POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Threat Intelligence Platforms (TIP)

Relatore
prof. Antonio Lioy

Candidato
Gabriele Chiesura

Anno accademico 2016-2017

† A mio nonno Luigi

Sommario

La tesi tratterà del lavoro che ho svolto nella divisione di sicurezza informatica di una società di consulenza e servizi tecnologici. Il progetto consiste nello sviluppo di una piattaforma che si presenta come un'applicazione web accessibile da browser (SaaS, Software as a Service) e che fornisce un servizio di threat intelligence. Le piattaforme di threat intelligence (TIP, Threat Intelligence Platform) sono una tecnologia emergente. Aiutano le organizzazioni a raccogliere, aggregare, correlare, contestualizzare ed analizzare minacce provenienti da più sorgenti in tempo reale, che possono essere un pericolo per la loro infrastruttura o per i loro servizi, aiutandole a prendere decisioni riguardo la prevenzione, il monitoraggio e la risposta ad eventuali incidenti. Si sono evolute per gestire l'enorme quantità di dati generati da fonti interne (log di sistema o degli apparati di rete) ed esterne (open source intelligence o feed da altre piattaforme) e per aiutare le squadre di sicurezza informatica ad identificare le minacce che sono rilevanti per la propria organizzazione. Importando informazioni da più fonti e con formati diversi, correlandole ed esportandole in un esistente sistema di ticketing o gestione degli incidenti aziendale, queste piattaforme permettono di automatizzarne la gestione proattiva e la loro mitigazione. La tesi introdurrà la threat intelligence, evidenziandone gli obiettivi e mostrandone i casi d'uso. Descriverà le piattaforme di threat intelligence, descrivendone i vantaggi, gli obiettivi e le caratteristiche generali di questo tipo di prodotto, accennando alcuni esempi di piattaforme esistenti. Descriverà l'architettura della piattaforma su cui ho lavorato e mostrerà come ho collaborato al fine di soddisfare i requisiti di monitoraggio che i clienti hanno manifestato. Descriverà quindi l'analisi che ho svolto e le scelte di progetto per ognuno dei moduli sviluppati, mostrandone infine i risultati ottenuti. Nelle conclusioni parlerò delle migliorie che sto continuando ad apportare ai moduli, avendo continuato a proseguire con il progetto nell'azienda.

Ringraziamenti

Vorrei ringraziare innanzitutto la mia famiglia, che è sempre stata presente e capace di darmi supporto durante tutti questi anni, aiutandomi e dandomi ottimi consigli nei momenti di difficoltà che ho incontrato.

Ringrazio sicuramente tutti i miei compagni e colleghi dell'università, in particolare Paola e Fabio, con cui ho condiviso tutto il percorso e che hanno reso piacevole studiare e passare le giornate al Politecnico.

Ringrazio il prof. Antonio Lioy, relatore di questa tesi, che con disponibilità e pazienza mi ha guidato nella sua stesura. Grazie al suo corso "Sicurezza dei sistemi informatici" mi sono appassionato a questa importante, vasta ed interessante materia. La qualità del suo insegnamento mi ha permesso di intraprendere senza difficoltà l'esperienza lavorativa che mi ha portato alla scrittura di questo elaborato.

Non dimentico e ringrazio inoltre i docenti che in questi due anni hanno contribuito alla mia formazione.

Ringrazio l'azienda, che mi ha dato l'opportunità di collaborare al progetto che è diventato tema di questa tesi. In particolare il dottor A. Busà, tutor aziendale, che mi ha seguito ed aiutato ad approfondire e applicare molti aspetti teorici da me studiati.

Indice

Capitolo 1

Introduzione

1.1 La threat intelligence

L'intelligence è il raccoglimento di informazioni che aiutano decisioni, con lo scopo di evitare un attacco o ridurre il tempo necessario a scoprire la presenza di una minaccia. La Cyber Threat Intelligence (o CTI) è un nuovo campo, che applica i concetti dell'intelligence alle minacce informatiche. Fornisce l'analisi degli attaccanti, delle loro motivazioni e dei loro metodi d'attacco grazie al raccoglimenti di dati, al loro arricchimento e alla loro contestualizzazione. Una volta integrata nei processi di protezione, la CTI aumenta il livello di sicurezza e di consapevolezza dell'azienda.

1.1.1 Utilità

Un uso efficace può permettere di capire:

Chi sta attaccando Permette di attribuire attacchi o attività malevole ad attori o gruppi di attori (cyber criminali, hacktivists, agenzie governative/nazionali)

La causa Capendo chi c'è dietro ad un attacco potrebbe fornire informazioni sulle motivazioni degli attori, quanto impegno ci metteranno (Advanced Persistent Threat, APT, oppure attacchi opportunistici) e quanto specifici saranno

Il fine Con questa informazione i difensori possono prioritizzare le azioni, basandosi sui prossimi obiettivi degli attaccanti

Come stanno attaccando I cosiddetti TTPs (Tactics, Techniques and Procedures), gli strumenti e le infrastrutture che usano

Da dove Correlando il paese di provenienza degli attaccanti e la sua situazione geopolitica aiuta i difensori a capire i loro nemici

Cosa li caratterizza I cosiddetti IOCs (Indicators Of Compromise), che aiutano a rilevare la presenza in un sistema e ad eventualmente segnalarla

Come fermarli Informazione che l'azienda può usare per proteggersi

1.1.2 Fonti e obiettivi

La cyber threat intelligence proviene da varie fonti:

Interne

Qualsiasi informazione raccolta dall'interno dell'organizzazione. Possono essere informazioni riportate dagli strumenti di sicurezza e dispositivi di rete (firewall, IDS, IPS) o dalle macchine e dispositivi degli utenti (antivirus). Una importante parte di intelligence proviene anche da analisi forense[1], capace di trovare materiale non immediatamente visibile o disponibile e che può essere utile al rilevamento di altri attacchi.

Comunità

Include qualsiasi informazione scambiata attraverso una relazione fidata con gruppi o membri che condividono lo stesso interesse. Sono presenti gruppi ben definiti e strutturati, come *Information Sharing and Analysis Centers* (ISACs [1]), i quali si dividono per settori, ad esempio servizi finanziari (FS-ISAC).

Esterne

Sono tutte quelle che non sono nè provenienti dall'interno dell'organizzazione nè da comunità o gruppi. Rientrano in questa categoria quindi tutte quelle di pubblico dominio o appartenenti a servizi privati, che forniscono informazioni acquistando una licenza per il loro servizio o un loro prodotto.

Tra le principali ci sono:

Open Source Intelligence (OSINT) Fonti disponibili pubblicamente, come giornali o riviste, radio o televisione, internet, articoli scientifici

Social Media Intelligence (SOCMINT) Materiale reperibile sui social networks

Human Intelligence (HUMINT) Materiale ricavato dal contatto interpersonale, come spionaggio, interviste, interrogatori

Deep Web Intelligence Il deep web è la parte del World Wide Web (WWW) non indicizzato dai normali motori di ricerca (opposto al classico "surface web"), come web mail, operazioni bancarie online, contenuto non HTML (immagini, video o audio), pagine a contenuto dinamico

Dark Web Intelligence Un sottoinsieme del deep web che permette ai suoi utenti di anonimizzare il loro accesso. Fa uso di reti come Tor [2] o I2P [3], che permettono anche di rendere anonimo l'hosting di un sito web. Grazie ad un alto livello di cifratura, risulta praticamente impossibile tracciare gli indirizzi IP degli utenti, permettendo lo scambio confidenziale ed anonimo di dati. Per le sue proprietà è spesso usata per operazioni illegali, come terrorismo, vendita illegale o botnet

La missione della CTI è ricercare e analizzare sviluppi tecnologici in tre aree:

- Cyber crimine
- Cyber attivismo (hacktivism)
- Cyber spionaggio

[1] L'informatica forense ricerca prove da computer e altri dipositivi di memorizzazione digitale, con lo scopo di identificare e/o recuperare informazioni utili

I dati raccolti ed analizzati durante la ricerca permettono di attuare misure preventive. E' importante capire che non bisogna usare i dati raccolti per una semplice analisi "a posteriori". In seguito ad un attacco, i relativi indicatori (*Indicators of Compromise*, IoC) sono infatti disponibili solo dopo giorni se non mesi. Una buona intelligence è quindi quella che non si focalizza su metodi d'attacco già utilizzati o noti, ma si concentra sulla comprensione dei correnti eventi che stanno succedendo nell'azienda con lo scopo di far emergere anche le azioni più difficili da notare, correlando molteplici indicatori per avere un quadro più chiaro della situazione. Considerando i seri impatti delle minacce informatiche, la CTI è diventata una soluzione efficace al mantenimento della sicurezza nazionale.

1.1.3 I quattro tipi di threat intelligence

Secondo il *Centro per la Protezione dell'Infrastruttura Nazionale Inglese* (CPNI), ci sono quattro tipi di threat intelligence:

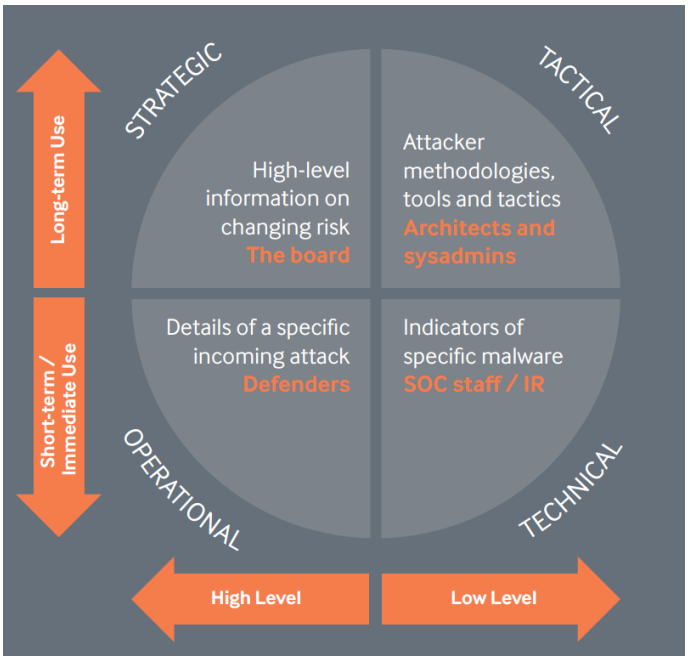

Figura 1.1. Tipi di threat intelligence (fonte: [4])

Intelligence strategica

Suggerisce decisioni o policy riguardo minacce a lungo termine o su larga scala per evitare che l'organizzazione venga attaccata. Fornisce uno schema generale delle intenzioni e capacità di una minaccia, includendone i TTPs.

Esempio di flusso di lavoro:

1. Determinare se le potenziali minacce si sono realizzate nel passato
2. Informarsi tramite altre organizzazioni se ci sono ulteriori minacce
3. Elencare chi potrebbe beneficiare dei dati raccolti per il corretto funzionamento dei servizi

Intelligence tattica

Più a basso livello, si focalizza su eventi specifici. Suggerisce su cosa un organizzazione dovrebbe concentrare l'attenzione per rispondere con gli strumenti a sua disposizione agli incidenti che la riguardano (tipicamente gli indicatori sono indirizzi IP, nomi DNS e hash).

Esempio di flusso di lavoro:

1. Estrarre indicatori chiave dai report degli incidenti di sicurezza e dalla documentazione dei gruppi di minacce
2. Determinare i cambiamenti necessari all'organizzazione per essere meno affetti dalle minacce determinate
3. Identificare gli aggiornamenti periodici che devono essere attuati ai propri strumenti e alla propria infrastruttura

Intelligence operativa

Dati i dettagli di un attacco specifico, valuta l'abilità dell'organizzazione di proteggersi dalle future minacce.

Esempio di flusso di lavoro:

1. Elencare chi contattare se la propria organizzazione ricevesse l'avviso di un attacco imminente
2. Cercare sul web informazioni pubblicate poco prima l'attacco subito
3. Se non sono presenti informazioni, identificare altri potenziali fattori di scatenamento dell'attacco

Intelligence tecnica

Sono gestiti e analizzati indicatori di basso livelllo (ad esempio IP di Command & Control o hash di malware).

Esempio di flusso di lavoro:

1. Ottenere la lista delle minacce giornaliera dal CISP (Cybersecurity Information Sharing Partner[2]) o da altre fonti e ad esempio impostare regole firewall o IDS per determinati IP
2. Controllare regolarmente le connessioni in uscita dalla propria rete interna
3. Se necessario, iniziare la procedura di risposta agli incidenti di sicurezza

1.1.4 Casi d'uso

Ci sono vari casi in cui l'utilizzo della threat intelligence fornisce un grande aiuto. I principali possono essere riassunti nella tabella in figura 1.1, ognuno caratterizzato dalla sua specificità (che dipenderà dalla specificità degli indicatori incontrati nel caso d'uso), dal tipo di intelligence (strategica o tattica) e dai vantaggi che l'utilizzo della threat intelligence garantisce.

[2]E' un gruppo formato da governi e organizzazioni con l'obiettivo di scambiare informazioni su minacce informatiche in tempo reale, in maniera sicura, confidenziale e in un ambiente dinamico, aumentando la conoscenza e l'attenzione sulle attuali minacce, riducendone quindi l'impatto

Caso d'uso	*Specificità*	*Strategica/Tattica*	*Vantaggi*
Pianificazione di sicurezza	Bassa	Strategica	Supporto decisionale e risposta efficace agli attacchi mirati
Risposta agli incidenti	Media	Entrambe	Riduzione tempo di contenimento, corretta identificazione e obiettivi dell'incidente
Monitoraggio infrastruttura	Alta	Tattica	Riduzione tempo di risposta, di escalation e di determinazione falsi positivi

Tabella 1.1. Casi d'uso della threat intelligence

Pianificazione di sicurezza

L'uso più largo della threat intelligence è la pianificazione per il futuro. L'intelligence strategica in questo caso è la più utile, che permette di raccogliere informazioni riguardo ai TTPs. Conoscendo dove gli attaccanti spenderanno le loro risorse nei loro attacchi permette di investire sulla propria difesa in maniera efficace e precisa.

Risposta agli incidenti

La risposta agli incidenti invece riesce a trarre vantaggio da entrambi i tipi di intelligence. Quando un incidente si presenta, l'intelligence tattica viene usata per determinarne gli obiettivi. Se ad esempio l'hash MD5 di un file malevolo viene trovato durante l'analisi in un dispositivo, gli obiettivi dell'incidente verranno estesi anche a quest'ultimo. Quando la squadra di risposta agli incidenti sarà stata in grado di attribuire all'incidente un attore o un gruppo di attori, l'intelligence stragetica risulta utile a definire i loro TTPs, che aiuteranno a determinare la direzione dei prossimo eventuali attacchi per focalizzare meglio la ricerca. Se ad esempio gli attori di quell'incidente preferiscono le *web shell* come metodo d'attacco, l'attenzione andrà spostata sui server web.

Non è detto che tutti gli indicatori o gli allarmi debbano portare ad un blocco o inserimento in "blacklist". Le azioni sono solitamente determinate dal rapporto tra impatto e livello di fiducia. Una vulnerabilità del sistema operativo Windows ha ad esempio un impatto nullo su un sistema di macchine Linux, oppure un indicatore confermato da fonti open-source (OSINT) ha un livello di fiducia inferiore ad uno confermato da un autorevole fornitore di threat intelligence. La risposta alla presenza di un indicatore segue quindi la matrice in figura 1.2, dove la voce *contesto* indica che è necessaria la correlazione dell'indicatore con altre fonti di intelligence per approfondirne i dettagli, la voce *allarme* indica che l'indicatore può essere integrato con gli strumenti di monitoraggio per segnalarne l'eventuale rilevazione (IDS, IPS), mentre la voce *blocco* conferma che l'indicatore è sicuramente da mettere in blacklist nei firewall, filtri e dispositivi di prevenzione.

Impatto/Fiducia	*Sconosciuta*	*Bassa*	*Media*	*Alta*
Nessuno	ignora	contesto	contesto	contesto
Basso	contesto	contesto	contesto	contesto
Medio	contesto	alert	alert	alert
Alto	contesto	alert	blocco	blocco
Critico	contesto	alert	blocco	blocco

Tabella 1.2. Matrice Impatto/Fiducia

Monitoraggio infrastruttura

L'intelligence usata in questo caso è quella tattica, venendo a contatto con indicatori di basso livello. Se ad esempio si presenta l'allarme di un antivirus, si inizia a correlarlo con i log, le catture del traffico di rete e i feed di intelligence delle varie piattaforme (indicatori di minacce note), per poter passare subito da allarme ad incidente ed iniziare il processo di risposta. E' importante in questo caso l'integrazione tra la threat intelligence ed il SOC, in modo da permettere a quest'ultimo un'analisi e una ricerca efficace.

1.2 Threat Intelligence Platforms

1.2.1 Approccio tradizionale alla sicurezza

L'approccio tradizionale alla sicurezza delle aziende coinvolge squadre di sicurezza che usano diversi strumenti e processi per condurre risposta agli incidenti, difesa della rete e analisi delle minacce. L'integrazione fra questi team e la condivisione di informazioni di interesse è spesso un processo manuale, che avviene attraverso scambio di mail, di fogli elettronici o portali di ticketing. Questo approccio non scala se i team, l'azienda o il numero di minacce e incidenti crescono. I grandi SOC (Security Operations Centers[3]) producono ed elaborano infatti migliaia di eventi al giorno. L'evoluzione delle metodologie d'attacco degli ultimi tempi ha fatto emergere quindi i limiti degli approcci tradizionali. Spesso infatti i flussi tra scoperta della minaccia, analisi ed integrazione dei suoi indicatori negli strumenti di monitoraggio e prevenzione dell'infrastruttura sono indipendenti gli uni dagli altri rispetto al tipo di indicatore, alla minaccia e alla fonte, come mostrato in figura 1.2.

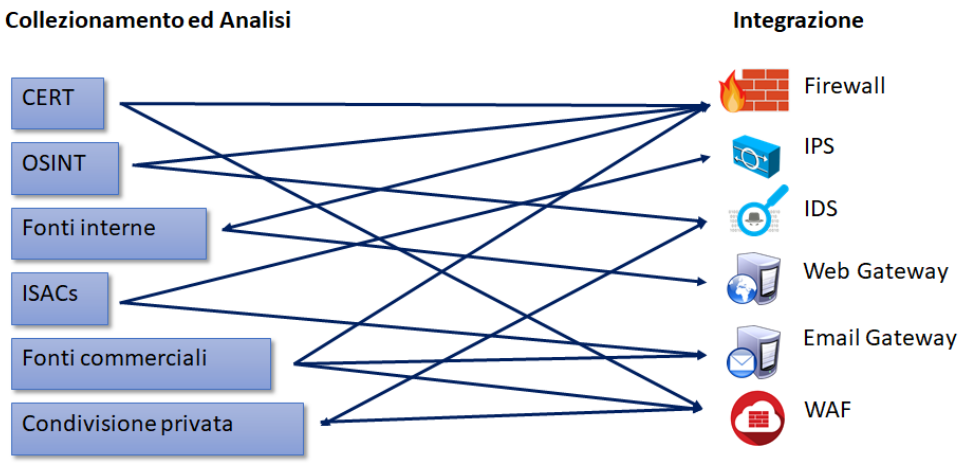

Figura 1.2. Approccio tradizionale

1.2.2 Obiettivi e funzioni delle TIP

Le piattaforme di threat intelligence mirano a risolvere proprio questo problema, cioè il collezionamento, l'archiviazione e la condivisione dell'intelligence, in modo da integrarne i risultati con gli strumenti di prevenzione e monitoraggio. Il loro vantaggio è quello di permettere la condivisione di informazioni in un luogo comune a tutti i team che si occupano della sicurezza dell'azienda, per coordinarsi e ricevere consigli e linee guida sulle contromisure da mettere in atto.

