di training è composto dalla somma di tutti i domini unici contenuti nei feed del Bambenek Consulting generati nei 29 giorni in esame[15]. A questa mole di dati vengono rimossi i domini presenti nel dataset di test, cioè il dataset ottenuto dall'intersezione dei domini che compaiono nei log del DNS della rete GARR con i domini che compaiono nei feed del Bambenek Consulting. I domini rimossi vengono poi sostituiti con domini benevoli presi in modo casuale dal feed del Majestic Million non presenti nel dataset di test, in modo da fornire al dataset abbastanza domini di classe legittima per l'addestramento.

Per esigenze di compatibilità con lo script di addestramento che si è voluto mantenere fedele all'orginale, la famiglia Dromedan viene rimossa dal set. Questo perché lo script non tollera efficientemente le classi con pochi elementi e questo può portare ad una interruzione casuale dell'esecuzione. Purtroppo, la durata importante dell'esecuzione non permette un'approccio del tipo "tento finché non riesco", in quanto probabilmente ci sarebbero voluti mesi.

[15] Che ricordiamo essere i giorni: 16, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 Settembre 2019; 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 17, 25, 26 Ottobre 2019.

Il classificatore viene fatto addestrare e testare solamente su domini in configurazione *full*, su di una macchina con le più recenti ma modeste soluzioni tecnologiche del mercato consumer. Il tempo di addestramento su un notebook con processore Intel i7-9750H, scheda video nVidia GeForce RTX 2060 e sistema operativo Ubuntu 19.04 è di circa 46 ore. L'addestramento viene fatto adeguando le librerie di Python richieste dallo script di addestramento all'utilizzo dei CudaCore presenti nella scheda grafica in unione alla capacità computazionale del processore, in modo da sfruttare le più recenti soluzioni di calcolo in parallelo e accelerare notevolmente l'esecuzione.

Come nell'esperimento precedente, il classificatore ora addestrato viene testato sul set di dati ottenuto nella fase di analisi e raccolta. Lo script utilizzato per il test delle performance è lo stesso, con le stesse modifiche apportate nell'esperimento precedente. Così facendo, le uniche variabili a cambiare sono la rete neurale e il set di addestramento su cui è generata.

Risultati

Report binario

Fornendo allo script la rete ed il dataset di test opportuno, si ottengono i seguenti risultati per la classificazione binaria:

- Class report:

	Precision	Recall	F1-Score	Support
White	0.9994	0.7769	0.8742	30001
Black	0.6928	0.9991	0.8182	15106
Micro Average	0.8513	0.8513	0.8513	45107
Macro Average	0.8461	0.8880	0.8462	45107
Weighted Average	0.8967	0.8513	0.8555	45107

Tab. 13: Riporta le performance di classificazione binaria della rete neurale generata per l'esperimento 1.

- Overall Accuracy: 0.8513091094508612

  White: 0.777

  Black: 0.999

- True Positive: 38400

La classificazione binaria evidenzia alcuni problemi rispetto alle reti dell'esperimento precedente. Si ha infatti poca precisione dei domini DGA ed i true-positive sono sensibilmente inferiori. Questo porta a dei valori molto più modesti e imprecisi. È logico e giusto pensare che ad una precisione più alta di una famiglia in classificazione binaria, corrisponda un recall più alta della famiglia opposta. Allo stesso modo questo avviene per una precisione più bassa di una e una recall più bassa dell'altra. In questa fase dell'esperimento si ha un esempio del fenomeno appena descritto se consideriamo le classi dei benevoli, *White*, e quelle dei malevoli, *Black*.

Report multiclasse

Con le stesse modalità si ottengono i dati relativi alla classificazione multiclasse:

- Class report:

	Precision	Recall	F1-Score	Support
Necurs	0	0	0	15000
Alexa	0.9994	0.7769	0.8742	30001
Dromedan	0	0	0	6
Vawtrak	0	0	0	100
Micro Average	0.9994	0.5167	0.6812	45107
Macro Average	0.2499	0.1942	0.2186	45107
Weighted Average	0.6647	0.5167	0.5814	45107

Tab. 14: Riporta le performance della classificazione multiclasse della rete neurale generata per l'esperimento 1.

- Overall Accuracy: 0.5167047243221673

  Alexa: 0.777

  Vawtrak: 0

  Necurs: 0

  Dromedan: 0

- True Positive: 23307

Dai dati appena forniti si evince che la rete performa in modo pessimo, con performance nulle, in tutte le classi dei domini malevoli presentate in fase di test. Si ha un valore di F1-Score decente per quanto riguarda i domini benevoli, ma questo non rende comunque la rete affidabile. I valori non così bassi della classificazione binaria sono dovuti a discrete performance di riconoscimento dei domini che appartengono alla famiglia Alexa. Un'accuratezza totale con valore simile al 50% fa sì che le performance ottenute dalla classificazione multiclasse siano molto simili a quelle ottenibili con il lancio di una moneta.

Matrici di confusione

Di seguito le matrici di confusione:

	White	Black
White	23307	6694
Black	13	15093

Tab. 15: Matrice di confusione della classificazione binaria della rete neurale dell'esperimento 1.

La matrice di confusione binaria non contiene molte informazioni non fornite dai parametri riportati precedentemente. Si evidenziano l'alta precisione nel determinare i domini benevoli e l'alta capacità di indicare correttamente grandissima parte dei domini malevoli.

Seguiamo con la matrice di confusione multiclasse:

	Geodo	Necurs	Alexa	Dromedan	Vawtrak	Altri
Necurs	14991	0	9	0	0	0
Alexa	6694	0	23307	0	0	0
Dromedan	6	0	0	0	0	0
Vawtrak	96	0	4	0	0	0

Tab. 16: Matrice di confusione della classificazione multiclasse della rete neurale dell'esperimento 1.