[3]Squadra di persone esperte di sicurezza informatica, che si occupano di monitorare e migliorare l'infrastruttura di un'azienda, attraverso prevenzione, rilevazione, analisi e risposta agli incidenti di sicurezza informatica

L'elaborazione di una minaccia si articola quindi in varie fasi, supportate da flussi di lavoro automatizzati che permettono di far passare una minaccia attraverso individuazione, gestione, analisi e processo di difesa, tracciandola fino al completamento. Nel dettaglio, le funzionalità principali:

Collezionamento Raccoglie e aggrega diversi formati di informazione provenienti da sorgenti diverse, come CSV, XML e JSON, da fonti pubbliche (OSINT), da social networks (SOCMINT) o dalle altre descritte precedentemente

Correlazione Analizza automaticamente e correla le informazioni, producendo il "who, why and how" di una certa minaccia, suggerendo (se possibile) le contromisure da adottare

Arricchimento e Contestualizzazione Mediante strumenti di terze parti o analisti di sicurezza, le informazioni sulle minacce raccolte dalla piattaforma possono essere arricchite, permettendo ai SOCs o ai team di Incident Response (IR) di avere quanti più dati possibili riguardo un *threat actor*, le sue potenzialità e la sua infrastruttura così da poter attuare la migliore difesa

Analisi Analizza automaticamente gli indicatori delle minacce e le relazioni che ci sono tra di loro. L'analisi permette l'identificazione di TTPs. In aggiunta, tecniche di visualizzazione e grafi di correlazione permettono di mostrare complesse relazioni, utili per visualizzare i dati sotto altre prospettive

Integrazione I dati generati devono offrire la possibilità di essere integrati con gli strumenti di sicurezza e i prodotti usati dall'azienda, consentendo il flusso diretto dall'indicatore di compromissione o incidente al sistema di ticketing, ai firewall, agli IDS o agli antivirus

Azione Una TIP evoluta deve anche gestire la risposta alla rilevazione di un attacco. Può creare un punto in comune tra i team di sicurezza delle organizzazioni e gli sviluppatori, in modo da agire congiuntamente, collaborando per sviluppare una soluzione alla minaccia. Può anche suggerire azioni di rimedio all'architettura di sicurezza e all'infrastruttura dell'organizzazione

Con l'utilizzo di una TIP, lo schema mostrato in figura 1.2 si trasformerebbe quindi in quello mostrato in figura 1.3. La piattaforma svolge il ruolo di concentratore di tutta l'informazione ottenuta dalle numerose fonti, convertendola in un formato comune. Grazie al lavoro di un team di analisti l'intelligence raccolta può essere approfondita ed arricchita con ulteriori dettagli e verifiche. Infine gli indicatori estratti dalle analisi sono integrati con i dispositivi di sicurezza con i loro formati specifici.

1.2.3 Organizzazione dell'informazione

Avere una struttura ordinata ed uniforme in cui archiviare l'intelligence prodotta è di grande utilità per utilizzarla in modo agevole in caso di necessità. Un formato organizzato e *machine readable* è quindi necessario per ottenere tutti i vantaggi della threat intelligence. Sono stati definiti vari linguaggi per descrivere informazioni di intelligence come incidenti, attori e indicatori di compromissione:

- openIOC (Open Indicators of Compromise)
- STIX (Structured Threat Information Expression)
- CybOX (Cyber Observables)
- TAXII (Trusted Automated Exchange of Indicator Information)

Molti di questi sono ancora in evoluzione e il loro utilizzo e diffusione fra gli strumenti di monitoraggio è ancora inconsistente.

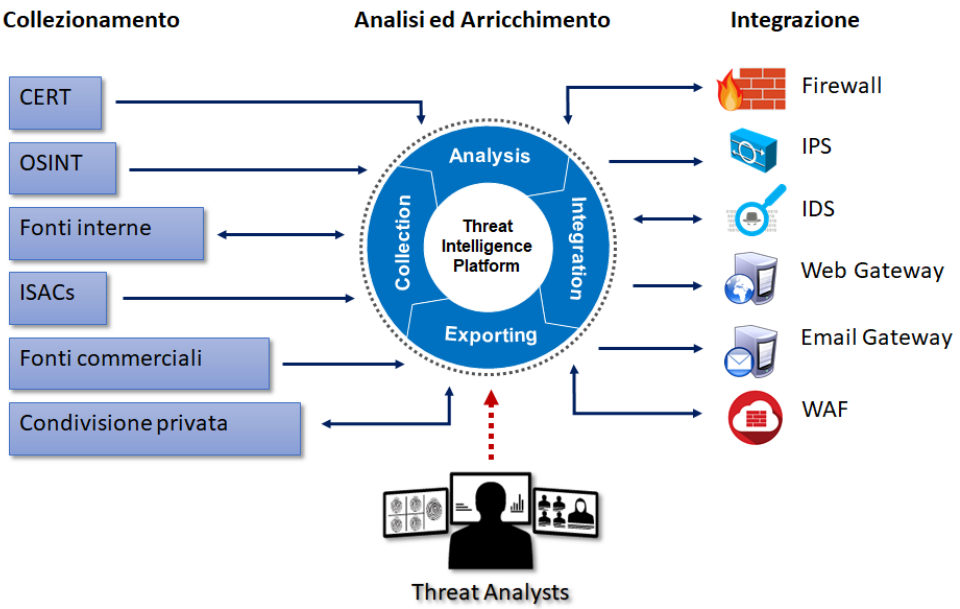

Figura 1.3. Approccio con TIP

STIX [5]

E' un linguaggio ed un formato di serializzazione open-source, usato per scambiare cyber threat intelligence in maniera consistente e machine-readable, in modo da essere di supporto all'analisi collaborativa delle minacce e lo scambio automatizzato di informazioni di intelligence. Definisce 12 oggetti (*STIX Domain Objects*, SDOs) e due tipi di relazioni (*STIX Relationship Objects*, SROs). Gli oggetti sono:

Attack Pattern Un tipo di TTP che descrive i modi in cui gli attaccanti cercano di compromettere un obiettivo

Campaign Un insieme di comportamenti degli attaccanti che descrivono un gruppo di attività malevole o attacchi che succedono in un determinato periodo e contro uno specifico insieme di obiettivi

Course of Action Un'azione presa per prevenire o rispondere ad un attacco

Identity Individui, organizzazioni o gruppi

Indicator Una caratteristica che permette di rilevare attività informatiche malevole o sospette

Intrusion Set Un gruppo di comportamenti e risorse con proprietà comuni che si crede siano governate da un singolo attore

Malware Un tipo di TTP, conosciuto anche come codice o programma malevolo, usato per compromettere confidenzialità, integrità o disponibilità delle informazioni o dei sistemi della vittima

Observer Data Rappresenta informazioni ossservate in un sistema o in una rete (ad esempio un indirizzo IP)

Report Collezione di threat intelligence focalizzata su uno o più temi, come la descrizione di un threat actor, un malware o una tecnica d'attacco, includendo i relativi dettagli

Threat Actor Individui, gruppi o organizzazioni che si crede stiano operando con intenzioni malevole

Tool Software che può essere usato da threat actor per compiere attacchi

Vulnerability Un errore in un software o di un sistema che può essere sfruttato per ottenere l'accesso ad un sistema o alla rete

Le relazioni invece sono:

Relationship Usato per collegare due oggetti, descrivendo la relazione che sussiste tra essi

Sighting Denota la credenza che un elemento di threat intelligence sia stato rilevato (ad esempio un indicatore o un malware)

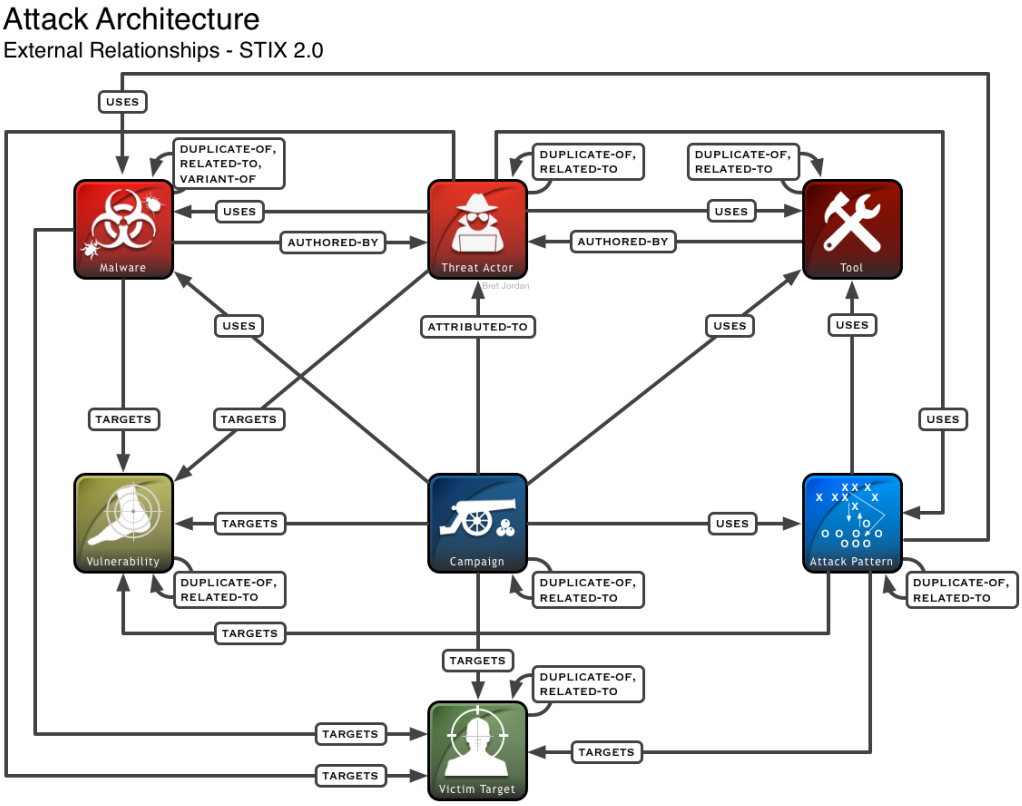

Figura 1.4. Oggetti e relazioni STIX (fonte: [6])

TAXII [7]

E' un protocollo a livello applicativo utilizzato per scambiare informazioni di threat intelligence su HTTPS. TAXII definisce RESTful API e una serie di requisiti sia per il client sia per il server. Definisce inoltre 2 servizi che permettono il supporto di una grande varietà di modelli di condivisione:

Collection Interfaccia ad una repository di oggetti di CTI che permette ad un producer di archiviare un insieme di informazioni di CTI che può essere richiesta da un consumer

Channel Mantenuto da un server TAXII, permette ad un producer di inviare dati a più consumer e permette ad un consumer di ricevere dati da più producer

I server TAXII sono rilevati grazie a query DNS ed usa HTTPS per tutte le comunicazioni. Permette di scegliere sia una modalità "push" sia una modalità "pull" e diversi tipi di messaggi, ad esempio di sottoscrizione, pausa di trasmissione, modifica iscrizione. E' stato sviluppato appositamente per scambiare CTI rappresentata in formato STIX.

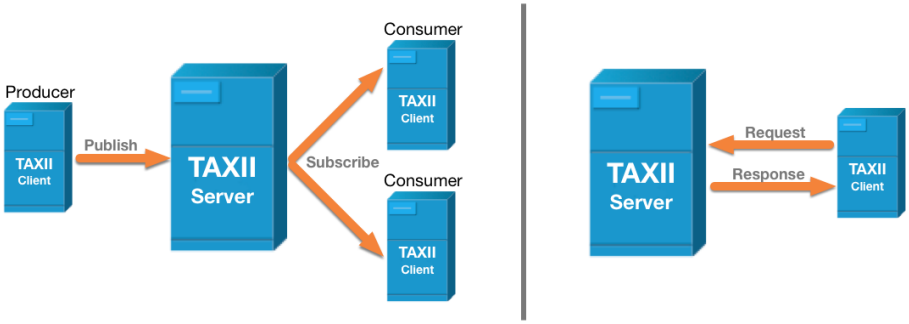

Figura 1.5. Modelli di condivisione TAXII (fonte: [8])

1.2.4 Caratteristiche comuni

Sebbene le varie piattaforme siano nate da idee indipendenti, il mercato incoraggia i produttori ad adottare un insieme di caratteristiche comuni da implementare in una piattaforma:

- Integrazione con sistemi SIEM
- Integrazione con strumenti di monitoraggio e prevenzione
- Arricchimento delle informazioni con OSINT e feed commerciali
- Condivisione con altre organizzazioni e/o con altre piattaforme

1.2.5 Analogie ed integrazione con sistemi SIEM

I SIEM (Security Information and Event Management) si occupano di due obiettivi: rilevare in (quasi) tempo reale gli incidenti di sicurezza e gestire efficacemente i log. Questi obiettivi erano in principio chiamati rispettivamente Security Event Management (SEM) e Security Information Management (SIM), ma oggi sono state fuse in un'unica entità chiamata SIEM. Ad alto livello, collezionano log ed informazioni da vari dispositivi di rete e li correlano per rilevare incidenti o comportamenti anomali delle attività, per archiviarli infine per un uso successivo (reporting, profilazione dei comportamenti). Quando configurati ed installati correttamente, aiutano le aziende a:

- Scoprire minacce interne ed esterne
- Monitorare gli accessi alle risorse da parte degli utenti
- Fornire relazioni di conformità (report of compliance)
- Supportare la risposta agli incidenti

Come visibile in figura 1.6, per ogni dispositivo è presente un *collettore*, il cui compito è quello di normalizzare i dati ricevuti prima di passarli all'*engine* del SIEM, cioè il corpo centrale incaricato di effettuare analisi e correlazioni sui dati, che verranno poi archiviate definitiamente nella base dati.

Combinare una piattaforma di threat intelligence con un sistema SIEM permette di ottenere i massimi vantaggi da entrambi. Uno schema di integrazione potrebbe essere quello in figura 1.7. La piattaforma raccoglie feed di minacce da fonti OSINT, commerciali e da altre piattaforme open-source. Le analisi su di essi permettono di essere confermati, contestualizzati ed arricchiti di dettagli. In caso di minacce o indicatori confermati, i relativi indicatori vengono inoltrati a dispositivi di prevenzione e di monitoraggio.

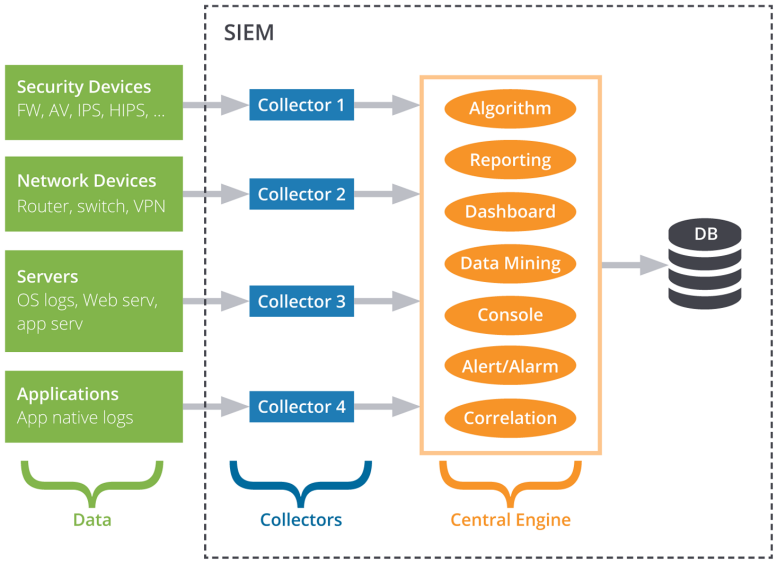

Figura 1.6. Architettura SIEM (fonte: [14])

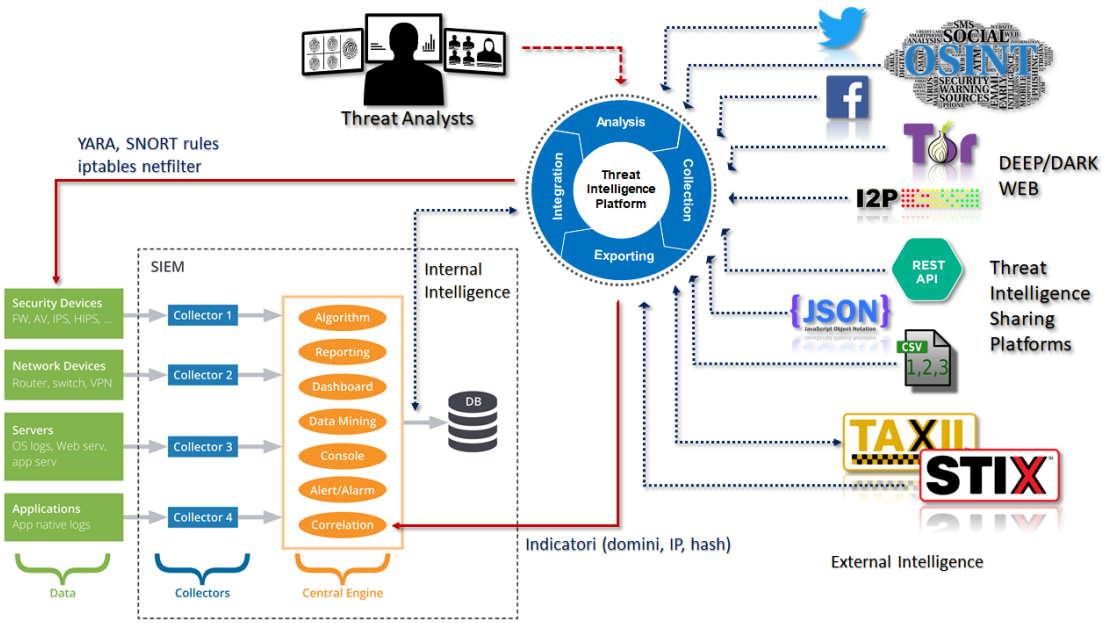

Figura 1.7. Integrazione TIP - SIEM

1.2.6 Esempi di piattaforme open-source

Sono nati vari progetti open-source con l'obiettivo di sviluppare una piattaforma. Alcuni esempi:

The Collective Intelligence Framework (CIF)[9] Un framework sviluppato dal *Research and Education Networking Information Sharing and Analysis Center* (REN-ISAC) nato per condividere indirizzi IP e domini, con supporto anche a hash. Espone API e offre estensioni per alcuni browser. Permette l'output in diversi formati e l'integrazione con vari strumenti come IDS o firewall

Collaborative Research Into Threats (CRITs)[10] Iniziata dalla *MITRE Corporation* e diventato un progetto open-source nel 2013, integra il protocollo TAXII per facilitare la condivisione, permette l'inserimento manuale di file STIX, domini, IP, email e altri indicatori. Permette l'export in vari formati: CSV, STIX e JSON

Mantis[11] Progetto di Siemens reso open-source. Permette di importare ed elaborare i più comuni linguaggi di descrizione di intelligence (IODEF, openIOC, STIX)

Malware Information Sharing Platform (MISP)[12] Sviluppata dalla NATO, aiuta a tracciare ed analizzare malware. Si integra con vari IDS e firewall, con varie fonti (import ed export openIOC), vari formati (XML, CSV) ed offre API RESTful

Avalanche o SOLTRA EDGE[13] Iniziativa del Financial Services Information Sharing and Analysis Center (FS-ISAC) di creare una piattaforma comune per la condivisione di indicatori di compromissione. Nato con l'idea di un prodotto gratis per l'utilizzo da parte dei membri delle organizzazioni di FS-ISAC, si è sviluppato in prodotto quasi commerciale usato anche da altri gruppi di condivisione.

Capitolo 2

Analisi

2.1 Soluzioni esistenti

Le piattaforme esistenti sono molto valide dal punto di vista delle numerose fonti da cui riescono ad ottenere le informazioni. Integrano numerosi servizi open-source ed alcune hanno un enorme base dati con uno storico che risale anche a molti anni fa, permettendo non solo l'analisi sui dati attuali, ma anche una ricerca nelle risorse passate. Tuttavia la maggior parte di tutte queste notifiche e segnalazioni servono a ben poco se non contestualizzate o arricchite di dettagli e verifiche. Le piattaforme in commercio forniscono infatti un gran numero di segnalazioni, ma sono tutte automatiche e prive di analisi che affermino una effettiva minaccia per il cliente. Mancano inoltre di integrazione con le caratteristiche, le specifiche e l'infrastruttura dei clienti, accettando in input un insieme limitato di informazioni, come il marchio, i domini e gli indirizzi IP. Nessuna offre un'integrazione più a basso livello o comunque personalizzata rispetto al cliente specifico, ricevendo ad esempio i loro log o informazioni in tempo reale provenienti dalla loro infrastruttura.