Una breve analisi chiarisce quasi a colpo d'occhio il problema che ha scaturito i bassi punteggi dei parametri ottenuti prima. L'addestramento della rete neurale ha sbilanciato in modo eccessivo le predizioni e la famiglia Geodo ha attratto la quasi totalità delle famiglie di DGA malevole, attraendo anche parte dei domini benevoli. Vengono attratti anche i pochi domini appartenenti a Dromedan nonostante questi fossero stati rimossi dal set di addestramento. Si ritiene che questo sia dovuto ad una binarizzazione della classificazione multiclasse: i domini vengono visibilmente fatti confluire solo su una coppia di famiglie.

Il problema, tra tante cause più complesse, può risiedere in uno sbilanciamento dei numeri dei domini appartenenti alle classi del set di addestramento. Nonostante la presenza dell'algoritmo del Multiclass-Imbalance, introdotto dalla ricerca su cui si basa lo script utilizzato per l'addestramento (16), che avrebbe

dovuto permettere un addestramento più omogeneo, si nota che un eccessivo sbilanciamento, come quello mostrato in questo caso, risulta difficile da mitigare.

3.3 Esperimento 2

Illustrazione e preparazione

L'esperimento 2 ricalca i passi fatti nell'esperimento 1 cercando di lavorare su un set di dati di addestramento più compatto e più omogeneo.

Come nell'esperimento 1 il classificatore viene addestrato con lo script originale dell'elaborato di ricerca di Duc Tran e colleghi. Il set di addestramento dedicato a questo esperimento che viene fornito allo script è simile a quello dell'esperimento precedente.

Vengono raccolti tutti i domini malevoli unici che compaiono nei feed del Bambenek Consulting nei 29 giorni presi in esame. A questi vengono rimossi i domini malevoli presenti nei log DNS della rete GARR in modo da far lavorare la rete su domini malevoli ad essa completamente sconosciuti. Con lo stesso principio, i domini malevoli rimossi vengono sostituiti con un egual numero di domini benevoli presi dal Majestic Million non presenti nei log DNS. A questo punto però, per ottenere omogeneità e bilanciamento, vengono rimosse le famiglie che contano meno di 200 elementi, mentre le famiglie più numerose vengono limitate a 20'000 domini unici. Sia l'inserimento dei domini presi dal MM, sia la rimozione dei domini eccedenti alle famiglie DGA più numerose viene fatto casualmente, estraendo domini in modo casuale con un piccolo script preparato per lo scopo.

Il dataset di addestramento risulta più compatto del precedente e la fase di addestramento con lo script dedicato, fatto girare sulla stessa macchina su cui si è preparato l'esperimento precedente, richiede un tempo di circa 18 ore. Anche in questo caso il framework richiamato dallo script che esegue l'addestramento è configurato in modo da utilizzare la capacità computazionale dei CudaCore presenti nella scheda grafica.

Terminato l'addestramento si esegue la fase di test, fornendo allo script utilizzato anche nei due esperimenti precedenti i dataset ricavati dalla fase di raccolta dati in configurazione *full*. Viene rimosso un solo dominio in quanto generava errori in fase di esecuzione dello strumento: visti i numeri dei domini riportati, la rimozione

di un elemento non dovrebbe implicare corruzione dei parametri delle performance rispetto agli esperimenti precedenti.

Risultati

Report binario

Fornendo allo script la rete ed il dataset di test relativi alla configurazione dei domini *full*, si ottengono i seguenti risultati della classificazione binaria:

- Class report:

	Precision	Recall	F1-Score	Support
White	0.9971	0.8797	0.9347	30000
Black	0.8063	0.9949	0.8907	15106
Micro Average	0.9183	0.9183	0.9183	45107
Macro Average	0.9017	0.9373	0.9127	45107
Weighted Average	0.9332	0.9183	0.9200	45107

Tab. 17: Riporta le performance della classificazione binaria della rete neurale generata per l'esperimento 2.

- Overall Accuracy: 0.9182592116348157

  White: 0.88

  Black: 0.995

- True Positive: 41419

Sia i punteggi della classificazione binaria, sia i true-positive ottenuti sono inferiori ma più in linea con quelli dell'esperimento 0 rispetto ai risultati ottenuti nell'esperimento 1. I dati si discostano in basso di qualche punto percentuale rispetto ai dati del lavoro di ricerca di riferimento. Le performance della classificazione binaria possono comunque essere definite discrete. Una precisione molto alta nel classificare i domini *White* corrisponde ad una recall molto alta dei domini *Black*, una precisione più bassa dei domini *Black* corrisponde ad una recall più bassa dei domini *White*. Tutto ciò comunque con questi 4 valori mediamente maggiori agli stessi dell'esperimento precedente.

Report multiclasse

- Class report:

	Precision	Recall	F1-Score	Support
Necurs	0.9926	0.7721	0.8686	15000
Alexa	0.9926	0.9371	0.9640	30000
Dromedan	0	0	0	6
Vawtrak	0	0	0	100
Micro Average	0.9926	0.8800	0.9329	45106
Macro Average	0.4963	0.4273	0.4581	45106
Weighted Average	0.9902	0.8800	0.9300	45106

Tab. 18: Riporta le performance della classificazione multiclasse della rete neurale generata per l'esperimento 2.

- Overall Accuracy: 0.8799937924001241

  Alexa: 0.937

  Vawtrak: 0.772

  Necurs: 0

  Dromedan: 0

- True Positive: 39693

Anche i punteggi ottenuti dalla rete nella classificazione multiclasse sono più in linea con i corrispettivi ottenuti nell'esperimento 0 che nell'esperimento 1. Le famiglie Dromedan e Vawtrak non vengono riconosciute, come nell'esperimento 1, ma si hanno buoni punteggi per quanto riguarda la famiglia malevola Necurs e i domini benevoli. Vi è comunque uno scarto di accuratezza del 12%, alto rispetto a quello del paper di riferimento, anche se in linea con i valori ottenibili da altre soluzioni di reti neurali o alberi decisionali (18). Si conferma il fatto che i valori identici di precisione delle classi Necurs ed Alexa non siano un errore di trascrizione. Questi valori, 99,2% di precisione, rendono ottima la soluzione in quanto difficilmente la rete classificherà un elemento delle due famiglie in modo errato.

Matrici di confusione

Di seguito le matrici di confusione relative all'esperimento:

	White	Black
White	26390	3610
Black	77	15023

Tab. 19: Matrice di confusione della classificazione binaria della rete neurale dell'esperimento 2.

La matrice di confusione binaria non contiene molte informazioni non fornite dai parametri riportati precedentemente, si può notare anche qui il chiasmo dei dati relativi a precisione e recall.

Segue la matrice di confusione multiclasse:

	Necurs	Alexa	Dromedan	Vawtrak	Altri
Necurs	11581	106	0	0	3313
Alexa	86	28112	0	0	1802
Dromedan	0	100	0	0	1
Vawtrak	0	5	0	87	9

Tab. 20: Matrice di confusione della classificazione multiclasse della rete neurale dell'esperimento 2.

Nella classificazione multiclasse non si ha una eccessiva dispersione dei domini benevoli, mentre la maggior parte dei domini della famiglia Necurs ai quali viene predetta la famiglia errata vengono attratti da famiglie di DGA. Entrambe le famiglie Vawtrak e Dromedan vengono predette soprattutto come domini benevoli, che si ricorda essere la classe più numerosa. Si può notare ad occhio anche qui l'alta precisione nel classificare i domini: di tutti i classificati Necurs solo 86 sono falsi positivi, di tutti i classificati legittimi solo 211 sono falsi positivi.

3.4 Conclusioni sugli esperimenti

Se si considerano le finalità pratiche del lavoro, ovvero quelle di implementare una soluzione che gli amministratori dei server DNS possano piazzare al fianco di pratiche come quella del blacklisting, probabilmente la scelta dovrebbe ricadere su soluzioni simili alla rete neurale utilizzata nell'esperimento 0.

La rete infatti ottiene ottime performance in ogni parametro rilevante, mediamente maggiori di quelle ottenute dalle altre. Sarebbe in grado di fornire un bassissimo numero di falsi positivi sui domini legittimi, sbagliando le predizioni su solo l'1% di questi. L'errore sui domini illegittimi è anch'esso sufficientemente basso.

Dall'altro lato però la rete neurale fornita dal Dipartimento di Ingegneria dell'Informazione ha richiesto mesi di preparazione ed il lavoro di diversi tecnici, tra le pratiche di ingegneria inversa e raccolta di informazioni. Questo richiederebbe costante attenzione all'aggiornamento dei dati nel caso in cui avvenga la scoperta di nuove famiglie di DGA, che richiederebbero ulteriori processi di ingegneria inversa e analisi altrettanto complesse.

La rete neurale dell'esperimento 1 d'altro canto ottiene risultati pessimi in classificazione multiclasse, ma accettabili nella classificazione binaria, soprattutto se si considera la capacità di riconoscere i domini benevoli con ottima precisione. La preparazione del suo addestramento però richiede molta meno supervisione, essendo il suo dataset di training una semplice somma dei domini contenuti in una attendibile risorsa online.

Visti i pochi controlli effettuati sui dati, si potrebbe pensare di implementare questa soluzione con successivi riaddestramenti periodici che sarebbero semplici da predisporre e resistenti ad un eventuale cambiamento del feed del Bambenek Consulting.

Ciò che però sconsiglia l'utilizzo di questa soluzione, oltre alle performance inferiori, è il netto fallimento dell'algoritmo della gestione dello sbilanciamento multiclasse. Il fatto che in questo caso non sia concettualmente funzionante una base fondamentale su cui si struttura l'idea della rete potrebbe indicare la presenza di altre limitazioni nascoste.

La rete neurale dell'ultimo esperimento fa da soluzione mediana. Le performance sono comunque ottime, abbiamo visto eccellente precisione nell'individuare le famiglie che riconosce, anche se il numero di famiglie riconosciute è inferiore. L'addestramento risulta semplice comparato a quello effettuato sulla rete dell'esperimento 0 e poco più complesso di quello fatto con la rete dell'esperimento 1. Nell'ultimo caso si tratta soltanto di inserire qualche algoritmo di controllo sul numero dei domini contenuti in ogni famiglia.
Le limitazioni scelte in questo lavoro inoltre rendono funzionante l'algoritmo del Multiclass-Imbalance proposto nel lavoro di Duc Tran e con buona probabilità ci si porta nel miglior caso d'uso di una rete LSTM.MI. L'aggiornamento della rete alla scoperta di una nuova famiglia di DGA risulterebbe comunque abbastanza semplice e potrebbe avvenire in modo completamente automatico, se gli algoritmi di controllo predisposti sono realizzati con le dovute attenzioni.

In conclusione, gli esperimenti si rivelano un atteso successo e si piazzano in linea con i risultati ottenuti in altre ricerche all'avanguardia della tecnica (9) (18) (25).

4. Commenti e sviluppi futuri

Abbiamo visto quanto sia importante per la comunità arginare il fenomeno dei malware. Una buona educazione e l'utilizzo di strumenti di protezione sono di solito sufficienti ad assicurare una discreta protezione all'utente medio. Nelle grandi reti però, basta un solo dispositivo vulnerabile a rendere l'intera rete potenzialmente esposta: se la catena ha un anello debole, l'intera catena non è resistente.