2.2 La piattaforma

La piattaforma sviluppata nel progetto a cui ho lavorato al contrario offre più che un prodotto un vero e proprio servizio di threat intelligence. La piattaforma è solo lo strumento che permette di ottenere informazioni dal web e dalle varie altre fonti e di visualizzarne i risultati, ma il grande vantaggio è rappresentato dall'analisi e la contestualizzazione effettuata da un team di analisti che controllano ogni segnalazione prodotta per fornire al cliente l'allarme di una minaccia verificata, completa di dettagli e azioni compensative per mitigarne gli impatti. Il servizio offre inoltre la possibilità di configurare la ricezione di dati e log provenienti dall'infrastruttura e dai dispositivi di rete del cliente (DNS, firewall), consentendo un'integrazione maggiore per fornire allarmi ancora più specifici e personalizzati.

2.2.1 Architettura

Tutte le macchine che compongono l'infrastruttura della piattaforma sono ospitate su un noto servizio di *cloud computing*. Secondo la definizione del NIST (National Institute of Standards and Technology [15]):

> Cloud computing enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Il cloud computing permette quindi a chi vuole fornire un nuovo servizio di concentrarsi solo sul suo design e sviluppo, senza preoccuparsi dell'infrastruttura fisica delle macchine e delle reti, riducendone drasticamente le complessità di gestione e i tempi di messa in produzione. Ci sono però anche svantaggi dal punto di vista della sicurezza e del trattamento dei dati, in quanto essi vengono affidati al fornitore del servizio di cloud computing, che deve garantire diverse proprietà, tra cui:

- Integrità dei dati
- Riservatezza dei dati
- Disponibilità dei dati e dei servizi
- Autorizzazione e controllo accessi

Le macchine utilizzano distribuzioni del sistema operativo Linux. Per accederci è necessario connettersi via SSH[1] autenticandosi fornendo una chiave RSA a 2048 bit ed ottenendo quindi un'interfaccia a linea di comando.

L'interfaccia di gestione delle istanze del servizio di cloud-computing permette il monitoraggio completo di ogni tipo di risorsa usata: traffico di rete, numero di letture/scritture su disco, occupazione di memoria RAM, percentuale di utilizzo della CPU. Permette inoltre di impostare allarmi di notifica quando il consumo di una delle risorse supera una certa soglia. L'interfaccia permette di configurarne i parametri di rete e fornisce inoltre la possibilità di configurare il *firewall virtuale* dell'istanza, che controlla tutte le connessioni in ingresso.

L'architettura della TIP è mostrata in figura 2.1 ed è composta da 3 elementi principali:

- Dashboard
- Engine
- Dati

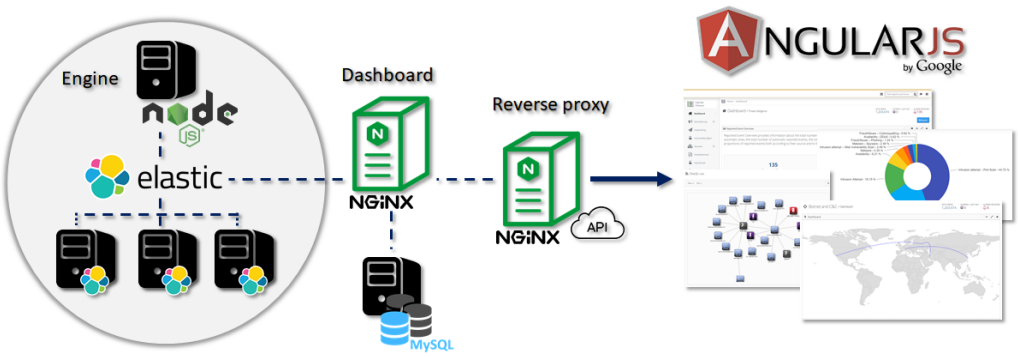

Figura 2.1. Architettura

Dashboard

L'interfaccia utente si presenta come una web application, accessibile quindi da browser, seguendo il modello di distribuzione *SaaS*, Software as a Service. Utilizza *AngularJS* (sezione 2.2.2), un framework che permette di caricare dinamicamente gli elementi e le sezioni richieste, dando all'utente

[1] Secure SHell, protocollo che permette di stabilire una sessione remota cifrata, fornendo un'interfaccia a linea di comando

la sensazione di rimanere nella stessa pagina. Gli utenti non accedono direttamente alla macchina della dashboard, ma passano attraverso un reverse proxy. Il dominio della URL della piattaforma è infatti associato all'indirizzo IP del reverse proxy. I vantaggi di questa soluzione sono molteplici, tra cui:

- Oscuramento ai client dell'infrastruttura della rete interna
- Esecuzione di operazioni di cifratura necessarie al protocollo HTTPS per alleggerire il carico del server applicativo
- Eventuale bilanciamento e distribuzione del traffico su più server
- Acceleratore applicativo (cache di contenuti)

Per la gestione del reverse proxy e per il server HTTP della dashboard utilizziamo *NGINX*, un software open-source. Come riporta il sito ufficiale [16]:

> NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Even if you don't expect to handle thousands of simultaneous requests, you can still benefit from NGINX's high-performance and small memory footprint. NGINX scales in all directions: from the smallest VPS all the way up to large clusters of servers.

La macchina di dashboard è quindi inaccessibile ed invisibile agli utenti, in quanto può solo dialogare con il reverse proxy.

Attraverso il reverse-proxy è possibile accedere anche alle *API* della piattaforma, che espongono alcune informazioni tra cui i feed e gli indicatori analizzati dagli analisti.

Engine

E' il motore che genera tutti gli allarmi che vengono notificati sulla piattaforma, effettuando le varie integrazioni con i servizi esterni. Utilizza Node.js ed è composto da moduli, ognuno per uno specifico servizio di monitoraggio (tra cui quelli sviluppati, descritti nel capitolo Progettazione 3). Il processo principale dell'engine (al quale sono "appesi" tutti i moduli) è eseguito grazie a *PM2*, un noto gestore di processi per applicazioni Node.js, che ne gestisce il logging, il monitoraggio ed il riavvio in caso di errori o superamento di una certa soglia di memoria allocata (memory leak). Ogni servizio di monitoraggio dipendente dal processo di engine principale sfrutta un modulo chiamato *node-shedule*, che permette di schedulare l'esecuzione di un programma Node.js ad esempio grazie a stringhe *crontab*[2]. Di seguito un esempio di schedulazione per eseguire una porzione di codice ogni giorno dal lunedì al venerdì alle 7:00.

```
var schedule = require("node-schedule");

schedule.scheduleJob("0 7 * * 1-5", function(){
  job();
});
```

Nella stessa macchina sono presenti vari script di monitoraggio, sia in Node.js (ma slegati dal processo principale) ma anche script *bash* (ad esempio il servizio Antidefacement, descritto nel capitolo 3.1).

[2]Comando linux per la pianificazione e schedulazione di comandi e processi

Dati

Possono ancora essere divisi in due gruppi:

MySQL Una macchina usata per contenere le informazioni riguardo alle utenze della piattaforma (indirizzi email, password, cliente), per gestire gli accessi (ed il loro log) e i permessi dei vari utenti e clienti.

Elasticsearch Tre macchine che formano un *cluster* Elasticsearch, il search engine usato dalla piattaforma (meglio dettagliato nell'apposita sezione 2.2.5) e che contiene tutti i dati utilizzati e mostrati dalla piattaforma, come gli indicatori, gli allarmi, gli incidenti di sicurezza. Viene anche utilizzato dai moduli dell'engine come supporto per le loro operazioni e la loro logica.

Siccome Elasticsearch non effettua in modo standard la cifratura dei dati che gestisce, le macchine del cluster sono state istanziate con disco cifrato, reso trasparente dal servizio di cloud-computing.

E' stato creato un indice per ogni strumento di monitoraggio e per ogni cliente, contenenti parametri, configurazioni e liste di risorse da monitorare.

2.2.2 AngularJS

Figura 2.2. AngularJS

AngularJS [17] è un framework JavaScript per lo sviluppo di applicazioni web lato cliente. Fornisce il suppporto e tutte le principali funzionalità per creare quelle che vengono chiamate *single page applications*, cioè applicazioni che vengono caricate dinamicamente, senza la necessità di ricaricare l'intera pagina. Tra le principali funzionalità:

- binding bidirezionale
- iniezione delle dipendenze
- supporto al pattern MVC (Model View Controller)
- supporto ai moduli
- separazione delle competenze
- testabilità del codice
- riusabilità dei componenti

Di seguito un esempio basico tratto dal sito ufficiale di AngularJS, che mostra il funzionamento del "binding" tra modello e vista:

```
<!doctype html>
<html ng-app>
  <head>
    <script
        src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>
  </head>
```

```
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```

Gli elementi di interesse sono:

tag ng-app Specifica l'elemento radice dell'applicazione Angular

tag ng-model Permette di associare il modello (la variabile) a cui verrà associato il valore rappresentato dal tag. In questo caso la stringa inserita nel campo di testo

{{ }} Quanto inserito all'interno delle due parentesi viene interpretato come codice javascript dello *scope* di Angular

Modificando il contenuto del tag input, il framework aggiorna automaticamente il modello (la variabile *yourName*) e la rispettiva vista attraverso l'espressione "{{ }}".

2.2.3 Node.js

Figura 2.3. Node.js

Come descritto sul sito uffficiale di Node.js [18]:

> Node.js è un runtime Javascript costruito sul motore JavaScript V8 di Chrome. Node.js usa un modello I/O non bloccante e ad eventi, che lo rende un framework leggero ed efficiente. L'ecosistema dei pacchetti di Node.js, npm, è il più grande ecosistema di librerie open source al mondo.

Il motore V8 di Chrome è il runtime di Google, usato anche dal browser Chrome e disponibile anche per Windows, anche se maggiormente usato su Linux. Node.js non usa il classico modello basato su thread e processi ma usa un modello di programmazione "asincrono", basato su eventi che sono gestiti a basso livello dal motore V8, rendendolo performante ed efficiente. Di seguito due esempi che mostrano la principale differenza tra i paradigmi di programmazione sincrona e asincrona:

```
// sincrono
var risultato = ottieni();

// attesa della precedente funzione...
console.log( risultato );
```

```
// asincrono
ottieni( function( dato ) {
  // funzione che verrà eseguita non appena il dato è disponibile
  console.log( dato );
});

// l'esecuzione prosegue subito dopo aver definito la funzione
```

Node.js è *single threaded*, con modello di concorrenza a *event loop*. Il singolo thread è in ascolto alla coda dei messagi che rappresentano porzioni di codice da eseguire. Ad ogni operazione (ad esempio richiesta HTTP) è possibile definire quale funzione dovrà essere eseguita all'ottenimento della risorsa richiesta, riprendendo ad ascoltare la coda dei messaggi. All'ottenimento della risorsa, Node.js si occuperà di eseguire la funzione associata, chiamata *callback*.

2.2.4 Dashboard

E' la homepage della piattaforma, la prima pagina che viene caricata dopo il login. Ha lo scopo di mostrare grafici e statistiche riguardo ai principali dati contenuti nella piattaforma. Per alcuni clienti che richiedono un report mensile riguardo a minacce, attacchi subiti e/o incidenti gestiti, questa sezione viene esportata mediante l'apposita funzione di conversione in PDF, in modo da poterne integrare i grafici nel report. L'intera sezione è personalizzata per ogni cliente, che sceglie quali grafici, viste o contatori mostrare.

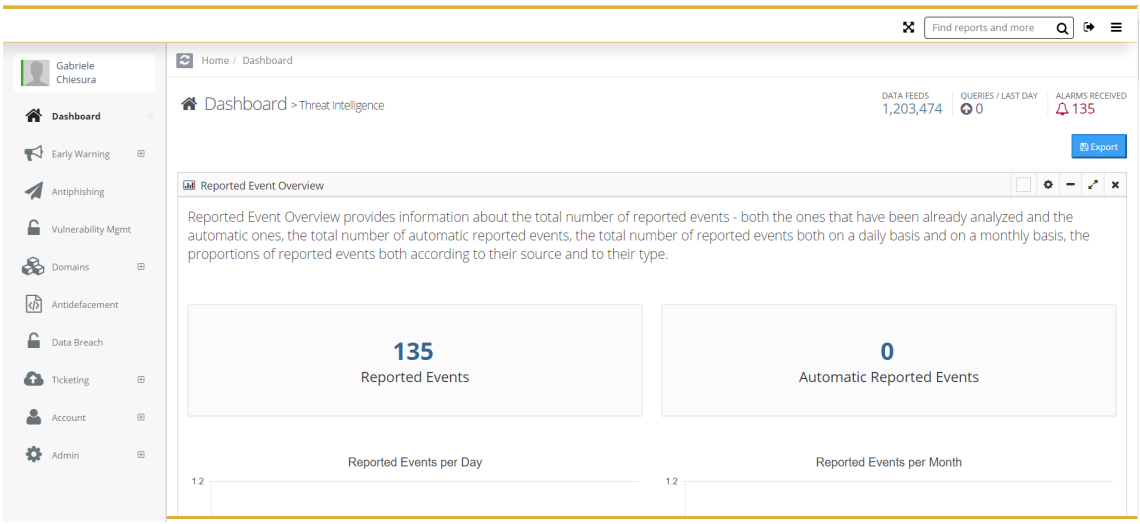

Figura 2.4. Dashboard

Di seguito alcuni esempi di grafici, che mostrano gli allarmi per mese (figura 2.5), le categorie degli incidenti di sicurezza gestiti (figura 2.6), la mappa degli attacchi che coinvolgono botnet (figura 2.7) e il grafo delle relazioni tra indicatori ed oggetti in formato STIX (figura 2.8).

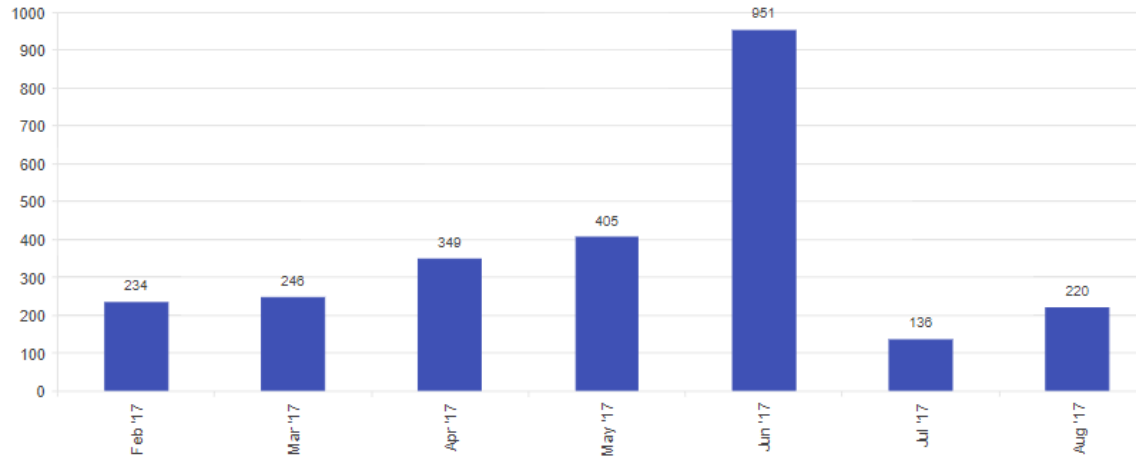

Figura 2.5. Dashboard - Allarmi per mese

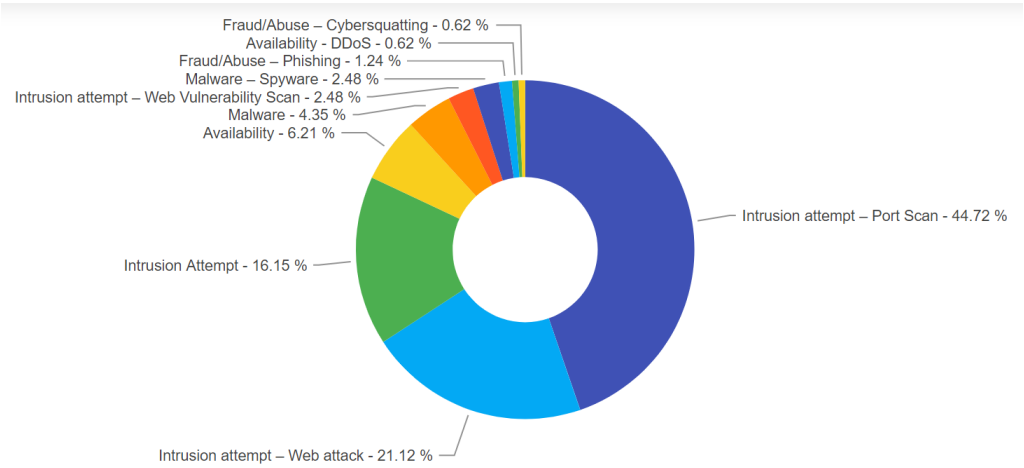

Figura 2.6. Dashboard - Categorie incidenti di sicurezza

2.2.5 Elasticsearch

Descrizione

Elasticsearch[19] è un motore di ricerca ed analisi di dati, distribuito e *RESTful*, che rispetta cioè l'architettura REST (REpresentational State Transfer [3]). Nell'anno 2016 risulta essere il motore di ricerca più utilizzato [20]. Permette di combinare vari tipi di ricerche: strutturate, non strutturate, geografiche, metriche e molto altro. Permette di eseguire aggregazioni sui dati ed estrarre i "migliori" dati, in base ad un punteggio associato alla qualità del match della ricerca. Utilizza complesse strutture dati (indice invertito, KDB-trees) per produrre risultati in maniera efficiente e veloce. Distribuito perchè fornisce ridonzanza, disponibilità, bilanciamento ed affidabilità dei dati grazie alla possibilità di configurare un *cluster* di nodi.

[3] Architettura software basata su un'interfaccia che trasmette dati su HTTP con una struttura ben definita delle URL e delle operazioni

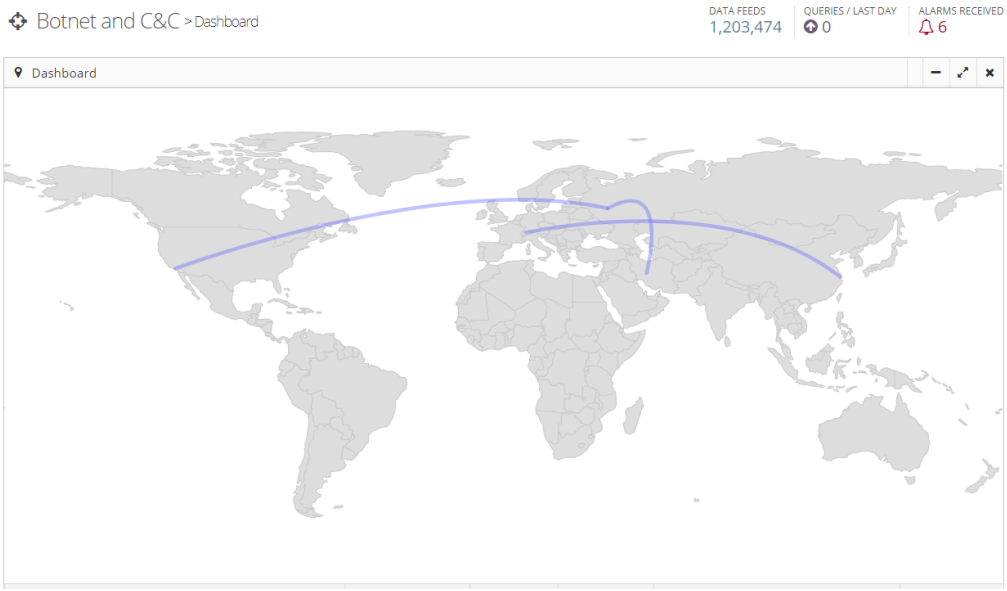

Figura 2.7. Dashboard - Mappa Botnet

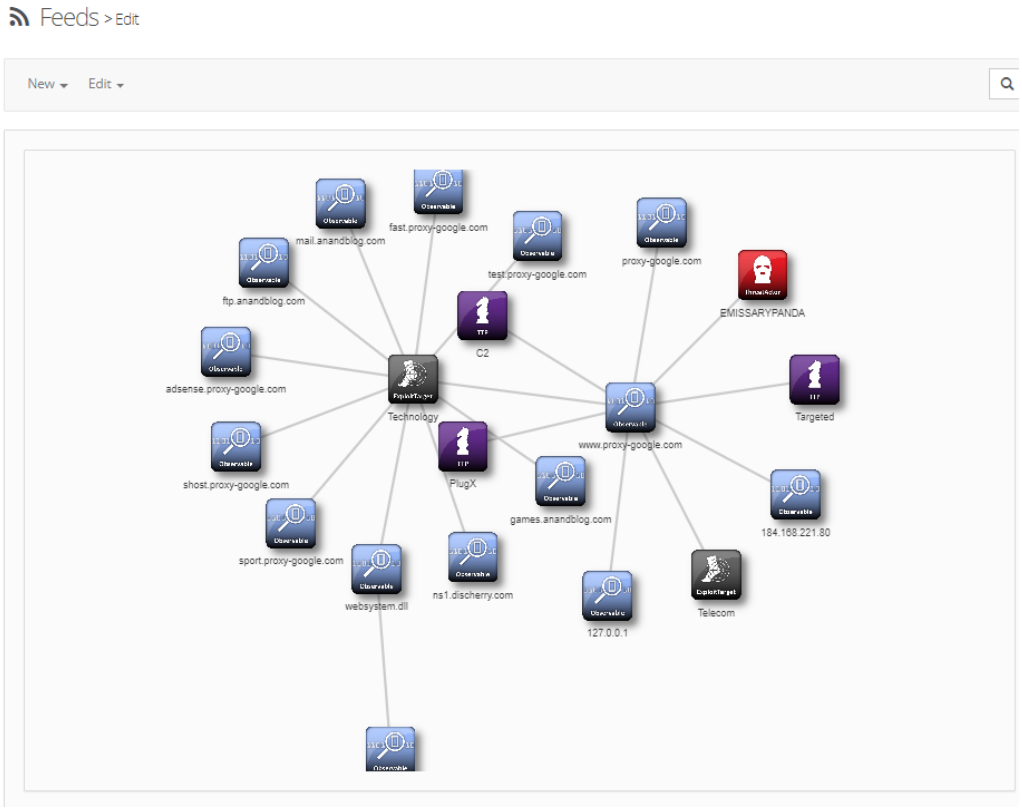

Figura 2.8. Dashboard - Grafo STIX

Struttura

Elasticsearch utilizza le API REST standard. Ogni elemento in Elasticsearch è definito come "documento" ed il suo formato è il *JSON*. Ogni documento è archiviato seguendo una gerarchia a due livelli: *indice* e *tipo*. Ad ogni indice è associato un *mapping*, quello che sarebbe lo *schema* in

Figura 2.9. Elasticsearch

un database relazionale. Espone API per *CURL* [21] e per diversi linguaggi di programmazione: Java, C#, Python, Javascript, PHP, Perl e Ruby. Permette l'aggiunta di estensioni come machine learning o analisi di big data (*Hadoop* [22]).

Utilizzo

La piattaforma ne fa ampio uso in due modi: tramite CURL per gli script bash e tramite le API Javascript, linguaggio del framework Node.js usato nell'engine.

Tramite CURL è possibile eseguire tutte le operazioni possibili, risultando anche molto comodo per estrazioni rapide. Le API CURL permettono infatti di eseguire ricerche (anche complesse) tramite un semplice comando. Per mostrare alcuni esempi, supponiamo che nel nostro indice siano presenti documenti riferiti ai dati di una persona:

```
{
  "nome": "Mario",
  "cognome": "Rossi",
  "data_nascita": "2/3/1965",
  "luogo_nascita": "Torino",
  "laurea": false
}
```

Potremmo per esempio voler ottenere il cognome delle prime 50 persone in ordine alfabetico nate a Torino.

Il comando CURL corrispondente sarebbe:

```
curl "localhost:9200/indice/tipo/_search?size=50&sort=cognome:asc,nome:asc\
&q=luogo_nascita:torino"
```

Anche tramite curl si possono fare ricerche più complesse, effettuando una POST ed la descrizione della ricerca nel corpo della richiesta POST. Supponiamo di voler estrarre il cognome delle prime 50 persone non laureate in ordine alfabetico nate a Torino nell'anno 1965.

```
curl -XPOST localhost:9200/indice/tipo/_search -d'
{
  "query": {
    "query_string": {
      "query": "luogo_nascita:torino AND laurea:true AND
          data_nascita:[1/1/1965 TO 31/12/1965]"
    }
  },
  "sort": [
       {"cognome": {"order": "asc"}},
       {"nome": {"order": "asc"}},
```

```
  ],
  "size": 50
}'
```

Tramite le API Node.js, la precedente ricerca sarebbe:

```
var elasticsearch = require("elasticsearch");
var client = new elasticsearch.Client({
  host: "localhost:9200"
});

client.search(
{
  index: "indice",
  type: "tipo",
  size: 50,
  sort: "cognome:asc,nome:asc",
  q: "luogo_nascita:torino AND laurea:true AND data_nascita:[1/1/1965 TO
      31/12/1965]"
}, function(error, response) {
  var results = response.hits.hits;
  // ...
});
```

2.2.6 Vista allarmi

In questa sezione vengono mostrati gli allarmi di tutti gli altri moduli. Chi li analizza ha la possibilità di arricchirli con ulteriori dettagli, segnalarli come falsi positivi o aggiungere azioni compensative per mitigare la minaccia.

Struttura

Gli allarmi sono divisi in due categorie: i *feeds*, cioè tutti gli allarmi generati dai moduli o provenienti da fonti esterne e i *threats*, cioè feed già analizzati o eventi caricati manualmente che sono stati confermati come minaccia per il cliente e/o la sua infrastruttura.

I campi presenti in ogni segnalazione sono:

- Titolo
- Tipo (ad esempio cyber attack, hacktivism, vulnerability, data breach, cybersquatting, phishing, botnet, darkweb, malware)
- Fonte (ad esempio antidefacement, controllo permutazioni, internal intelligence)
- Dettagli
- Descrizione
- Azioni richieste
- Risorse aggiuntive

E' stato scelto di non poter caricare allegati in questa sezione per forzare la strutturazione delle informazioni (indicatori, azioni compensative e altre informazioni), così da sfruttare al meglio i vantaggi e le ragioni di una TIP. Inoltre se tutti i dati sono inseriti nei rispettivi campi è possibile utilizzare in maniera completa l'apposita tabella presente nella piattaforma (mostrata in

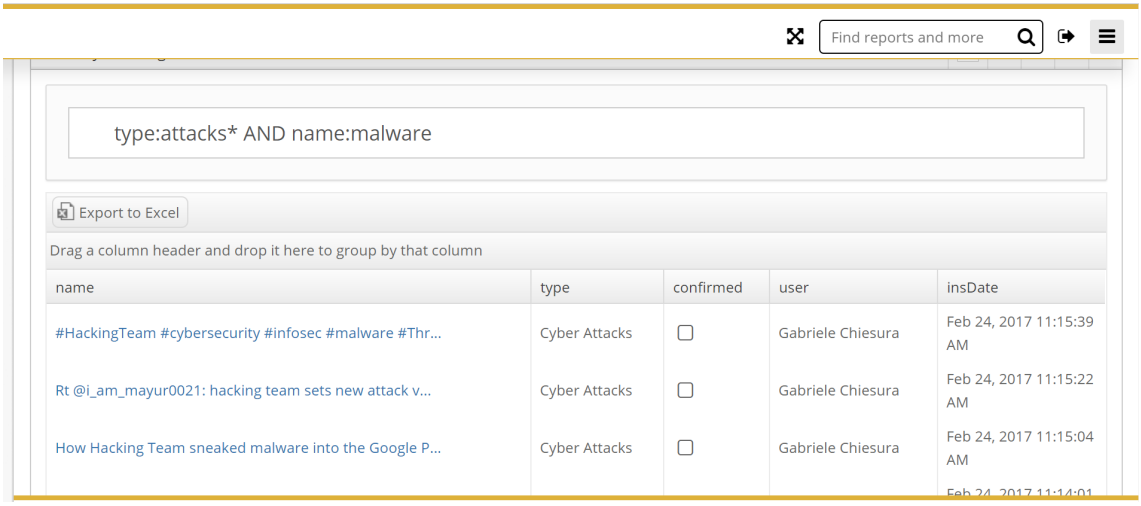

Figura 2.10. Allarmi - Tabella ricerca

figura 2.10), che offre una barra di ricerca dove è possibile ottenere gli elementi archiviati attraverso la stessa sintassi utilizzabile nella stringa di ricerca di una query Elasticsearch. E' possibile esportare i risultati mostrati nella tabella in formato Excel.

I campi dettagli, descrizione, azioni richieste e risorse aggiuntive non sono semplici campi di testo, ma un componente speciale che permette la formattazione del testo (grassetto, italico, font, colori), aggiunta di tabelle, elenchi e immagini, visibile nella figura 2.11.

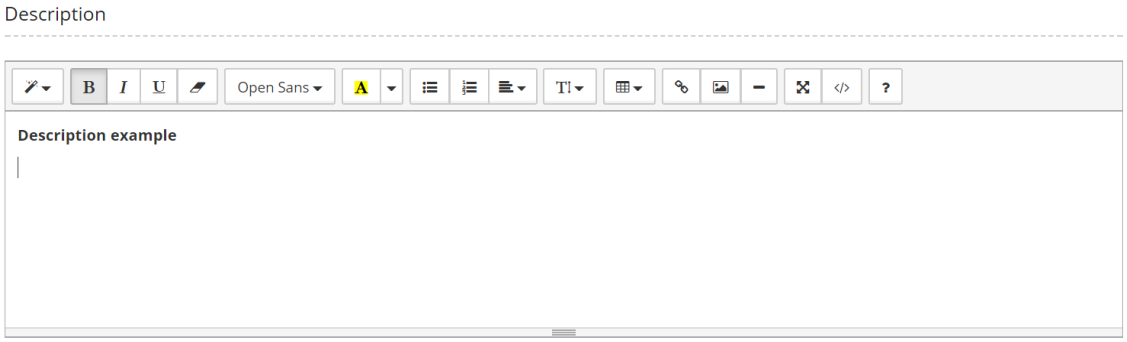

Figura 2.11. Allarmi - Input

Ogni allarme generato viene accompagnato anche da una mail di notifica, inviata ad un'apposita lista di distribuzione a cui sono associati tutti gli analisti che lavorano e monitorano la piattaforma. La mail contiene semplicemente il tipo di allarme ed il cliente in questione, senza mostrarne i dettagli (che potrebbero essere dati sensibili).

2.3 I nuovi moduli

2.3.1 Antidefacement

Il problema

I clienti hanno richiesto un servizio che dovrebbe rilevare, come dice il nome, un attacco di *defacement* (defacciamento), ossia un'operazione che consiste nella modifica del contenuto di una o più pagine di un sito web mediante l'introduzione illecita di testi critici o sarcastici, con motivazioni

che vanno dalla dimostrazione di abilità fino a ragioni ideologiche o di protesta. Questa operazione può essere effettuata sfruttando vulnerabilità del software, sistema operativo o server che gestisce il sito web o con tecniche di social engineering. Subire un attacco di defacement ha impatti negativi sulla reputazione e credibilità della vittima, mostrando la debolezza della propria infrastruttura o dei propri sistemi di sicurezza (si pensi ad esempio al sito di una banca). Per questi motivi è necessario un intervento quanto più tempestivo possibile al fine di ridurre al minimo il periodo di tempo in cui il sito mostra contenuti non legittimi.

Figura 2.12. Esempio di defacement (fonte: [23])

L'analisi

Non risulta disponibile in rete un software di antidefacement pronto e/o semplice da usare o integrare, per cui il team del progetto è subito partito con il design di una nuova soluzione.

Dato che un defacement può essere rilevato con un cambiamento della pagina web, l'idea di base è quella di effettuare il download del suo codice sorgente, controllando che siano presenti solo contenuti legittimi, segnalando eventuali cambiamenti. Un semplice confronto degli screenshot della pagina potrebbe infatti non bastare a rilevare modifiche illegittime.

Siccome sarebbe utile fornire anche lo screenshot della pagina quando verrà mostrato un allarme, abbiamo deciso di integrare uno strumento che permette di ottenere con una sola esecuzione sia il codice sorgente HTML sia lo screenshot, salvandoli in due file distinti. Il modulo dovrà quindi periodicamente scaricare il codice sorgente di ogni pagina monitorata e confrontarlo con una versione "fidata". L'hash sarebbe un'ottima ed efficiente soluzione, se non fosse che le pagine hanno praticamente sempre una parte dinamica che lo farebbe cambiare completamente. Serve quindi una soluzione che permette di confrontare due documenti senza avere una risposta binaria "è uguale" o "è diverso". Ricercando parole chiave tipo "confronto hash documenti", abbiamo trovato un software utilizzabile dalla linea di comando Linux che permette il calcolo di *Context Triggered Piecewise Hash* [24] (CTPH), anche noti come *fuzzy hash*. L'idea di base è dividere l'input in blocchi e calcolarne l'hash per ogni blocco. Se una parte dell'input cambiasse, cambierebbe solo l'hash relativo al blocco in cui era contenuto il cambiamento, lasciando inalterati gli altri. Contando il numero di blocchi uguali si può dedurre la percentuale di similarità.

Il software da linea di comando si chiama *ssdeep* [25] ed è ormai quasi uno standard anche in ambito analisi malware. Permette infatti di confrontare malware o file malevoli per determinare se sono versioni modificate dello stesso. Spesso infatti gli attaccanti usano exploit o malware già usati, cambiandone solo alcuni parametri. Di seguito un esempio di confronto di due file:

```
# calcolo del fuzzy hash del primo file
```

```
ssdeep -b file1 > fuzzy\_file1

# fuzzy\_file1:
# -----------
# ssdeep,1.1--blocksize:hash:hash,filename
# 3072:1NazMtQk07WEzt5uSvOXzGJ9oXVRJ/ddYo5zRxQI4m2CnrezuZiL:
# 1cz1QEVkamFfF91RxQI4m3ezu6,"file1"
# -----------

# calcolo del fuzzy hash del secondo file
ssdeep -b file2

# -----------
# ssdeep,1.1--blocksize:hash:hash,filename
# 3072:1NazMtQk07WEzt5uS5COXzGJ9oXVRJ/ddYo5zRxQI4m2CnrezuZib:
# 1cz1QETCkamFfF91RxQI4m3ezuE,"file2"
# -----------

# confronto del valore precedentemente calcolato con il secondo file
ssdeep -bm fuzzy\_file1 file2

# output:
# file2 matches fh1:file1 (96)
```

L'opzione *-b* viene usata per rimuovere il percorso completo dei file dall'output, lasciando solo il nome del file. L'opzione *-m* invece specifica che il primo parametro è il fuzzy hash di un file, da confrontare con il secondo parametro del comando che rappresenta un altro file.

Siccome bisognerà usare Elasticsearch per archiviare strutture dati delle URL monitorata, useremo il comando *CURL* per eseguire le operazioni. Elasticsearch documenta ed espone infatti API utilizzabili con operazioni HTTP. Di seguito due esempi per inserire un documento in Elasticsearch ed uno per ottenerlo tramite una ricerca:

```
curl -XPUT "localhost:9200/indice/tipo/123" -d'{
  "campo" : "valore"
}'

curl "localhost:9200/indice/_search?size=1&q=campo:valore"
```

Dato che però i documenti sono in formato JSON, servirà un altro software per manipolare tale formato dalla linea di comando. Dopo alcune ricerche abbiamo trovato *JQ* [26]. JQ permette di estrarre attributi e di generare oggetti selezionando da un oggetto JSON i campi desiderati. Di seguito alcuni esempi di utilizzo:

```
# input
# {"foo": 42, "bar": "less interesting data"}

jq '.foo'
# output
42

# input
# {"user":"stedolan","titles":["JQ Primer", "More JQ"]}

jq '{user, title: .titles[]}'
# output
{
  "user": "stedolan",
  "title": "JQ Primer"
```