Inoltre, per un Internet più equo e più giusto, non è auspicabile che solo chi può permettersi soluzioni all'avanguardia od una adeguata istruzione sia schermato dalla minaccia dei malware. È decisamente più giusto, infatti, che sia la stessa rete a fornire una protezione efficace, in modo che non ci siano nodi più deboli degli altri e in modo che l'intera rete ne giovi. È per questo che l'implementazione di una soluzione di protezione a monte, come quella presentata in questo lavoro, può rivelarsi utile e necessaria.

Le reti neurali studiate in questo lavoro sono state addestrate su dati controllati e su casi sufficientemente particolari. Risulta, infatti, che nei log DNS della rete GARR analizzati precedentemente non compare un numero eccessivamente grande di famiglie di DGA. Le 3 famiglie di algoritmi che compaiono, Necurs, Vawtrak e Dromedan, sono solo una piccola parte di quelle che il classificatore è stato addestrato a riconoscere. È quindi difficile prevedere il comportamento del classificatore nel caso in cui venisse a contatto con altre famiglie. Ci si immagina comunque, viste le ottime performance ottenute dai classificatori nel riconoscere la famiglia Necurs, che nel caso in cui le famiglie contengano un numero sufficientemente alto di domini nel set di addestramento delle reti, gli strumenti riportati siano in grado di ottenere buoni punteggi di riconoscimento.

Un così basso numero di famiglie di domini malevoli presenti nei nostri log può essere dovuto a molti fattori. Innanzi tutto, ci si augura che la percentuale di domini malevoli contattati sia così bassa per tutti i server DNS del globo. Se così non fosse, si ricorda al lettore che la rete su cui sono stati effettuati i test è una rete prettamente universitaria. Questo porta a presupporre che gli utenti della rete siano più formati in competenze informatiche della media della popolazione e che quindi siano più consapevoli delle scelte che fanno quando decidono di aprire un allegato potenzialmente pericoloso, quando navigano su siti poco affidabili, quando scaricano delle risorse. Come abbiamo già ricordato, una discreta formazione dell'utente è uno dei primi ostacoli che la proliferazione dei malware deve superare.

In aggiunta, da analisi meno precise e meno rilevanti si è scoperto che la totalità del traffico malevolo registrato nel periodo scelto per questo lavoro dagli strumenti in nostro possesso è generato da una decina di macchine infette. Questo numero è molto piccolo rapportato alle dimensioni della rete GARR intera. Il dato è comunque confortante, ma viene mitigato dal fatto che, nonostante le macchine infette siano oggettivamente poche, il traffico che generano è relativamente elevato. Soprattutto le macchine infette da malware che usano DGA della famiglia Necurs possono arrivare ad effettuare 2600 query DNS malevole ogni ora.

Le reti neurali presentate in questo lavoro possono rappresentare un ottimo strumento di arginamento delle minacce. Abbiamo visto che in un campione di nomi di dominio unici, l'errore generato nelle soluzioni migliori è nell'ordine dell'1 o 2%. Ricordiamo inoltre che soprattutto le soluzioni completamente nuove presentate in questo lavoro, quelle degli ultimi due esperimenti, sono state testate su nomi di dominio a loro completamente sconosciuti. Come abbiamo indicato infatti, il set di training ed il set di test non condividono nessun nome di dominio. In un utilizzo più pratico e realistico, la maggior parte delle richieste fornite al DNS, sia relative a domini legittimi che a domini più pericolosi e dannosi, sono conoscibili. Questo significa che gran parte dei domini che vengono presentati al DNS hanno probabilità praticamente piena di essere riconosciuti e classificati correttamente. Ci si occuperà infatti di addestrare la rete su traffico reale e si presuppone che essa sia in grado di riconoscere senza problemi elementi che già conosce.

Una piccola parte dei nomi di dominio legittimi, magari quelli meno diffusi o più rari, sono quelli che finiscono nella zona di predizione della rete ed è appunto tra questi che si registrano le ottime prestazioni che le abbiamo visto ottenere nei capitoli precedenti. Secondo questo logico ragionamento c'è quindi un'altissima probabilità che l'inserimento su di un DNS di un sistema di blacklisting basato sulle reti neurali qui presentate non infici quasi minimamente sulla navigazione degli utenti. Nel caso peggiore l'utente sarà impossibilitato a navigare sul 2% di una piccola parte dei domini legittimi sui quali è intenzionato a recarsi. A tale perdita, però, corrisponde un blocco mediamente efficiente da parte del DNS della gran parte dei domini malevoli che l'eventuale macchina infettata dell'utente richiederà senza supervisione.

Nasce quindi il problema di dover scegliere quali perdite possano compensare quali guadagni. Non sta a noi scegliere, ma potrebbero essere inficiate da questo sistema le pagine web di piccole realtà internaute, con conseguente censura involontaria del loro lavoro legittimo. Per questo si crede sia molto importante approfondire lo studio di questi sistemi e raggiungere performance degli strumenti utilizzati ancora più alte e sicure.

Nel caso in cui si scegliesse poi di implementare una soluzione efficace ma non perfetta, starà al tecnico implementatore prevedere dei meccanismi in grado di disabilitare la censura per i domini legittimi o di unire il sistema a dei controlli più specifici, come ad esempio controlli di contenuto o delle informazioni di registrazione del dominio bloccato. Una possibile implementazione delle soluzioni allo stato attuale potrebbe infatti segnalare all'amministratore della richiesta il fatto che possa essere potenzialmente nociva, così quindi che almeno per quanto riguarda il browsing o il download da parte di un utente, sia l'utente stesso a poter scegliere di continuare o meno. Esattamente come avviene oggi giorno quando si fa richiesta di un sito https con la chiave di cifratura non valida: il sistema chiede conferma all'utente per continuare la navigazione.

Resta quindi ancora un po' di strada da fare affinché questi sistemi siano efficienti e completamente autonomi. Parte del lavoro, nel caso in cui si decidesse di utilizzare una rete LSTM.MI come è stato fatto in questa ricerca risiede sicuramente nel perfezionare l'algoritmo del Multiclass-Imbalance. Abbiamo visto che non in tutti i casi riesce a compensare lo sbilanciamento del set di addestramento. Set di addestramento che, come abbiamo ripetuto più volte, ha un suo grado di casualità e soffre di sbilanciamento.
Uno studio più approfondito del numero ideale di elementi che impedisce lo sbilanciamento della rete può rivelarsi utile per aumentarne le performance.

Per ottenere risultati precisi e ottimali, facendo ricerca sulle potenzialità dell'intelligenza artificiale per l'intercettazione del traffico verso server di comando e controllo, c'è la necessità di testare le funzionalità degli strumenti su reti con più traffico e richieste. Testare gli strumenti in condizioni più generiche può rivelarsi utile per ottenere strumenti più performanti in situazioni più specifiche, come quella in cui ci si trova in questo caso di studio.

In conclusione, gli strumenti presentati, anche se incompleti, possono ritenersi pronti ad una eventuale implementazione. L'autore però consiglia maggiori ricerche e approfondimenti sugli argomenti in questione prima dell'implementazione definitiva, in modo da rendere più semplice ed efficiente l'intero sistema.

5. Appendice

5.1 Esempio di log DNS della rete GARR

Riportiamo qualche riga di esempio del file di log del DNS della rete GARR del 23 Settembre 2019:

```
2019-09-23 20:05:06.742030; 2a4aa87272aa37a556a3fdf621fbef5c6ff68836;
193.206.141.38; udp; IN; osgbcfcdst.unitn.it.; A; NXDOMAIN; 0; 1

2019-09-23 20:05:06.744906; 97d94c6c9e4c3f3c87ed61add2c5a1f24fc6d408;
193.206.141.38; udp; IN; ssl.gstatic.com.; A; 216.58.198.3; 296; 1

2019-09-23 20:05:06.749408; 149ae9f1cdf826f2a1065fdb3be46ef9abb8ea04;
2001:760:ffff:ffff::aa; udp; IN; dns1.iue.it.; A; 192.167.90.1; 3600; 1

2019-09-23 20:05:06.749686; dcc41ba6af3add455ae2d9db58b8df7d70b5cba8;
193.206.141.38; udp; IN; ntp.inrim.it.; A; 193.204.114.105; 300; 1

2019-09-23 20:05:06.750109; 321386512c79921df2feccfe27bdea1fc88c3da1;
2001:760:ffff:ffff::aa; udp; IN; tus001.unitus.it.; A;193.205.144.2; 1200; 1

2019-09-23 20:05:06.765094; a930269bbe25e2a893b22850f9657cfd34d4e190;
193.206.141.38; udp; IN; alfa.infm.it.; A; SERVFAIL; 0; 1
```

Per rendere il testo più leggibile per il lettore, tra le varie sottostringhe degli elementi sono stati aggiunti degli spazi non esistenti nel file originale. Come è facilmente rilevabile, le varie informazioni sono separate da dei caratteri “;”. Le varie informazioni contenute, in ordine di come compaiono in ogni riga sono: la data, il codice hash della macchina che ha effettuato la richiesta, l’indirizzo IP, indirizzo IP interno al DNS che riceve e

processa la query, tipologia di protocollo di trasmissione (udp o tcp), tipologia di connessione (ingresso, uscita, interna), dominio da risolvere, tipologia di risposta (indirizzo, cname, testo o altre), indirizzo IP del dominio richiesto, porta del dominio richiesto.

Le informazioni importanti per questo lavoro in realtà si riducono a ciò che è contenuto nella stringa del nome di dominio. Il software che si occupa dell'estrazione di questi dati da tutti i log interessati, scorre le righe del log ed all'interno di ogni riga cerca l'elemento che si trova in sesta posizione. Come già detto, le posizioni sono delimitate da un carattere di fine stringa, in questo caso il "punto e virgola".

Quando si è affermato che le richieste DNS malevole relative alla famiglia di DGA Vawtrak provenivano tutte dalla stessa macchina o sottorete, si è analizzato e confrontato l'hash dell'indirizzo IP del richiedente. Per affermare che qualcuna, tra le query DNS malevole, fosse riuscita a contattare il server di comando e controllo abbiamo analizzato la risposta fornita dal DNS: se questa non è NXDOMAIN le probabilità che il dominio risposto sia quello del server C&C sono altissime.