```
}
{
  "user": "stedolan",
  "title": "More JQ"
}
```

Siccome molti degli strumenti di supporto elencati sono utilizzabili da linea di comando, abbiamo deciso di sviluppare il modulo come uno *script bash*.

2.3.2 Monitoraggio domini

Il monitoraggio domini viene svolto con due moduli separati: *controllo permutazioni* e *scadenza domini*.

Controllo Permutazioni

Il problema

Il modulo di controllo permutazioni si concentra su un attacco di questo tipo: un utente vuole accedere al sito "dominio.it", ma scrive erroneamente "domnio.it", senza una 'i'. Se un attaccante fosse il proprietario di quest'ultimo dominio e ospitasse una pagina web uguale a quella del dominio vittima "dominio.it", potrebbe ottenere password o dati sensibili dalla vittima. Potrebbe inoltre tentare di rubare i cookie di sessione dell'utente o avviare il download di file malevoli. Lo stesso effetto si avrebbe se seguendo dei link l'utente si trovasse nella pagina sbagliata, ma senza accorgersi del dominio presente nella barra di ricerca. Questo genere di attacco è chiamato *typosquatting* (squatting: occupazione abusiva, typo: errore di battitura), ed è un sottoinsieme degli attacchi di tipo *cybersquatting*, i quali consistono nel registrare un dominio che può essere di interesse per una società, al fine di trarne profitto ad esempio vendendolo ad un prezzo più elevato.

L'analisi

Dato un dominio, bisognerebbe quindi calcolare tutti i possibili errori di digitazione che potrebbero prodursi digitandolo oppure calcolare tutti i domini con lettere molto simili a quello originale. In seguito a ricerche sul web con parole chiave "typosquatting" abbiamo trovato uno strumento che esegue proprio questo compito.

E' uno script in linguaggio *python* che, dato un dominio in input, calcola tutte le permutazioni inerenti ad un suo possibile typosquatting ed per ognuna di esse esegue query DNS di tipo A, $AAAA$, MX e NS. Inoltre se le query di tipo A o $AAAA$ hanno successo, effettua il download della pagina HTML e calcola la percentuale di similarità con il comando ssdeep. Le informazioni ottenute dallo script sono riportate nella tabella 2.1. Attraverso un'opzione dello script, è possibile ottenere queste informazioni in formato JSON, comodo per l'integrazione con Node.js.

Ho raccolto inoltre alcune statistiche per alcuni dei principali domini, mostrando la percentuale di domini che possono essere risolti ad un indirizzo IP tra tutti quelli calcolati dallo script. Il risultato è mostrato nella figura 2.13.

I tipi di permutazione *addition*, *insertion*, *omission*, *repetition*, *replacement*, e *transposition* puntano ad un possibile errore di digitazione dell'utente. In generale tutti mirano a trovare un dominio il cui nome cambi il meno possibile dall'originale, così che sia più facile confonderlo.

Ho trovato particolarmente interessante il tipo *bitsquatting*. Al contrario della maggior parte degli altri tipi, non derivano da un errore di battitura, ma risultano dal cambiamento di un bit del codice ASCII di un carattere del dominio. E' un caso che è possibile ottenere grazie ad un errore nella memoria del calcolatore. In un interessante articolo [27] viene descritto come e quanto frequentemente questo succeda. Come riporta l'introduzione:

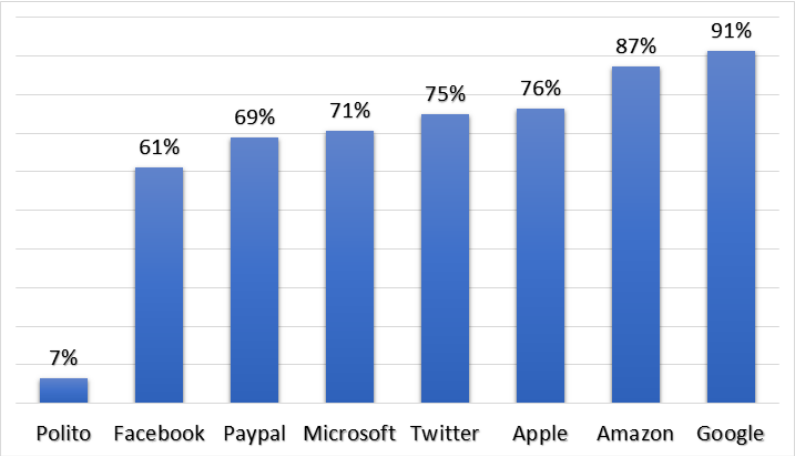

Figura 2.13. Typosquatting domini principali

> Computer hardware, especially RAM, can suffer from random errors that manifest as corruption of one or more bits. The causes of these errors range from manufacturing defects to environmental factors such as cosmic rays and overheating. While the probability of a single error is small, the total error amount in all RAM connected to the Internet is significant. Malicious attackers can exploit these random errors remotely.

Un errore può capitare in molteplici punti: nella memoria RAM, in quella cache o durante la trasmissione dei dati. La causa principale risulta essere il calore. Ad alte temperature infatti i dispositivi subiscono con maggiore frequenza errori di memoria. Questo problema risulta pertanto maggiormente critico per i dispositivi mobili, che devono sopportare condizioni ambientali anche estreme. Un modo comune per indicare il tasso di errori di un dispositivo è in *Failures In Time* (FIT), che indica il numero di errori durante un miliardo di ore di funzionamento. Per una moderna memoria DRAM una stima ottimistica è intorno a 120 FIT (un errore ogni 950 anni) per megabit di memoria. Una comune memoria RAM di 4 GB avrebbe quindi 3 errori al mese, con una stima ottimistica. Ovviamente la grande maggioranza dei dispositivi utilizzano metodi di controllo errori come "checksum" o CRC (Cyclic Redundancy Code), che prevengono il fatto che un dato sia usato se esso è stato alterato, ma ci sono alcuni sistemi che per vari motivi non li implementano o sono molto semplici. Nell'articolo mostrano infatti che in un periodo di sei mesi di prove sperimentali, sono stati registrati 16000 casi di errore.

Scadenza Domini

Il problema

Il modulo scadenza domini si occupa di monitorare la "data di scadenza" per un dominio per evitare un attacco di *cybersquatting* (o domain grabbing). Supponiamo che il proprietario di un dominio si dimentichi di rinnovarlo ed un utente lo compri al posto suo. Quest'ultimo potrebbe cambiare l'associazione <nome DNS, indirizzo IP>, facendo puntare il nome del dominio appena comprato all'IP di un suo server, continuando a ospitare lo stesso sito (o uno molto simile), essendo così in grado di ottenere dati sensibili. Un attacco di cybersquatting può anche essere fatto semplicemente per rendere inutilizzabile un servizio a scopo di danneggiarne l'immagine, oppure per guadagnarci rivendendolo al legittimo proprietario ad un prezzo più alto. Tuttavia non risulta così semplice sfruttare il fatto che il proprietario di un dominio non lo rinnovi in tempo. Il servizio di registrazione dove è stato acquistato solitamente tenta di contattare il proprietario, assicurandosi che sia sua intenzione non rinnovarlo più. Il processo che segue un dominio dalla registrazione alla sua scadenza può essere rappresentato dallo schema in figura 2.14. La scadenza sarebbe comunque da segnalare in quanto i servizi associati a quel dominio smetterebbero di funzionare.

Tipo permutazione	*Permutazione*	*A, AAAA, NS, MX, SSDEEP*
Original*	www.polito.it	130.192.181.193 SSDEEP:100%
Addition	www.politoa.it	-
Addition	www.politob.it	-
Addition	www.politoc.it	-
Addition	www.politop.it	81.88.48.113
Bitsquatting	vww.polito.it	-
Bitsquatting	uww.polito.it	-
Bitsquatting	www.xolito.it	-
Bitsquatting	www.0olito.it	-
Bitsquatting	www.pomito.it	212.227.34.3
Bitsquatting	www.polato.it	51.255.224.245
Bitsquatting	www.polipo.it	46.37.14.18 NS:ns1.publinord.it
Homoglyph	wwvv.polito.it	-
Homoglyph	www.poiito.it	-
Homoglyph	www.polit0.it	-
Homoglyph	www.p0lito.it	-
Homoglyph	www.po1ito.it	-
Hyphenation	www.p-olito.it	-
Hyphenation	www.po-lito.it	-
Hyphenation	www.pol-ito.it	-
Insertion	www.politro.it	-
Insertion	www.po0lito.it	-
Insertion	www.pmolito.it	-
Insertion	www2.polito.it	-
Insertion	www.poloito.it	-
Insertion	wwwq.polito.it	-
Insertion	ww3w.polito.it	-
Insertion	www.polpito.it	31.11.33.60
Insertion	ww2w.polito.it	-
Insertion	www.pollito.it	119.28.76.113
Omission	www.plito.it	72.52.4.119 NS:ns1.sedoparking.com MX:localhost
Omission	www.polit.it	185.53.179.6 NS:ns1.parkingcrew.net MX:mail.h-email.net
Omission	wwwpolito.it	-
Omission	www.polio.it	5.9.152.28
Repetition	wwww.polito.it	-
Replacement	wew.polito.it	-
Replacement	ww2.polito.it	-
Replacement	w3w.polito.it	-
Replacement	www.lolito.it	185.53.179.14 NS:ns1.parkingcrew.net MX:mail.h-email.net
Replacement	www.politi.it	185.53.179.8 NS:ns1.parkingcrew.net
Subdomain	www.po.lito.it	54.72.9.51 NS:ns1.parkingcrew.net MX:mail.h-email.net
Subdomain	www.pol.ito.it	94.141.22.82
Transposition	www.ploito.it	-
Transposition	www.poilto.it	-
Vowel-swap	www.polita.it	151.1.210.41 NS:ns10.weblink.it MX:smtp1.stop-spam.it
Vowel-swap	www.pulito.it	46.37.14.18 NS:ns1.publinord.it
Vowel-swap	www.polite.it	209.15.13.134
Various	www.politoit.it	-
Various	www.polito-it.com	-

Tabella 2.1. Output dello script python

L'analisi

Per questo tipo di monitoraggio è sufficiente sfruttare un servizio comune di tipo "whois". Dato il nome di un dominio, il servizio risponde con un insieme di dati riferiti al proprietario del domino,

Figura 2.14. Processo scadenza dominio

alla data di registrazione e di scadenza. Esistono numerosi servizi online che lo offrono anche attraverso API. Ne ho quindi scelto uno che consente gratuitamente fino a 1000 richieste al mese, dato che l'informazione è statica ed è possibile aggiornarla una volta al mese. Un esempio di risposta del servizio, che restituisce la risposta in formato JSON:

```
{
  "name" : "polito.it",
  "created" : "1996-01-29 00:00:00",
  "changed" : "2017-06-12 15:10:21",
  "expires" : "2018-01-29 00:00:00",
  "status" : "registered",
  "nameservers" : [
    "leonardo.polito.it",
    "ns3.polito.it",
    "giove.polito.it",
    "ns1.garr.net"
  ],
  "contacts" : {
    "owner" : "...",
    "admin" : "..."
  }
}
```

2.3.3 Pastebin

Il problema

Il modulo si occupa di monitorare il paste-site *Pastebin*[28]. E' un'applicazione web che permette agli utenti di inviare frammenti di testo (*paste*), di solito codice sorgente, per la visualizzazione pubblica. Data la natura aperta e libera dei paste e la caratteristica di essere pubblicamente consultabile, si sono verificati svariati casi in cui moduli derivanti da phishing venivano inviati direttamente a dei paste presenti sulla rete. È anche considerata un'usanza comune di abuso dei paste l'elencazione anonima di dati personali quali password di sistema, server privati, o informazioni di accesso ad account.

L'analisi

Pastebin offre le API per avere accesso a tutti i paste pubblici tramite l'acquisto di un account "PRO", da cui è possibile impostare un'indirizzo IP in "whitelist", da cui è possibile effettuare le richieste tramite le API. Di seguito un esempio di risposta:

```
[{
  "scrape_url": "https://pastebin.com/api_scrape_item.php?i=QCft0ta4",
  "full_url": "https://pastebin.com/QCft0ta4",
  "date": "1506701029",
  "key": "QCft0ta4",
  "size": "17",
  "expire": "0",
  "title": "",
  "syntax": "java",
  "user": ""
},
{
  "scrape_url": "https://pastebin.com/api_scrape_item.php?i=vKfPsNaB",
  "full_url": "https://pastebin.com/vKfPsNaB",
  "date": "1506701021",
  "key": "vKfPsNaB",
  "size": "11158",
  "expire": "1506704621",
  "title": "",
  "syntax": "text",
  "user": ""
}]
```


Con la URL presente nel campo "scrape_url" è possibile ottenere il contenuto del paste. Scaricando tutte i paste periodicamente e ricercando all'interno di ciascuno opportune parole chiave è possibile individuare riferimenti al cliente ed eventuali informazioni riservate.

Secondo la documentazione delle API, un buon rapporto tra quantità di documenti da scaricare e la frequenza con cui scaricarli per riuscire ad elaborarli tutti senza perdersene alcuni è 100 documenti al minuto. Ho deciso comunque di svolgere un'analisi per vedere quanti documenti vengono pubblicati effettivamente ogni minuto. Ho avviato un primo prototipo del modulo, il cui compito era solo quello di scrivere in un log file quanti documenti nuovi erano presenti ogni minuto. Questo è stato possibile seguendo il seguente algoritmo, inizializzando una variabile con l'ID del documento più recentemente pubblicato:

1. Effettua il download degli ultimi 100 documenti
2. Scrivi su file il numero di documenti che seguono l'ID del più recente
3. Aggiorna l'ID più recente

Ripetendo questa sequenza di operazioni per alcuni giorni, ho in seguito estratto dal file che in media i nuovi documenti pubblicati in un minuto sono 14, con un picco di 30. Mi è sembrato quindi logico seguire quanto consigliato dalla documentazione ufficiale, lasciando comunque attivare il modulo ogni minuto per essere il più reattivi possibili ad un possibile match. I documenti possono inoltre essere cancellati e al caricamento di un paste è possibile impostare la sua durata, cioè per quanto tempo sarà disponibile pubblicamente. Se l'intervallo di due attivazioni consecutive del modulo fosse maggiore della minima durata che può avere un paste (10 minuti), si rischierebbe di perdere alcuni documenti.

2.3.4 Gestione vulnerabilità

Il problema

La gestione delle vulnerabilità dei propri asset è un aspetto molto importante per la sicurezza dell'infrastruttura di un'azienda. Quasi ogni attacco informatico si basa sullo sfruttamento di una di esse tramite *exploit* (script, documento o sequenza di comandi). Alcune definizioni:

ISO 27005[29] A weakness of an asset or group of assets that can be exploited by one or more threats, where an asset is anything that has value to the organization, its business operations and their continuity, including information resources that support the organization's mission

IETF RFC-2828[30] A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy

NIST[31] A flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy

CNSS[32] Weakness in an IS, system security procedures, internal controls, or implementation that could be exploited

Questo modulo ha quindi l'obiettivo di monitorare le vulnerabilità degli asset del cliente, notificandoli con diversi livelli di priorità a seconda del loro impatto.

L'analisi

Sono presenti online numerosi database open-source delle vulnerabilità note, ad esempio l'*NVD* (National Vulnerability Database) presente sul sito del *NIST* (National Institute of Standards and Technology). Raccoglie tutte le vulnerabilità e configurazioni errate dei principali software, standardizzate e organizzate in modo da facilitare ed automatizzare la loro gestione, l'analisi di sicurezza di un sistema e la sua conformità. Ogni vulnerabilità è identificata da un identificatore *CVE* (Common Vulnerabilities and Exposures)[33], uno standard per l'enumerazione ed identificazione delle vulnerabilità o dei problemi di configurazione, e può riferirsi ad un elenco di *CPE* (Common Platform Enumeration)[34], un formato per specificare prodotti software e piattaforme. Per monitorare le vulnerabilità di un sistema o di un'infrastruttura è quindi necessario ottenere dal cliente la lista dei suoi asset sotto forma di lista di CPE, in modo da controllare se uno di questi è presente in un CVE.

Il modulo scaricherà quindi il database delle vulnerabilità in formato JSON. Di seguito un esempio di possibile database:

```
{
  "cve" : {
    "CVE_data_meta" : {
    "ID" : "CVE-2000-0672"
    },
    "references" : {
      "reference_data" : [ {
        "url" :
            "http://archives.neohapsis.com/archives/bugtraq/2000-07/0309.html"
      }, {
        "url" : "http://www.securityfocus.com/bid/1548"
      }, {
        "url" : "https://exchange.xforce.ibmcloud.com/vulnerabilities/5160"
      } ]
    },
    "description" : {
      "description_data" : [ {
        "lang" : "en",
        "value" : "The default configuration of Jakarta Tomcat does not
            restrict access to the /admin context, which allows remote
            attackers to read arbitrary files by directly calling the
            administrative servlets to add a context for the root directory."
      } ]
```

```
    }
  },
  "configurations" : {
    "nodes" : [ {
      "operator" : "OR",
      "cpe" : [ {
        "vulnerable" : true,
        "cpeMatchString" : "cpe:/a:apache:http_server:3.1",
        "cpe23Uri" : "cpe:2.3:a:apache:http_server:3.1:*:*:*:*:*:*:*"
      }, {
        "vulnerable" : true,
        "cpeMatchString" : "cpe:/a:apache:tomcat:3.0",
        "cpe23Uri" : "cpe:2.3:a:apache:tomcat:3.0:*:*:*:*:*:*:*"
      }, {
        "vulnerable" : true,
        "cpeMatchString" : "cpe:/a:apache:tomcat:3.1",
        "cpe23Uri" : "cpe:2.3:a:apache:tomcat:3.1:*:*:*:*:*:*:*"
      } ]
    } ]
  },
  "impact" : {
    "baseMetricV2" : {
      "cvssV2" : {
        "vectorString" : "(AV:N/AC:L/Au:N/C:P/I:N/A:N)",
        "accessVector" : "NETWORK",
        "accessComplexity" : "LOW",
        "authentication" : "NONE",
        "confidentialityImpact" : "PARTIAL",
        "integrityImpact" : "NONE",
        "availabilityImpact" : "NONE",
        "baseScore" : 5.0
      },
      "severity" : "MEDIUM",
      "exploitabilityScore" : 10.0,
      "impactScore" : 2.9,
      "obtainAllPrivilege" : false,
      "obtainUserPrivilege" : false,
      "obtainOtherPrivilege" : false,
      "userInteractionRequired" : false
    }
  },
  "publishedDate" : "2000-07-20T04:00Z",
  "lastModifiedDate" : "2017-10-10T01:29Z"
}
```


2.3.5 Controllo botnet

Il problema

Una grande parte degli attacchi informatici degli ultimi anni è rappresentata dagli attacchi di tipo DOS (Denyal Of Service[4]). Solitamente gli attaccanti usano una grande rete di calcolatori connessi, chiamata botnet. In questo caso l'attacco prende il nome di DDOS (Distributed DOS). I calcolatori

[4]Attacco che consiste nell'esaurire le risorse di un sistema informatico al fine di impedirgli di erogare il servizio ai suoi utilizzatori

della botnet vengono attivati e prendono ordini da un calcolatore centrale detto di *Command and Control* (C&C o C2). I calcolatori partecipanti alla botnet possono essere sia istanziati dagli attaccanti oppure sono calcolatori di utenti che sono stati infettati da malware o virus che permettono di comunicare con il C&C attraverso il suo nome DNS o direttamente con il suo IP. Il nome DNS non sempre è statico, ma spesso vengono usati algorimi di generazione domini (Domain Generation Algorithms, *DGA*). Attraverso questi algoritmi, i domini di C2 vengono generati dinamicamente, evitando di lasciare nel codice del malware la lista pre-calcolata, che potrebbe essere facilmente essere messa in blacklist sui firewall, negli IDS o negli IPS.

Controllando tutte le query DNS effettuate da una rete, si può controllare se il dominio DNS o l'indirizzo IP sono indicatori di server controllo botnet.

Sebbene l'obiettivo del modulo sia la ricerca di server C2, in automatico siamo in grado di segnalare qualsiasi indicatore di compromissione che passa tra le query e le risposte, non solo quelli riferiti a botnet. I domini contattati possono ad esempio essere riferiti a server di distribuzione di malware, siti che ospitano pagine di phishing o comunque potenzialente pericolosi e associati a minacce.

L'analisi

Siccome la mole di query che riceve il server DNS è elevata (una media di 247 al minuto), abbiamo bisogno di un "log collector". E' un software open-source che accetta in input una certa varietà di log da diverse fonti (web, mobile, IoT e sensori, syslog) in diversi formati e li elabora permettendo di esportarli in vari sistemi (Elasticsearch, MongoDB, Hadoop). In questo modo usiamo una soluzione consolidata per questo tipo di compito. Per ricevere i log abbiamo deciso di usare *syslog*[5], che permette inoltre di configurare regole di archiviazione e di rotazione dei dati ricevuti, in modo da avere uno storico dei log analizzati. Per evitare che il log vengano trasferiti in chiaro, è stata configurata una *VPN* unidirezionale tra il server DNS e la macchina del log collector, configurandola opportunamente attraverso il servizio di cloud-computing utilizzato.

Servirà poi avere una fonte da cui ricevere i più recenti indicatori giornalmente, in modo da confrontarne i domini con i campi delle query DNS. Li otteniamo quindi da una piattaforma di scambio e condivisione di threat intelligence e le sue API, che li raccoglie grazie a vari tipi di fonti. Di seguito un esempio di indicatori prelevati con le API:


```
[{
  "indicator": "phohww11888.org",
  "description": "",
  "created": "2017-10-06T23:19:24",
  "title": "",
  "content": "",
  "type": "domain",
  "id": 103738579
},
{
  "indicator":
      "0e4763d4f9687cb88f198af8cfce4bfb7148b5b7ca6dc02061b0baff253eea12",
  "description": "",
  "created": "2017-10-06T23:19:24",
  "title": "",
  "content": "",
  "type": "FileHash-SHA256",
  "id": 103738580
}]
```


[5]Protocollo di rete usato per trasferire informazioni di log

Capitolo 3

Progettazione

3.1 Antidefacement

Scaricando il codice sorgente di ogni pagina monitorata siamo in grado di evidenziare ogni introduzione illegittima di codice, come ad esempio link a pagine esterne malevole o di pubblicità o codice di Cross-Site Scripting di tipo persistente[1].

3.1.1 Inizializzazione

Viene eseguito uno script per ogni cliente, passando il suo identificatore come parametro, così da comporre l'indice Elasticsearch relativo nel seguente modo: "indice-$cliente", dove nella variabile "cliente" viene salvato il paramentro passato allo script. Lo script viene lanciato con il comando *PM2*.