5.2 Esempio di feed DGA di Bambenek Consulting

Riportiamo qualche riga del feed DGA del Bambenek Consulting del giorno 6 Ottobre 2019:

```
clgqfbpnoltyjol.com,Domain used by Cryptolocker - Flashback DGA for 04 Oct 2019,2019-10-04,http://osint.bambenekconsulting.com/manual/cl.txt

lepqlimlanmrnkm.net,Domain used by Cryptolocker - Flashback DGA for 04 Oct 2019,2019-10-04,http://osint.bambenekconsulting.com/manual/cl.txt

...

xxqssstkmpev.pw,Domain used by tinba DGA for 06 Oct 2019,2019-10-06,http://osint.bambenekconsulting.com/manual/tinba.txt

xxqtltgeooeu.pw,Domain used by tinba DGA for 06 Oct 2019,2019-10-06,http://osint.bambenekconsulting.com/manual/tinba.txt

...

nndpcrnjccevjjlpgavbt.us,Domain used by necurs (domains valid ~4 days),2019-10-06,http://osint.bambenekconsulting.com/manual/necurs.txt

nndypxixrcvtptxfcxjyq.sh,Domain used by necurs (domains valid ~4 days),2019-10-06,http://osint.bambenekconsulting.com/manual/necurs.txt

...
```

Come si è cercato di mostrare inserendo spaziature e puntini, i possibili domini assumibili da un server di comando e controllo, forniti dal feed DGA, vengono disposti ordinati per famiglia. Significa quindi che tutti i domini di una famiglia vengono inseriti l'uno dopo l'altro; quando i possibili domini generabili da una famiglia terminano, si passa alla successiva.

Nell'esempio si sono voluti riportare diversi domini di diverse famiglie. Abbiamo Cryptolocker, Tinba e Necurs. Per ogni dominio riportato, il feed fornisce informazioni riguardo l'appartenenza alla famiglia, la data in cui il dominio è valido e un link al manuale fornito dal Bambenek Consulting. Tali informazioni sono separate da caratteri "virgola".

Il software che si occupa di estrarre informazioni dai feed bambenek scorre tutte le righe del file di testo e, per ogni riga, estrae le prime due stringhe. La prima viene memorizzata subito e riporta il nome di dominio, la seconda riga viene di nuovo sezionata per ricercare la famiglia del dominio. Per farlo, il software memorizza i caratteri subito seguenti a "Domain used by" e le salva su una stringa. Un piccolo rimaneggiamento di questa stringa riporterà poi il nome di dominio.

Va reso noto che il link al manuale di ogni famiglia è autogenerato e non sempre digitando quell'indirizzo su browser è possibile trovare informazioni. Per quanto riguarda le famiglie di DGA che compaiono più frequentemente in questo lavoro, purtroppo non è stato possibile accedere al manuale fornito dal Bambenek Consulting di nessuna di queste.

5.3 Esempio di feed Majestic Million

Riportiamo qualche riga del feed del Majestic Million, aggiornato a Ottobre 2019:

1	1	google.com	com	494267	2875853	google.com	com	1	1	494059	2873436
2	2	facebook.com	com	487449	2978572	facebook.com	com	2	2	489262	2997596
3	3	youtube.com	com	440988	2393141	youtube.com	com	3	3	440855	2388459
4	4	twitter.com	com	438944	2438417	twitter.com	com	4	4	438769	2436582
5	5	linkedin.com	com	335241	1490484	linkedin.com	com	5	5	335094	1489273
6	6	instagram.com	com	334368	1630536	instagram.com	com	6	6	334108	1627183
7	7	microsoft.com	com	315742	1226947	microsoft.com	com	7	7	316005	1233733
8	8	apple.com	com	292879	1054125	apple.com	com	8	8	292968	1052734
9	1	wikipedia.org	org	290976	1157355	wikipedia.org	org	9	1	291084	1157741
10	9	plus.google.com	com	277306	1263831	plus.google.com	com	10	9	277642	1265213

Il file è fornito in formato *.csv*, quindi i dati vengono forniti su tabella con righe e colonne.

La riga principale (omessa per motivi di leggibilità) riporta i nomi delle colonne sottostanti. Da queste se ne estrae che riportano, in ordine: il ranking del dominio nome di dominio, il ranking del dominio tra quelli con quel dominio di primo livello (.org, .com, etc.), il numero di sottoreti a cui vengono riferite, il numero di indirizzi IP a cui vengono riferiti, il dominio IDN a cui si riferiscono e nelle colonne successive vengono riferiti i valori del periodo precedente.

Del file abbiamo deciso di riportare i primi 10 domini, facilmente riconoscibili. Nei domini riportati non avviene una variazione tra il nome di dominio ed il dominio IDN.

Il software che si occupa di estrarre i dati dal feed, utilizza una libreria apposita per l'apertura dei csv e poi scorre tutte le righe e salva su una struttura dati apposita le stringhe contenute nella terza colonna, le stringhe dei nomi di dominio.

5.4 Esempio di realizzazione di uno script

Introduzione

Di tutti gli script e di tutto il loro codice sorgente utilizzato per questo lavoro si è deciso di riportare due esempi. Il primo è costituito dal codice sorgente della funzione utilizzata per estrarre dati dal Bambenek Feed; fornisce quindi un esempio di raccolta dei dati relativa a tutti i feed testuali che vengono utilizzati. Il secondo esempio è costituito dal codice sorgente che viene utilizzato per effettuare il confronto di intersezione tra i vari feed.

Non vengono riportati esempi del codice legato alle reti neurali, né lo script di addestramento, né lo script di test delle performance, in quanto sono lavori non prodotti espressamente per questo lavoro, ma risorse gentilmente offerte, controllate e verificate, dal Dipartimento di Ingegneria dell'Informazione.

Non è inoltre possibile allegare tutto il codice scritto a questo elaborato in quanto conterrebbe moltissime informazioni, alcune molto banali, e sarebbe leggibile solamente ad un pubblico strettamente tecnico. Inoltre, del codice che riporteremo, non verranno contestualizzati i riferimenti tecnici alle variabili o alle funzioni che presenta.
Ci si augura che il riportarne alcuni estratti più significativi sia sufficiente a farne comprendere da parte di un tecnico la bontà ed il funzionamento.

Primo esempio

Di seguito un esempio di codice sorgente in linguaggio Python3.7:

```
def caricaBf(filepath_bf):
    """Carica su di un dizionario i domini del BF

    Parameters
    ---------
    filepath_mm : str
        Stringa contenente il filepath del .csv scaricato dal Majestic Million

    Returns
    -------
    bf_list : dict
        Dizionario con:
            key : str
                dominio del bf
            value : obj
                oggetto contenente informazioni sul dominio
    """

    with lzma.open(filepath_bf, mode='rt') as xzfile:
        line = xzfile.readline()
        for x in range(14):
            line = xzfile.readline()
        bf_list = {}
        while line:
            bf_list.update({line.split(',')[0] : BambenekFeed(domain=line.split(',')[0],
                                                              family=familyFromDescr(descr =
line.split(',')[1])
                                                              )})
            line = xzfile.readline()
    return bf_list
```

Il codice è ben commentato ed indica, nel primo virgolettato, su quali parametri agisce la funzione. In particolare, la funzione utilizza il percorso del file del feed Bambenek Consulting per aprirlo, decomprimerlo attraverso la libreria *lzma* ed iniziare a leggerlo. Vengono quindi saltate le prime 14 righe di intestazione de file, dopodiché si inizia la vera e propria estrazione delle informazioni. Viene estratta la prima stringa di ogni riga per valorizzare il dominio e viene estratta la successiva che viene passata ad una funzione che ne estrae la famiglia. Il tutto viene caricato su di un dizionario che ha come chiave il dominio del Bambenek feed e come

valore un oggetto che funge da struttura dati per contenere alcune informazioni sul dominio, come ad esempio la famiglia. Quando il codice ha terminato l'operazione di lettura su di un file, restituisce al codice che ha chiamato la funzione il dizionario contenente i nomi di dominio e le loro famiglie. Come già annunciato, il codice utilizzato per estrarre informazioni dal Majestic Million funziona allo stesso modo. L'unica differenza risiede nel fatto che mentre il codice riportato scorre le linee di un file di testo, il codice per l'estrazione dal Majestic Million scorre con le librerie opportune una colonna di un file *.csv* di calcolo elettronico.

Secondo esempio

Di seguito un esempio di codice sorgente in formato Python3.7:

```
def confrontaUnFile(filepath_log, filepath_out, mm_list, bf_list, analyzed_traffic):
    """Confronta un'ora del log del Garr con i domini già in mm_list, bf_list e analyzed_traf
fic

    Scorre il file compresso del log alla ricerca di domini in mm_list e bf_list.
    Se trova dei domini sia nel log che in mm_list che non sono in analyzed_traffic
    aggiorna analyzed_traffic e scrive i nuovi domini nel file al filepath_out.
    Se trova dei domini sia nel log che in bf_list che non sono in analyzed_traffic
    aggiorna analyzed_traffic e scrive i nuovi domini nel file al filepath_out.

    Parameters
    ---------
    filepath_log : str
        Filepath completo log di un'ora del garr (formato: ../pdns_yyyy_mm_dd_hh.log.xz)
    filepath_out : str
        Stringa contenente il filepath del .csv con i domini della whitelist
    mm_list : dict
        Dizionario con chiave i domini contenuti nel Majestic Million
    bf_list : dict
        Dizionario con chiave i domini contenuti nel Bambenek Feed e value un obj
        che contiene info quali la famiglia del DGA
    analyzed_traffic : dict
        Dizionario con chiave i domini già contenuti nella whitelist

    Returns
    -------
    analyzed_traffic : list
        analyzed_traffic[0] : dict
            Contiene i domini già nella whitelist
        analyzed_traffic[1] : dict
            Contiene i domini già nella blacklist
    """

    with lzma.open(filepath_log, mode='rt') as garr_feed:

        line = garr_feed.readline()
        gdomain = GarrLog(domain=line.split(';')[5].rstrip('.'),
                          #resolved_ip=line.split(';')[7] #Esempio per aggiungere altri param
etri
                          ) #Carica la prima linea del log
        output_file = open(filepath_out, mode='a')
        output = csv.writer(output_file)
```

```
        while line:

            ### Confronto ###
            gdomain.domain = line.split(';')[5].rstrip('.') #Carica linea del log
            domain = gdomain.domain

            if domain in mm_list: #Se il dominio sta nella lista del MM
                if domain not in analyzed_traffic[0]: #Se il dominio non è già stato analizza
to
                    analyzed_traffic[0].update({domain : MajesticMillion(domain = domain,
                                                                         family = "majesti
c")}) #Appende i nuovi dati al dizionario in RAM
                    output.writerow(["legit", "majestic", domain]) #Appende i nuovi dati al f
ile di testo"""

            else: #Se non sta in MM potrebbe essere malevolo

                #Questo else accelera di molto l'esecuzione dato che le chiamate ai domini
                #del MM sono infinitamente di più rispetto a quelle ai domini malevoli.
                #Inoltre, poichè non si ha intersezione tra MM e BF, funziona.

                if domain in bf_list: #Se il dominio sta nella lista del BF
                    if domain not in analyzed_traffic[1]: #Se il dominio non è già stato anal
izzato
                        analyzed_traffic[1].update({domain : BambenekFeed(domain = domain,
                                                                          family = bf_list[domai
n].family)}) #Appende i nuovi dati al dizionario in RAM
                        output.writerow(["dga" , bf_list[domain].family, domain]) #Appende i
nuovi dati al file di testo

            line = garr_feed.readline()

    return  analyzed_traffic
```

Anche in questo caso il codice è commentato, e vista la crucialità della sezione riportata, si è deciso di commentare a volte ogni singola linea e di riportare i commenti in questo esempio.

La funzione viene utilizzata per confrontare i domini di un file di log con i domini relativi al Bambenek Consulting feed e a quelli del Majestic Million. Ci sarà poi un algoritmo che implementerà il ripetersi di questa funzione 24 volte, per ogni ora di un giorno di test, e altre 29 volte ancora per completare l'analisi dei dati. Infine non ripeteremo concetti tecnici già espressi nei commenti.

Bibliografia

1. TCP/IP Overview and History. ***The TCP/IP Guide.*** **[Online]** Settembre 2005. [Riportato: 22 Giugno 2020.] http://www.tcpipguide.com/free/t_TCPIPOverviewandHistory.htm.

2. Postel, J. User Datagram Protocol: Internet Standard. ***RFC 768.*** **[Online]** 28 Agosto 1980. [Riportato: 24 Giugno 2020.] https://tools.ietf.org/html/rfc768.

3. Darpa Internet Program. Transmission Control Protocol: Protocol Specification. ***RFC 793 STD 7.*** **[Online]** Settembre 1981. [Riportato: 24 Giugno 2020.] https://tools.ietf.org/html/std7.

4. Domain Name System. ***Wikipedia: The Free Encyclopedia.*** **[Online]** [Riportato: 17 Giugno 2020.] https://en.wikipedia.org/wiki/Domain_Name_System.

5. Elisan, C.C. *Malware, Rootkits and Botnets: A Beginner's Guide.* s.l. : McGraw Hill, 2013. 978-0-07-179205-9.

6. Malware. ***Wikipedia, L'enciclopedia libera.*** **[Online]** [Riportato: 17 06 2020.] https://it.wikipedia.org/wiki/Malware.

7. McAfee Labs Threats Report - August 2019. ***McAfee Labs.*** **[Online]** Agosto 2019. [Riportato: 16 Giugno 2020.] https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf.

8. Virus (informatica). ***Wikipedia, L'enciclopedia libera.*** **[Online]** [Riportato: 17 Giugno 2020.] https://it.wikipedia.org/wiki/Virus_(informatica).

9. *A survey of botnet detection based on DNS.* **Alieyan K., ALmomani A., Manasrah A., Kadhum M.M.** 2017, The Natural Computing Applications Forum.

10. *A framework for DNS based detection and mitigation of malware infections on a network.* **Stalmans E., Irwin B.** s.l. : IEEE, 2011, Vol. Information Security South Africa.

11. Mazzetti, A. *Reti Neurali Artificiali.* s.l. : Apogeo, 1991. 88-7303-002-5.

12. *Long short-term memory.* **Hochreiter S., Schmidhuber J.** 1997, Neural Comput., p. 1735-1780.

13. D.O., Webb. *The organization of behavior; a neuropsychological theory.* New York : Wiley, 1949.

14. Minsky M., Papert S. *Perceptrons: an introduction to computational geometry.* s.l. : MIT Press, 1969. ISBN 0-262-63111-3.

15. *A thorough review on the current advance of neural network structures.* **Dupond, S.** s.l. : Annual Reviews in Control, 2019.

16. *A LSTM based framework for handling multiclass imbalance in DGA botnet detection.* **Tran D., Mac H., Tong V., Tran H.A., Nguyen L.G.** 2017, Neurocomputing.

17. *The protection of information in computer systems.* **Saltzer J.H., Schroeder M.D.** s.l. : Proceedings of the IEEE, 1975.

18. *An Evaluation of DGA Classifiers.* **Sivaguru R., Choudhary C., Yu B., Tymchenko V., Nascimento A., De Cock M.** 2018, IEEE International Conference on Big Data.

19. ***Consortium GARR.*** **[Online]** [Riportato: 24 Giugno 2020.] https://www.garr.it/it/.

20. Lempel–Ziv–Markov chain algorithm. ***Wikipedia: The Free Encyclopedia.*** **[Online]** [Riportato: 17 Giugno 2020.] https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Markov_chain_algorithm.

21. Free OSINT. ***Bambenek Consulting Free OSINT.*** **[Online]** [Riportato: 17 Giugno 2020.] https://osint.bambenekconsulting.com/feeds/.

22. Majestic Million. ***Majestic.*** **[Online]** [Riportato: 15 Giugno 2020.] https://it.majestic.com/reports/majestic-million.

23. New action to disrupt world's largest online criminal network. ***Microsoft.com.*** **[Online]** 10 Marzo 2020. [Riportato: 10 Giugno 2020.] https://blogs.microsoft.com/on-the-issues/2020/03/10/necurs-botnet-cyber-crime-disrupt/.

24. Information-stealing 'Vawtrak' malware evolves, becomes more evasive. ***Sophos.com.*** **[Online]** 19 Dicembre 2014. [Riportato: 10 Giugno 2020.] https://nakedsecurity.sophos.com/2014/12/19/information-stealing-vawtrak-malware-evolves-becomes-more-evasive/.

25. *Predicting domain generation algorithms with long short-term memory networks.* **Woodbridge J., Anderson H.S., Ahuja A., Grant D.** 2016, CoRR abs/1611.00791.

26. ISO/IEC 7498-1:1994. ***iso.org.*** **[Online]** 1994. [Riportato: 22 Giugno 2020.] https://www.iso.org/standard/20269.html**.**

Ringrazio i chiam.mi Relatore, Prof. Luca Spalazzi, e Correlatori, Prof. Alessandro Cucchiarelli e Prof. Christian Morbidoni, per la grandissima conoscenza e per l'infinita disponibilità che hanno messo a disposizione per il completamento di questo lavoro.

Ringrazio i colleghi, omonimi o quasi, David e Davide per essere stati i compagni migliori in fin troppe occasioni.

Ringrazio la mia famiglia per avermi concesso le opportunità che mi hanno reso ciò che sono.

Ringrazio Elena per essere la luce che illumina questo percorso e per darmi sempre la forza di un sole che nasce.