```
pm2 start antidefacement.sh cliente
```

La lista delle URL monitorate per cliente viene precaricata nell'opportuno indice attraverso un altro script che accetta in input il file contenente la lista delle URL. Per ognuna di esse, esegue il seguente comando per salvarle in Elasticsearch:

```
curl -XPOST "localhost:9200/antidefacement-$cliente/list" -d'{
  "url" : "https://www.polito.it",
  "status" : "uninitialized"
}'
```

3.1.2 Il diagramma a stati

Ogni pagina web monitorata dal servizio segue il diagramma a stati mostrato in figura 3.1. Le frecce rosse solo le uniche transizioni di stato che richiedono l'intervento manuale di un'analista, mentre le altre sono automatiche. Lo scheletro dello script può essere semplificato in questo modo:

```
urls=$(get_urls)

while true; do
  for url in "${urls[@]}"; do
```

[1] Vulnerabilità di applicazioni web che non effettuano opportunamente operazioni di escaping e sanitizzazione dell'input, permettendo ad un utente di salvare del codice nel server dell'applicazione, che poi sarà eseguita nel browser degli utenti che accederanno a quel contenuto, permettendo all'attaccante di ottenere ad esempio cookie di sessione

```
    status=$(get_status "$url")

    if [[ "$status" == "uninitialized" ]]; then
      uninitialized "$url"
    elif [[ "$status" == "learning" ]]; then
      learning "$url"
    elif [[ "$status" == "running" ]]; then
      running "$url"
    elif [[ "$status" == "falsepositive" ]]; then
      falsepositive "$url"
    elif [[ "$status" == "falsewarning" ]]; then
      falsewarning "$url"
    elif [[ "$status" == "incident" ]]; then
      incident "$url"
    fi

  done
done
```

Come si può vedere, il monitoraggio è continuo (*while true*), passando da una URL all'altra continuamente.

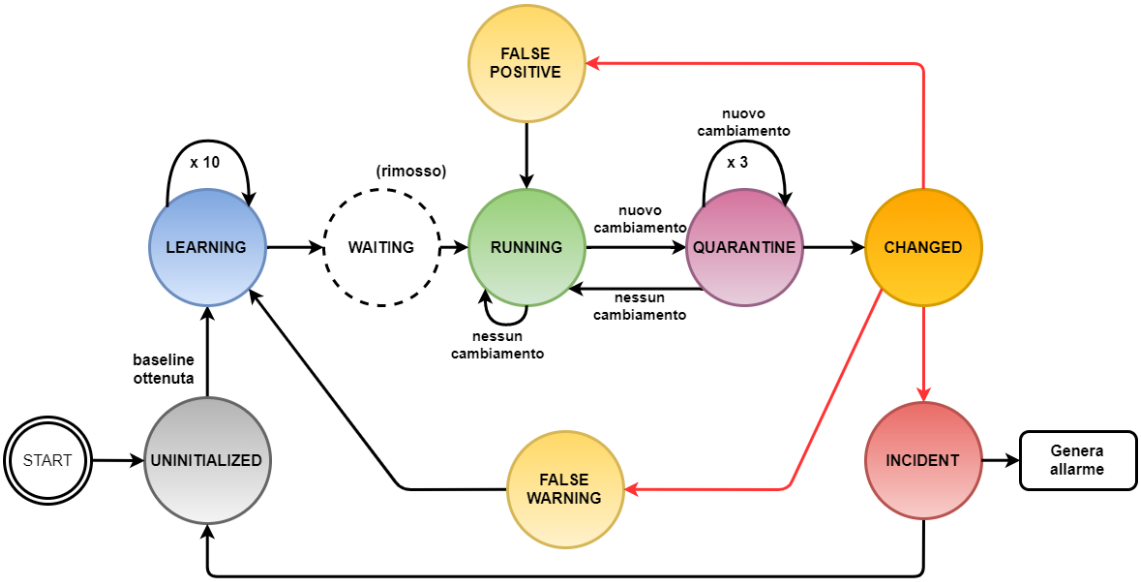

Figura 3.1. Antidefacement - Diagramma a stati

La funzione *get_urls* si occupa di estrarre dall'apposito indice di Elasticsearch tutti gli ID delle URL monitorate. Principalmente svolge la seguente sequenza di comandi:

```
curl 'localhost:9200/antidefacement-$cliente/list/_search?size=1000&fields' |
    jq .hits.hits[]._id
```

Il parametro *fields* evita di scaricare tutto il contenuto dei documenti, ma solo indice, tipo e ID, mentre l'ultimo comando *jq* permette di estrarre dal JSON di risposta di Elasticsearch un vettore con gli ID delle URL monitorate. Una volta ottenuta questa lista, lo script inizia ad iterare su di essa, e per ogni ID ottiene lo stato della URL relativa attraverso la funzione *get_status*, che principalmente esegue la seguente sequenza di comandi:

```
curl localhost:9200/antidefacement-$cliente/list/$url?stored_fields=status |
    jq .fields.status
```

Avendo ottenuto lo stato della URL considerata, con un semplice confronto viene eseguita la funzione corrispondente.

Per i passaggi di stato, viene utilizzato il seguente comando di update:

```
curl -XPOST localhost:9200/antidefacement-$cliente/list/$url/_update -d
    '{"doc" : { "status" : "nuovo stato"}}'
```

Uninitialized

Lo stato iniziale di ogni URL. Questa è una fase critica, in quanto è quella in cui viene effettuato il download della *baseline*, cioè il modello, la versione legittima. Bisogna assicurarsi che la versione della pagina ottenuta sia quella ufficiale. Eventuali errori in questa fase produrrebbero un falso allarme. Nel dettaglio, gli elementi ottenuti in questa fase sono:

- HTML
- Screenshot
- Fuzzy hash del codice HTML scaricato precedentemente

HTML e screenshot vengono salvati in due file separati dallo script ausiliario, mentre il fuzzy hash viene calcolato nel seguente modo:

```
current_fuzzy_hash=$(ssdeep -b current_url.html)
```

Un nuovo documento viene salvato con questi dati nel database, impostando lo stato di questa URL in *learning*.

Learning

E' stata introdotta per studiare la pagina monitorata ed al tempo stesso limitare la generazione di falsi allarmi. Una pagina può presentarsi diversa anche tra due download effettuati ad una breve distanza temporale. Si pensi ad esempio ad una pagina con elementi dinamici, annunci pubblicitari che cambiano ad ogni ricaricamento della pagina, e così via. Un parametro determina quante volte ripetere questa fase, al momento impostato a 10. Ad ogni giro, viene scaricato il codice sorgente della pagina e vengono estratte le informazioni relative a *Stringhe Tipiche*, *Link* e *Differenza Fuzzy Hash*.

In un attacco di defacement sono presenti nella maggior parte dei casi stringhe tipiche dell'attacco, quali *own*, *hack*, *owned*, *0wn*, *squad*, *h4ck* e simili. Abbiamo raccolto un dizionario che comprende tutte le parole di questo genere, in italiano, inglese, ma anche russo, cinese e polacco. Se durante il monitoraggio di una pagina dovesse comparire una di queste stringhe, potrebbe essere un buon indicatore di un potenziale attacco di defacement. Vengono salviate quindi tutte le stringhe tipiche già presenti nella pagina, con le loro occorrenze, così da rilevare ogni stringa tipica aggiunta (oltre a quelle già esistenti) alla pagina. Le stringhe vengono estratte grazie all'esecuzione dei seguenti comandi:

```
grep -iw "$typical_string" html_content
grep -i "$typical_string" html_content
```

L'opzione *-i* permette di ignorare lettere maiuscole o minuscole, mentre l'opzione *-w* forza il match con una parola intera.

Riguardo ai link della pagina, vengono estratti tutti i domini contenuti in essi, così da segnalare ogni eventuale dominio aggiuntivo e/o non legittimo inserito dopo questa fase. I link vengono estratti grazie alla seguente espressione regolare con il comando grep:

```
(http|ftp|https|//):?(//)?([a-zA-Z0-9./?=_-])+\.([a-zA-Z0-9./?=_-])+
```

Viene poi estratto il dominio del link: se ad esempio nella pagina è presente il link

http://dominioesempio.it/folder/esempio.pdf,

verrà estratto il dominio *dominioesempio.it*.

Viene infine calcolato il fuzzy hash della pagina scaricata e viene comparato con quello ottenuto nella prima fase di "uninitialized", ottenendo così la differenza percentuale dalla versione di baseline. Viene usato il comando *ssdeep* nel seguente modo:

```
similarity_percentage=$(ssdeep -bm <(printf "\%s" "$baseline_fuzzy_hash")
    current_page.html)
diff_percentage=$(( 100 - $similarity_percentage ))
```

Se la pagina è dinamica, ci si aspetta che ad ogni download possa differire dalla prima versione scaricata. Questo valore verrà usato per determinare la soglia con cui far scattare un allarme.

Al termine dell'ultimo giro di learning, per la URL sono salvati quindi l'unione di tutti i link e stringhe tipiche trovate, insieme alla sequenza di differenze percentuali fuzzy hash. Viene scelto come soglia di differenza fuzzy hash il massimo valore trovato dopo i giri di learning. L'idea è che se c'è una versione della pagina che ha prodotto una differenza dalla baseline di un valore x, allora dobbiamo ammettere che durante la fase di monitoraggio possa succedere che si trovi una differenza al più del valore x. Dopo aver ottenuto questa soglia, la URL passa allo stato di *running*, inserendo in Elasticsearch tutte le informazioni ricavate attraverso il seguente comando:

```
curl -XPUT localhost:9200/antidefacement-$cliente/html/$current_id -d'{
  "baseline_fuzzy_hash" : "$baseline_fuzzy_hash",
  "html" : "$baseline_html",
  "screenshot" : "$screenshot",
  "strings" : "$strings",
  "strings_partial" : "$strings_partial",
  "diff_history" : "$diff_history",
  "max_diff" : "$max_diff",
  "links" : "$links"
}'
```

Siccome a volte il learning non produce una baseline accettabile (soglia di differenza 100% che non farebbe mai scattare un allarme), abbiamo introdotto nella piattaforma una sezione "Antidefacement", dove vedere tutte le URL monitorate e la loro baseline. All'interno di questa sezione è possibile far ripetere la fase di learning ad una URL attraverso un apposito pulsante.

Waiting

La fase di waiting non è più presente nella versione attuale del modulo. Era stata introdotta perchè una volta che una URL aveva finito la fase di learning, avevamo bisogno di controllare quali link e stringhe tipiche avesse trovato e quale valore di soglia di differenza avesse calcolato, in modo tale da "approvare" i parametri trovati e confermare il passaggio alla fase di monitoraggio. Questo però rallentava di molto l'ingresso di una URL alla fase di running (la fase di learning poteva ad esempio finire di notte), lasciandola scoperta ad un potenziale defacement. Abbiamo deciso quindi di rimuovere questa fase, facendo saltare una URL direttamente dal learning al running, notificando però con una mail i parametri calcolati e le informazioni per decidere se lasciare la URL in running oppure fermarla e rimandarla alla fase di learning. Nella mail sono quindi presenti:

URL La URL in oggetto

Link La lista dei link trovati nella pagina

Sequenza differenze fuzzy hash La sequenza delle differenze fuzzy hash calcolate durante i giri di learning

Soglia differenza fuzzy hash Il massimo valore trovato nella sequenza (la soglia)

Stringhe tipiche Le stringhe tipiche e le loro occorrenze

Contesto stringhe tipiche La porzione del codice HTML in cui è presente ciascuna stringa tipica trovata (così da evitare di dover aprire manualmente il codice sorgente e cercare ogni stringa)

Il contesto delle stringhe tipiche è l'unico dato che non è salvato in Elasticsearch, ma viene calcolato al momento dell'invio della mail attraverso il seguente comando:

```
context=$(grep -i -c 2 "$stringa")
```

L'opzione *-c 2* permette di estrarre le 2 righe antecedenti e seguenti alla riga dove si è trovato un match.

Running

La fase principale, quella in cui una URL viene effettivamente monitorata dopo averla analizzata. Esegue lo stesso codice di un giro della fase di learning, estraendo le stesse informazioni. Se sono presenti nuovi link, nuove stringhe o una differenza maggiore alla massima trovata nella fase di learning, prima di notificare un allarme entra in gioco il concetto di *quarantena*. Nelle fasi di sviluppo del modulo abbiamo notato che basta un errore temporaneo per far generare un allarme. Abbiamo così introdotto la quarantena: per generare un allarme per una URL, in essa lo script deve trovare un cambiamento per tre volte consecutive. Questo limita di molto la generazione di falsi positivi.

Se la quarantena supera i tre giri, vengono eseguiti i seguenti confronti per determinare con quale priorità notificare un allarme:

1. Priorità alta se sono state trovate stringhe tipiche aggiuntive o se la differenza fuzzy hash dalla baseline è superiore all'80%
2. Priorità media se la differenza fuzzy hash dalla baseline è superiore al 40%
3. Priorità bassa se la differenza fuzzy hash dalla baseline è superiore alla soglia, se sono state trovate stringhe tipiche aggiuntive con un match parziale (la stringa è contenuta in un'altra parola) o se sono stati trovati link aggiuntivi

Viene così generato un allarme, visualizzabile nell'apposita sezione della piattaforma, contenente i dettagli riguardo la causa che lo ha fatto scattare (stringhe tipiche, link, differenza fuzzy hash), l'output del comando Linux *diff* tra il codice sorgente di baseline e quello corrente, lo screenshot della pagina di baseline e di quella corrente. Nel caso di un allarme, la URL passa allo stato *changed*, altrimenti rimane nello stato running.

Changed

Nella piattaforma è visibile l'allarme relativo, dove è possibile analizzarlo. Lo script non esegue nulla, attendendo l'interazione sulla piattaforma. Dopo l'analisi è possibile etichettarlo in uno dei seguenti modi, inviando la URL nel relativo stato:

False Positive

C'è stato un errore temporaneo nel download della pagina (HTTP Code 503 - Service Unavailable), che ha prodotto di conseguenza un cambiamento consistente del codice sorgente, generando un'alta differenza fuzzy hash, oppure ci sono stati aggiornamenti di elementi dinamici che non sono un rilascio di codice vero e proprio ma che hanno comunque prodotto una differenza in grado di far superare la soglia fuzzy hash (cambiamento di ID, dimensioni di elementi HTML, sezioni pubblicitarie). La url passa direttamente allo stato di *running*, ignorando il cambiamento trovato e riprendendo il monitoraggio con gli stessi parametri.

False Warning

La pagina è cambiata legittimamente, ad esempio dopo il rilascio di un aggiornamento o di una nuova versione. La url torna allo stato di *learning* per scaricare una nuova baseline, visto che è cambiata definitivamente.

Incident

Si è verificato un defacement o sono stati introdotti contenuti non legittimi nella pagina. L'allarme viene spostato dalla sezione allarmi alla sezione degli incidenti, dove il nostro team di analisti o il SOC del cliente possono analizzarlo, arricchirlo di informazioni e soprattutto di azioni compensative che il cliente deve attuare per mitigare gli effetti dell'attacco. Non appena l'incidente è stato risolto, la URL passa allo stato *uninitialized* per riprendere ad essere monitorata.

3.2 Controllo permutazioni

3.2.1 Inizializzazione

Tutti i domini di cui si vogliono monitorare i relativi typosquatting devono essere inseriti in un opportuno indice Elasticsearch, uno per cliente: domini-cliente/lista. Inoltre sempre nello stesso indice ma diverso tipo vengono inseriti le permutazioni già di proprietà del cliente, già comprate per gestire questo tipo di attacco oppure già analizzati come legittimi: domini-cliente/legittimi.

3.2.2 Esecuzione

Per ogni dominio da monitorare contenuto nella lista "domini-cliente/lista" viene lanciato lo script python dal modulo che ho sviluppato e ne viene analizzato l'output. Una permutazione viene ignorata se appartiene già al cliente oppure se è già stato analizzato come dominio legittimo (appartenente quindi alla lista "domini-cliente/legittimi"). Negli altri casi, un allarme è generato se rispetta tutte queste condizioni, mostrate graficamente attraverso un diagramma di flusso anche in figura 3.2:

1. Alla permutazione è associato almeno un campo rilevante (che mostri che il dominio ha una certa attività), cioè un record *MX*, *NS* o, più importante, *A* o *AAAA* (cioè che sia associato ad un indirizzo IP)

2. Se è già stata segnalata la stessa permutazione ed è cambiato uno dei record (ad esempio è stato aggiunto un mail server). In questo caso la priorità dell'allarme è *bassa* se non è presente un valore di fuzzy hash, o è sotto la soglia. Se è presente ed è sopra la soglia, la priorità è alta

3. Se non è mai stata segnalata quella permutazione

Viene quindi creato un allarme contenente tutti i record DNS e l'eventuale similarità fuzzy hash associati alla permutazione analizzata. Se la permutazione era già stata segnalata e non ancora analizzata, il modulo aggiornerà l'allarme precedente con la sitazione attuale, avendo usato come ID in Elasticsearch l'hash SHA1 della permutazione, calcolato con la libreria standard Javascript *Crypto*:

```
var crypto = require("crypto");
var id = crypto.createHash("sha1").update(permutazione).digest("hex");
```

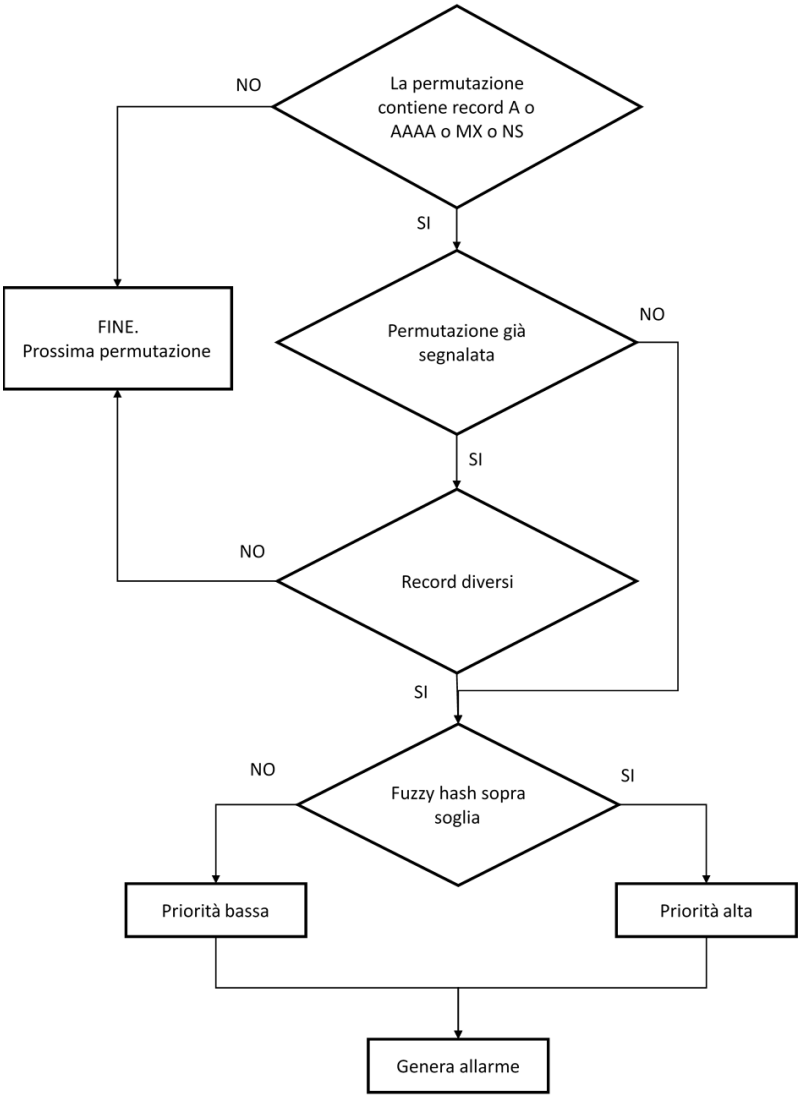

Figura 3.2. Permutazioni - Diagramma di flusso

3.3 Scadenza domini

3.3.1 Inizializzazione

Tutti i domini di cui si voglia monitorare la scadenza devono essere inseriti in un opportuno indice Elasticsearch, uno per cliente: domini-cliente/lista.

3.3.2 Esecuzione

Questo servizio è un modulo Node.js dell'engine. E' schedulato per attivarsi ogni lunedì. Per ogni cliente e per ogni dominio da monitorare, effettua una chiamata alla URL del servizio "whois" estraendo dalla risposta il campo *expires*, riferito alla scadenza del dominio. Notifica quindi con priorità bassa quando mancano meno di due mesi alla scadenza e con priorità alta quando manca meno di un mese.

Per gestire le date ho usato un modulo di Node.js chiamato *moment*, che permette elaborazione e confronto di oggetti di tipo ora e data in modo molto comodo. Ad esempio, il confronto precedente è implementato così:

```
// richiesta modulo
var moment = require("moment");

// creazione oggetto moment per scadenza dominio
var scadenza = moment(expires);

// creazione oggetti moment per un mese e due mesi
var fra1mese = moment().add(1, "months").endOf("day");
var fra2mesi = moment().add(2, "months").endOf("day");

var priorita;
if ( scadenza.isBefore(fra2mesi) ) {
  if ( scadenza.isBefore(fra1mese) )
    priorita = "alta";
  else
    priorita = "bassa";
}

if ( priorita ) {
       // segnala allarme scadenza
       // ...
}
```

Secondo questo algoritmo, il modulo continuerà a segnalare ogni lunedì anche dopo la data di scadenza, fino a quando non venga rinnovato. Questo è voluto dato che, come descritto nella sezione 2.3.2, anche dopo la scadenza si hanno ancora diversi giorni per poter rinnovare il dominio.

3.4 Pastebin

3.4.1 Inizializzazione

E' necessario creare un indice in Elasticsearch per ogni cliente per cui è attivo il servizio, indicizzando un documento contenente tutte le espressioni regolari da usare con il seguente formato:

```
{
  "domains": ["esempio\.it", "dominio\.(it|com)"],
  "mails": ["@mail\.it", "@mail\.server\.com"],
  "ip": ["130\.192\.\d{1,3}\.\d{1,3}"]
}
```

E' inoltre necessario salvare in un altro indice le espressioni regolari di tipo "hacktivism", seguendo il seguente formato:

```
{
  "patterns": ["[^\w\d]hack[^\w\d]|^hack[^\w\d]|^hack$|[^\w\d]hack$",
               "[^\w\d]dump[^\w\d]|^dump[^\w\d]|^dump$|[^\w\d]dump$"]
}
```

3.4.2 Esecuzione

Il modulo è uno script Node.js che scarica quindi i paste pubblici alla ricerca di match con determinate *espressioni regolari* (mail di un certo dominio, parole chiave, carte di credito per una banca, indirizzi IP ed altre), che per brevità chiameremo *regex*.

Come già descritto nella sezione 2.3.3 esegue ogni minuto il download degli ultimi 100 paste pubblicati. Per il cliente che usa questo servizio ho configurato e personalizzato in totale 309 regex,

divise per tipo come mostrato in figura 3.3. La categoria "hacktivism" è una lista di regex comune a tutti i clienti che usano il servizio e comprende expressioni come *hack*, *dos*, *own* o *#op*. In seguito il dettaglio di come questa categoria viene usata.

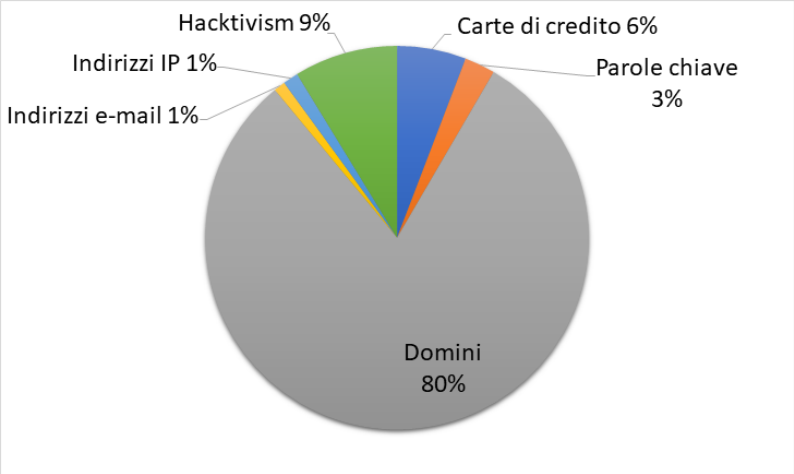

Figura 3.3. Pastebin - Tipi di espressioni regolari

Le regex che ho scritto hanno seguito un processo di raffinamento che è durato alcune settimane, in modo da evitare il maggior numero di falsi positivi, ma allo stesso tempo in modo da perdere meno match possibili. Supponiamo ad esempio di voler trovare la corrispondenza con i numeri di carte di credito che iniziano con 1234-56**-****-****. Ho scritto una regex che trovi solo le sequenze di 4 gruppi di numeri in cui il primo gruppo sia uguale alla sequenza dei 4 numeri che lo compongono ed il secondo inizi con i primi due numeri del secondo gruppo ricercato, separati (o meno) da un qualsiasi numero di caratteri di spaziatura o trattini. Inoltre i 4 gruppi di numeri devono essere seguiti e/o preceduti da un carattere NON numerico. Il risultato è il seguente:

```
[^\d]1234[\s\-]*56\d{2}[\s\-]*\d{4}[\s\-]*\d{4}[^\d]
```

Stessa cosa vale per le regex riferite all'acronimo del cliente. Se il cliente fosse ad esempio "Politecnico Di Torino", l'espressione regolare per l'acronimo sarebbe:

```
^pdt[^\w\d]|[^\w\d]pdt[^\w\d]|[^\w\d]pdt$
```

Brevemente, significa che l'acronimo non deve essere preceduto e/o seguito da un carattere alfanumerico (può quindi essere presente solo un simbolo o un carattere di spaziatura).

La porzione di codice che esegue quanto descritto può essere riassunta come segue:

```
var request = require("request");

request({
  url: "https://pastebin.com/api_scraping.php?limit=10"
  }, function(err, resp, body) {
    var pastes = JSON.parse(body);

    pastes.forEach(function(p) {
      request({
        url: p.scrape_url
      }, function(err, resp, body) {
        var paste = JSON.parse(body);

        espressioniregolari.forEach(function( regex ){
          var result = paste.match(new RegExp(regex, "ig"));
            if (result) {
```

```
            // genera allarme
            // ...
          }
        });
      });
    });
});
```

In fase di test, ho notato che in presenza di paste di grosse dimensioni il modulo si bloccava, o comunque risultava molto lento. In alcuni casi rimaneva bloccato per più di un minuto, facendo perdere il seguente "risveglio". Ho quindi deciso di dividere i paste in due gruppi in base alla loro dimensione. In seguito ad un'analisi durata una settimana in cui facevo scrivere su un apposito log file la dimensione di ogni paste scaricato, ho ottenuto che la dimensione media dei paste era di circa 30 KB. Mantenendo una certa larghezza nella soglia, ho quindi deciso che i paste di dimensione superiore a 50 KB venissero filtrati e spostati in una seconda lista. Questa lista viene passata come input ad un altro script Node.js, slegato dal processo principale dell'engine. Questo script è una versione leggermente modificata del modulo, dove la differenza principale sta nel fatto che ogni regex non ha come input il paste completo, ma una porzione. Ogni paste infatti verrà diviso in blocchi da 50 KB e verrà eseguita la ricerca della regex su ogni blocco. Questa divisione al momento è fissa sulla dimensione, quindi potenzialmente potrebbe "spezzare" il paste proprio in corrispondenza di un possibile match. Ho deciso di mantenere questa soluzione per non aggiungere elaborazione aggiuntiva nel calcolo della divisione in blocchi dei paste.

Il codice che rappresenta la divisione in blocchi dei paste di grandi dimensioni è il seguente:

```
largepastes.forEach(function(p) {
  request({
    url: p.scrape_url
  }, function(err, resp, body) {
    var paste = JSON.parse(body);

    var chunkSize = 50*1024;
    // 50 KB, supponendo che la maggior parte dei caratteri siano ASCII,
    // quindi rappresentati in un byte

    var chunks = body.match(new RegExp("(.|[\\r\\n]){1,"+chunkSize+"}","g"));
    chunks.forEach(function( chunk ){
      espressioniregolari.forEach(function( regex ){
        var result = chunk.match(new RegExp(regex, "ig"));
        if (result) {
            // genera allarme
            // ...
        }
      });
    });
  });
});
```

Ogni paste scaricato segue il seguente flusso per l'eventuale generazione di un allarme, mostrato anche nella figura 3.4:

1. Per ogni regex (tranne quelle di tipo hacktivism) che ha un match nel paste viene salvato in una lista un oggetto che contiene la regex ed il match nel documento. Se la lista è vuota non viene generato nessun allarme e si passa al prossimo paste.

2. Per ogni regex di tipo hacktivism che ha un match nel paste viene salvato in un'altra lista un oggetto dello stesso tipo del precedente. Se la lista è vuota il warning generato sarà di tipo *Reputazione*, altrimenti sarà di tipo *Hacktivism*

3. Ogni occorrenza trovata dalle regex nel testo viene sostituita con un tag HTML che la evidenzia in giallo, così da facilitare l'analisi dell'allarme. Nel caso di un paste grande, viene estratto da esso la porzione in cui compare la corrispondenza con una regex (vengono presi 300 caratteri prima e dopo la corrispondenza). Questo per evitare che il disco del server si saturi e che le prestazioni di ricerca di Elasticsearch si degradino inutilmente indicizzando documenti pastebin
4. Viene creato un nuovo allarme, dove nel campo *Dettagli* viene messo il risultato delle precedenti fasi (il paste o le sue porzioni con le corrispondenze evidenziate, le regex che hanno avuto un match, il link del paste sul sito ufficale)

Nel caso di paste grandi, oltre che estrarre solamente il contesto del match, viene anche salvata la prima riga del documento, in quanto in caso di dump è lì che viene specificata la sintassi del documento.

Il contenuto del paste eventualmente segnalato in un allarme viene mostrato in esso come stringa e non come codice, altrimenti sarebbe possibile scrivere un paste con del codice javascript che verrebbe eseguito al momento di visualizzare l'allarme relativo, rendendo la piattaforma vulnerabile ad attacchi di tipo cross-site scripting persistente.

3.5 Gestione vulnerabilità

Il servizio è composto da due moduli: uno per l'ottenimento del database delle vulnerabilità, l'altro per confrontare gli asset del cliente con il database.

3.5.1 Inizializzazione

E' necessario salvare in un indice Elasticsearch per ogni cliente la lista dei cpe da monitorare, nel seguente formato:

```
{
  "cpe": "cpe:/a:apache:http_server:2.2.25"
}
```

3.5.2 Download database

Un modulo dell'engine si occupa di scaricare giornalmente il database delle vulnerabilità, salvando i dati in un apposito indice di Elastisearch.

La porzione di codice può essere così riassunta:

```
var request = require("request");

request({
  url: "https://url.to.database/file.json"
  }, function(err, resp, body) {

    var result = JSON.parse(body);

    result.forEach(function(elem) {
      elastic.index(
      {
        index: "cve",
        type: "cve",
        id: elem.CVE_ID,
        body: elem
```

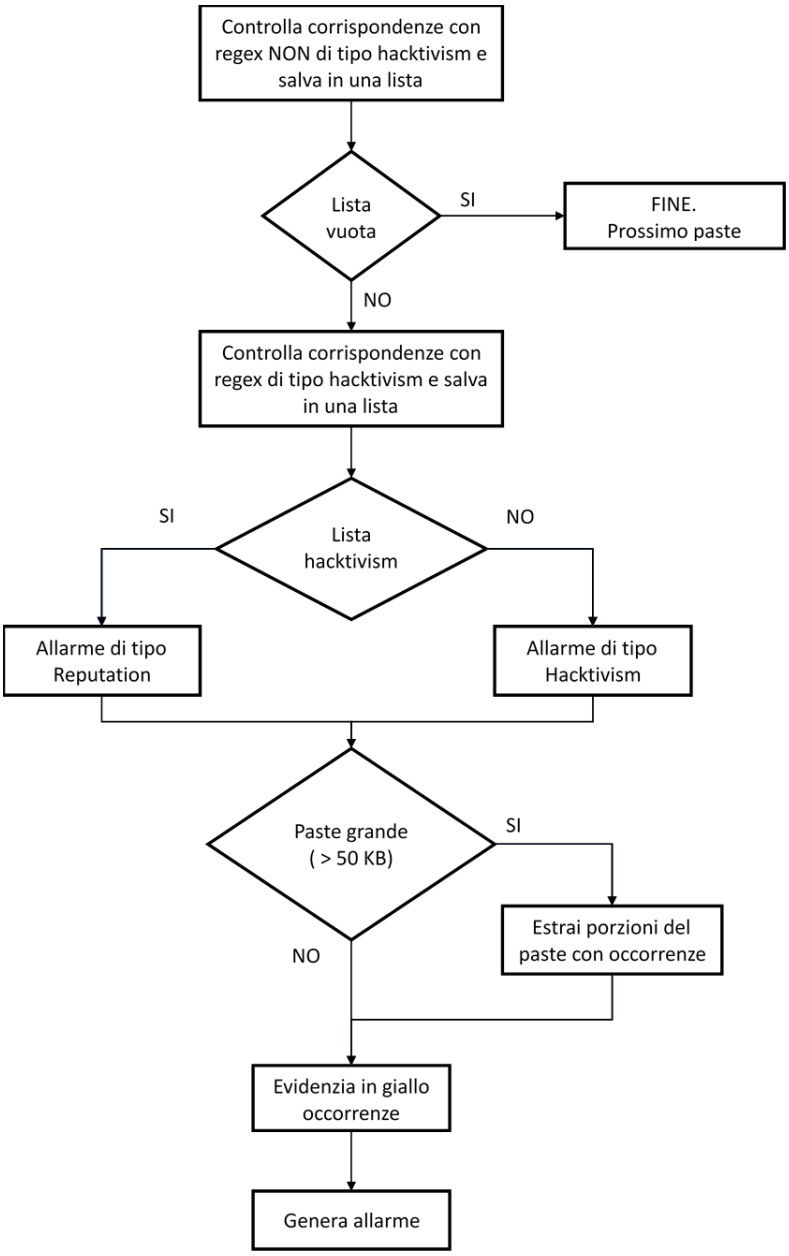

Figura 3.4. Pastebin - Diagramma di flusso

```
      },
      function(error, response) {
      });
    });
});
```

Come si può vedere, per indicizzare un CVE in Elasticsearch ho usato il suo ID CVE, così che se dovessero esserci aggiornamenti sulla vulnerabilità, ogni giorno si aggiornerebbe anche il documento relativo nell'indice.

3.5.3 Confronto CPE

Per ognuno degli asset aziendali (anch'essi indicizzati in Elasticsearch con l'opportuno CPE) viene controllato ogni giorno attraverso il modulo di gestione vulnerabilità che non ne esista una che lo contenga, attraverso la seguente porzione di codice:

```
cpecliente.forEach(function(cpe) {

  elastic.search(
  {
    index: "cve",
    type: "cve",
    size: 100,
    q: 'cve.configurations.nodes.cpe.cpeMatchString:"' + cpe + '"'
  },
  function(error, response) {
    // crea allarme
    // ...
  });
});
```

Nel caso in cui venisse trovato un asset nel database, viene notificato attraverso un allarme della piattaforma, con un livello di priorità che dipende dal CVSS (Common Vulnerability Scoring System) assegnato, valore che varia da 0.0 a 10.0. Nel database, per ogni vulnerabilità, è infatti presente un *Impact Score*, in base al quale viene scelta la priorità dell' allarme:

- Priorità bassa se il punteggio è minore 4.0
- Priorità media se il punteggio è maggiore o uguale a 4.1 e minore di 7.0
- Priorità alta se il punteggio è superiore o uguale a 7.0

Nella piattaforma è quindi possibile vedere l'allarme, con azioni compensative suggerite che prevedono solitamente di aggiornare l'asset o il cambio della sua configurazione. Nell'allarme sono inoltre presenti tutte le informazioni mostrate nel CVE in cui è presente il CPE trovato. L'ID usato per indicizzare l'allarme è generato dal CPE e dal CVE, così che se ci fossero aggiornamenti sul CVE l'allarme si aggiornerebbe. Inoltre in questo modo non viene generato un nuovo allarme ogni giorno.

3.6 Controllo botnet

Il servizio di monitoraggio botnet è quindi diviso in due moduli: uno per l'ottenumento di tutti gli indicatori tramite le API della piattaforma descritta nel capitolo Analisi 2.3.5, l'altro per scaricare i log indicizzati ed effettuarne il confronto con gli indicatori ottenuti. E' inoltre presente il log collector che si occupa di trasformare in JSON ed indicizzare ogni log proveniente dal DNS.

3.6.1 Log collector

Per convertire in formato JSON ogni log DNS e per indicizzarlo in Elasticsearch abbiamo configurato una espressione regolare nel log collector. In automatico crea gli oggetti JSON dalle righe del file scritto da syslog e li inserisce in un opportuno indice: *queryDNS-cliente/queries*. Una regola di rotazione dei log è stata configurata su syslog, in modo da archiviare le query dell'ultimo mese sul server, comprimendo il file di output giornalmente. Si avrà un archivio compresso *tar.gz* per ognuno dei 30 giorni precedenti a quello corrente, così da avere un piccolo storico in caso di analisi.

3.6.2 Download indicatori

Un modulo Node.js effettua ogni giorno il download di tutti i nuovi indicatori. Questi indicatori vengono aggiunti all'indice di Elasticsearch, disponibili sia per il modulo botnet, sia grazie ad un'apposita sezione della piattaforma dove è possibile interrogare questo indice.

La porzione di codice responsabile di ottenere ed indicizzare gli indicatori può essere così sviluppata:

```
var request = require("request");

request({
  url: "https://esempiopiattaforma.com/api"
  }, function(err, resp, body) {

  var result = JSON.parse( body );

    result.forEach(function(elem) {
      elastic.index(
      {
        index: "indicators",
        type: "indicators",
        id: elem.id,
        body: elem
      },
      function(error, response) {
      });
    });
});
```

3.6.3 Confronto query DNS

Il modulo è uno script Node.js dell'engine che è schedulato per scaricare i log dall'indice ogni 2 minuti. Siccome la quantità dei log è elevata, per non saturare la memoria del server e per non degradare le prestazioni di ricerca di Elasticsearch, cancella gli indicatori dopo averne confrontato le query e le risposte DNS con la lista di indicatori. Per estrarre in ordine i log indicizzati viene eseguita la seguente query:

```
elastic.search(
{
  index: "queryDNS-cliente",
  type: "queries",
  size: 5000,
  sort: "timestamp:asc"
},
function(error, response) {
  var logs = response.hits.hits;
  // ...
});
```

Viene salvato inoltre il timestamp del log più recente, che verrà usata in seguito per eliminare tutte gli elementi elaborati.

Per ogni log DNS ottenuto viene eseguita la seguente query per vedere se è presente una corrispondenza con un indicatore:

```
var log = {
```

```
  query: "query",
  answer: "answer"
};

elastic.search(
{
  index: "indicators",
  type: "indicators",
  size: 1000,
  _source: [ "indicator" ],
  body: {
    query: {
      filtered: {
        filter: {
          bool: {
            should: [
              {
                term: { indicator: log.query }
              }
            ]
          }
        }
      }
    }
  }
},
function(error, response) {
  // ...
});
```

Il numero di indicatori da ricercare viene limitato a 1000 attraverso il campo *size* della query. Il campo *_source* indica che Elasticsearch deve ritornare solo l'attributo "indicator" nella risposta. L'attributo *should* nella query permette di indicare un vettore di condizioni che sono tra loro in OR logico. La query viene sempre messa, mentre la risposta viene messa in OR solo se il tipo di query appartiene ai seguenti tipi: A, AAAA, CNAME, MX, NS, PTR.

Per ogni match il modulo genera quindi un allarme con il log intero che conteneva almeno un indicatore e riporta inoltre la data in qui la query è stata effettuata (convertendo il timestamp della query) e quali indicatori sono stati trovati (può essere sia la query sia la risposta).

Dopo aver analizzato i log indicizzati, l'indice viene svuotato grazie alla seguente porzione di codice, usando la funzionalità di *delete by query* di Elasticsearch, elimina tutte le query indicizzate prima dell'ultimo timestamp ottenuto:

```
elastic.deleteByQuery({
  index: "queryDNS-cliente",
  type: "queries",
  body: {
    query: {
      range: {
        timestamp: {
          lte: last_timestamp
        }
      }
    }
  }
}, function (error, response) {
});
```

Capitolo 4

Risultati

I risultati dei moduli fanno riferimento ad allarmi generati nel periodo Marzo 2017 - Luglio 2017, tranne le statistiche relative al modulo *pastebin* che ho sviluppato e rilasciato da Maggio 2017 e le statistiche del modulo *botnet* che sono riferite solamente all'ultima settimana di Luglio 2017.

4.1 Antidefacement

Il modulo è attivo per due clienti ed in totale le pagine monitorate sono 144.

Per ogni URL da monitorare, lo script impiega mediamente 14 secondi per eseguire una fase di running o di learning. Gli allarmi generati sono molti, tutti falsi positivi tranne in un caso in cui c'è stata l'introduzione illecita di un collegamento ad una pagina di *malvertising*, cioè una pagina con contenuti pubblicitari (anche legittimi) che però è stata segnalata come malware site. In quel caso siamo stati in grado di segnalarlo ai gestori della pagina del cliente, che hanno rimosso il collegamento malevolo. Come si può vedere dal grafico in figura 4.2 la maggior parte degli allarmi (74%) sono di priorità bassa, causato principalmente dal valore di differenza fuzzy hash che supera anche di poco la soglia, come visibile nella figura 4.1, dove il 48% degli allarmi sono di tipo "*Fuzzy Hash*". Molto spesso succede infatti che la pagina durante la fase di *learning* si mostri "statica", ottenendo come soglia il valore 0. E' sufficiente quindi un minimo cambiamento nella pagina (cambio di ID di un oggetto HTML, cambio della sezione pubblicitaria, aggiornamento della data) a causare la generazione di un allarme. Il 20% degli allarmi hanno priorità alta perchè il 16% sono stati causati da un errore temporaneo della pagina, rappresentato dal tipo "*Pagina non disponibile*". Se la pagina non è disponibile, la percentuale di similarità sarà 0 e farà scattare quindi un allarme con priorità alta. Tuttavia questo ha permesso anche di monitorare la disponibilità delle pagine dei clienti.

Dal punto di vista delle prestazioni risulta invece molto utile la modifica che permette ad una URL di passare direttamente da learning a running. Questo ha reso molto più efficiente il monitoraggio dello script, automatizzando un processo che poteva diventare molto lungo da svolgere a mano (si pensi all'occasione in cui una URL terminava il learning fuori dall'orario lavorativo, ad esempio sabato). In questo modo è necessario accedere alla piattaforma solo nei rari casi in cui lo script calcolava una baseline errata o quando era presente contenuto illegittimo.

4.2 Controllo permutazioni

Il modulo è attivo per tre clienti, per un totale di 305 domini monitorati.

Come mostra la figura 4.3 il 3% degli allarmi sono stati segnalati come potenziale typosquatting, significando ad esempio che il dominio non è di proprietà del cliente e che ha del contenuto malevolo.

La figura 4.4 mostra quanti allarmi contenevano l'attributo specificato sulla sinistra della figura. Si può vedere che sono stati segnalati due domini con valore fuzzy hash elevato. Questi in realtà

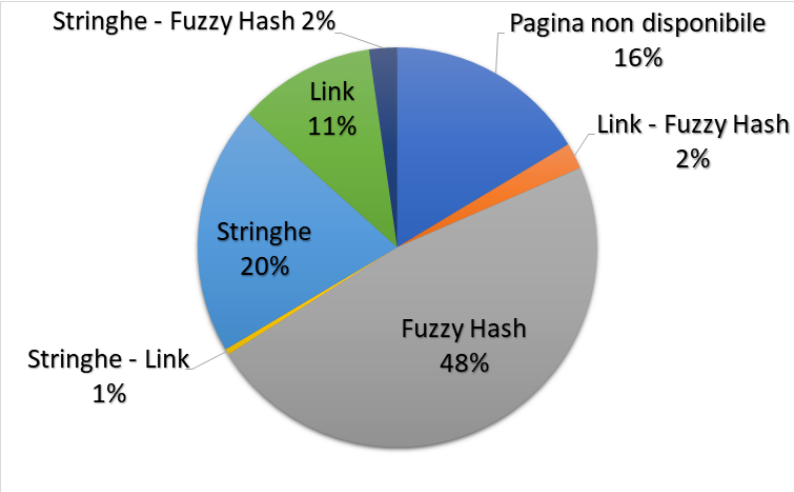

Figura 4.1. Antidefacement - Cause allarmi

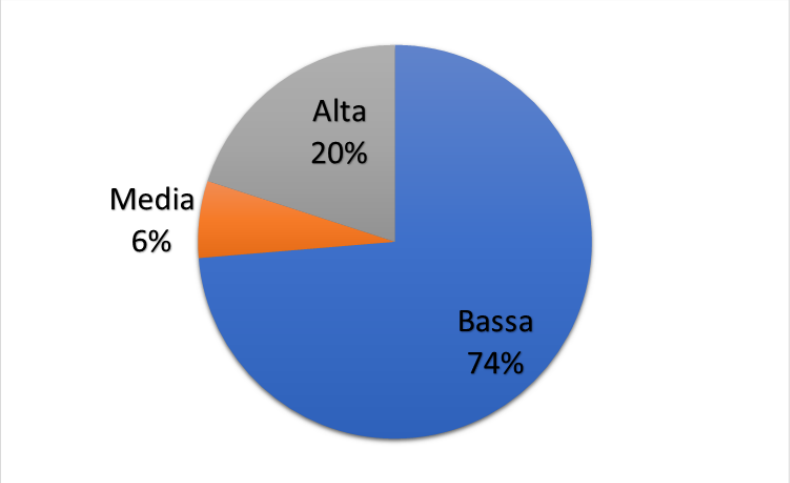

Figura 4.2. Antidefacement - Priorità allarmi

erano di proprietà del cliente e quando vi si accedeva veniva effettuata una *redirect* sul sito originale, per questo il fuzzy hash alto. Ho quindi messo le permutazioni in questione nella lista dei domini legittimi. Si può notare che quasi tutte le permutazioni avevano sia un indirizzo IP sia un name server, mentre solo la metà aveva anche un mail server. Solo in un caso la permutazione aveva tutti gli attributi ed è il caso di uno dei domini rimandava al sito originale.

La figura 4.5 invece mostra quali sono i tipi di permutazioni più frequentemente segnalate. Il tipo *subdomain* è quello più popolare, seguito dal *bitsquatting*. Si nota che i tipi di permutazioni più frequenti non sono quelle che derivano da errori di battitura, ma dalle permutazioni che assomigliano visivamente al dominio originale o che sono generate da un errore del dispositivo. L'obiettivo principale può essere quindi quello di effettuare phishing grazie a questi domini, così che quando si incontra un collegamento con tale permutazione, per distrazione non ci si accorga che non è quello originale.

4.3 Scadenza domini

Il modulo è attivo per gli stessi tre clienti del modulo di controllo permutazioni, monitorando quindi gli stessi domini.

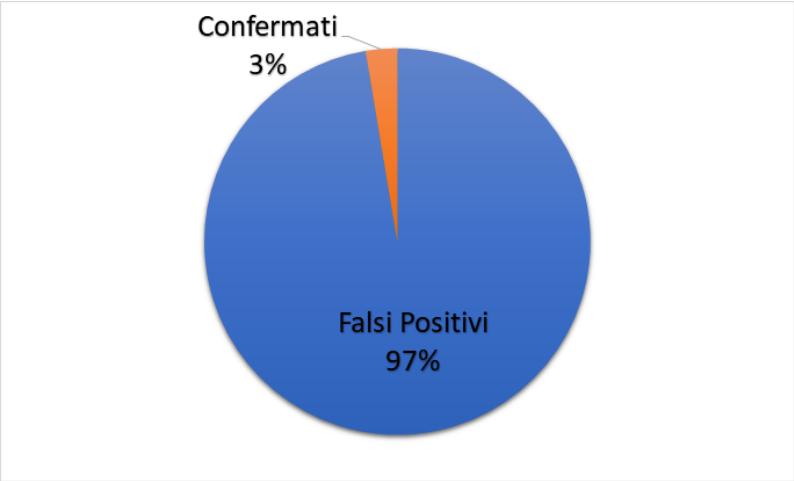

Figura 4.3. Permutazioni - Falsi positivi

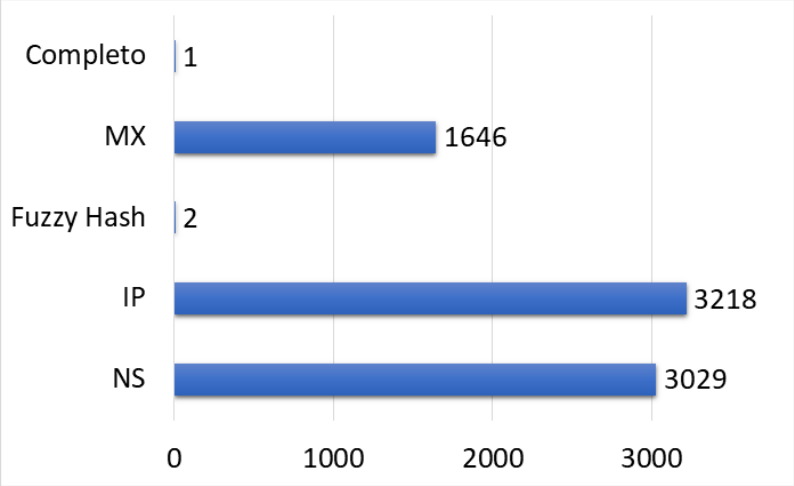

Figura 4.4. Permutazioni - Cause allarmi

La figura 4.6 mostra che il 31% degli allarmi hanno segnalato un dominio che scadeva in meno di un mese dal controllo. Ignorando il caso in cui un dominio da monitorare sia stato aggiunto al monitoraggio a meno di un mese dalla sua scadenza (5 casi su 305), risulta che il 45% delle segnalazioni fatte a due mesi dalla scadenza sono state ignorate fino a quando mancava meno di un mese alla scadenza, in cui è stato generato anche il secondo allarme. Per il 2% delle segnalazioni invece il dominio non è stato rinnovato entro la data di scadenza, continuando a segnalarli anche dopo una e due settimane.

4.4 Pastebin

Il modulo è attivo per un solo cliente.

Questo modulo è quello che ha il miglior rapporto allarmi confermati/allarmi segnalati. Come si può vedere dalla figura 4.7, il 9% delle segnalazioni sono riferite a potenziali minacce o data breach del cliente. Preciso che in falso positivo possono ricadere paste in cui erano presenti riferimenti al cliente ma che non rappresentavano una minaccia.

Come mostra il grafico 4.8, i tipi di segnalazioni sono equilibrate tra hacktivism e reputazione, dove la maggior parte delle segnalazioni di tipo hacktivism contengono termini che però erano

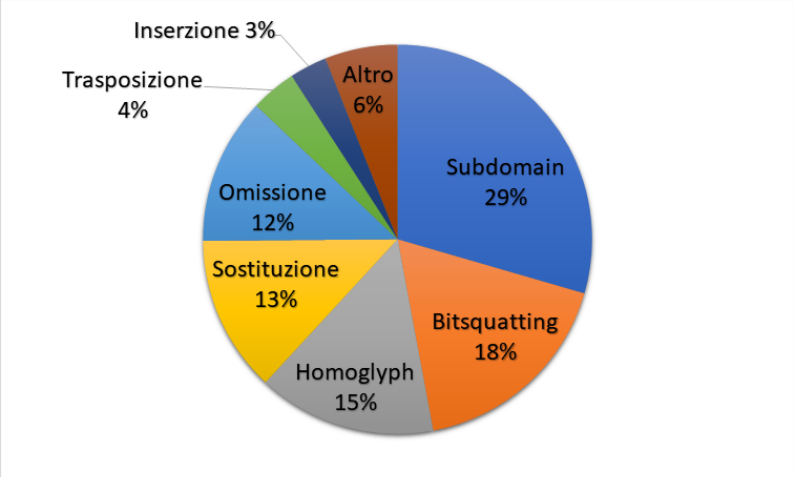

Figura 4.5. Permutazioni - Tipi permutazioni

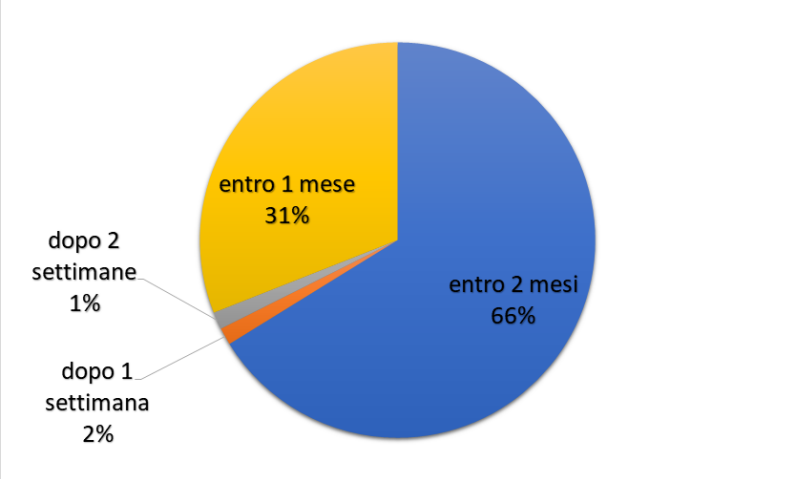

Figura 4.6. Scadenze - Cause allarmi

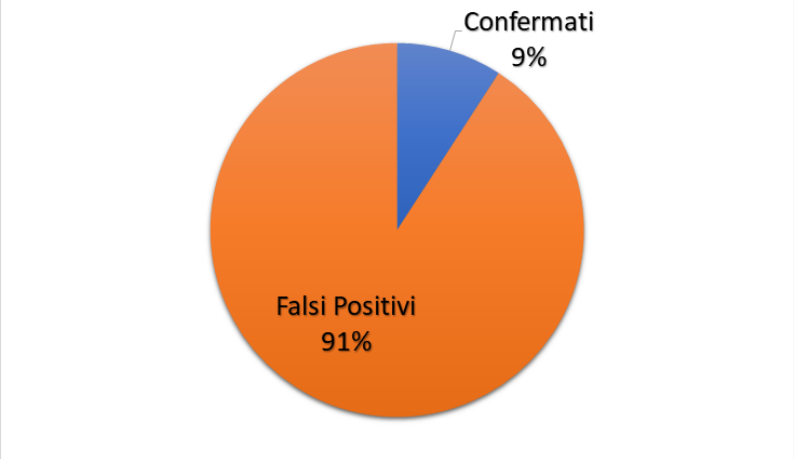

Figura 4.7. Pastebin - Falsi positivi

inerenti all'attivismo informatico (ad esempio la parola "dos" era intesa come il numero due in lingua spagnola o portoghese).

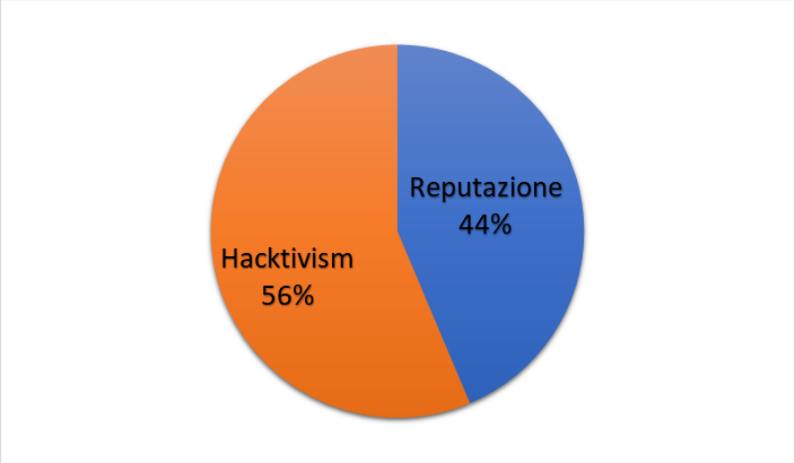

Figura 4.8. Pastebin - Tipi di allarmi

Le prestazioni del modulo seguono un andamento abbastanza lineare rispetto alla dimensione del paste. Il tempo necessario per eseguire il match di un paste con tutte le regex infatti cresce in media linearmente con la sua dimensione, mostrando che il calcolo aggiuntivo per la divisione di un paste in blocchi non è eccessivo come mostrato in figura 4.9. Va considerato che ho calcolato le tempistiche di elaborazione quando tutti i servizi dell'engine erano attivi, quindi ci potrebbero essere latenze aggiuntive dovute alla concomitanza di esecuzione di altri moduli.

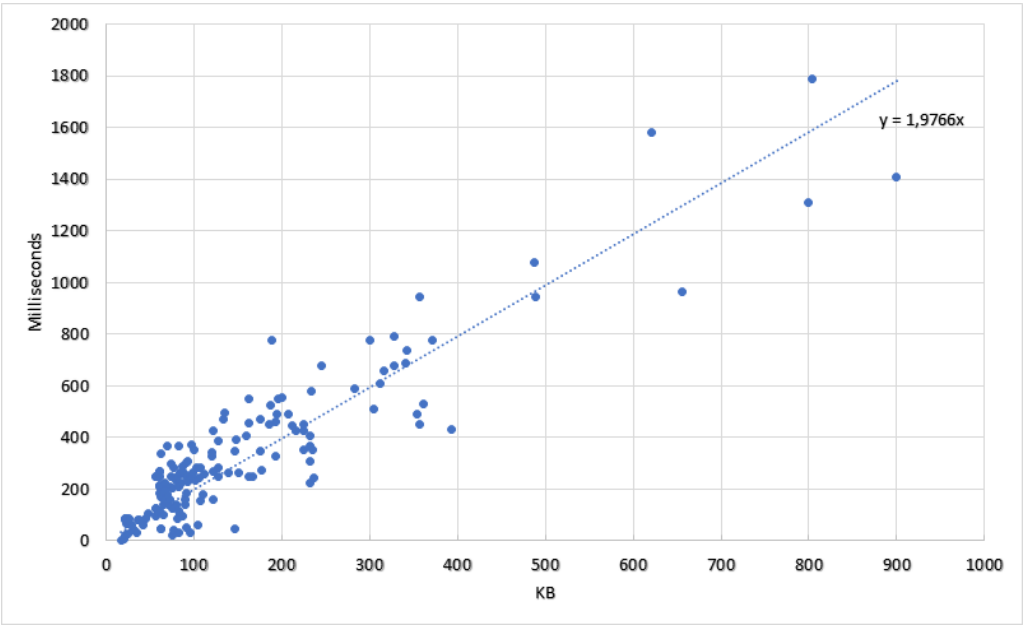

Figura 4.9. Pastebin - Tempo / Dimensione

Dal grafico riferito ai tipi di espressioni regolari che hanno generato allarmi, rappresentato in figura 4.10, si può vedere che la maggioranza di allarmi è stata generata da regex di tipo *parole chiave*. In realtà quel numero è dato dall'espressione regolare riferita all'acronimo del cliente. Per quanto la abbia scritta in modo da evitare il maggior numero di falsi positivi, non potevo limitarla troppo in quanto avrei rischiato di perdere documenti significanti. Gli allarmi delle regex di tipo *indirizzi email* sono riferite a dump di credenziali da siti web minori, dove i dipendenti del cliente

si sono registrati con le credenziali aziendali. Per questi si è quindi suggerito quindi il cambio password ed il consiglio di non registrarsi su siti per scopi privati con le credenziali aziendali. Gli allarmi di tipo *dominio* confermati invece sono sempre stati associati a siti di phishing che avevano come target il cliente. Su questi allarmi è quindi stato consigliato di richiedere lo shutdown della pagina. Per gli allarmi riferiti alle carte di credito invece è stato consigliato di eseguire il processo anti-frode.

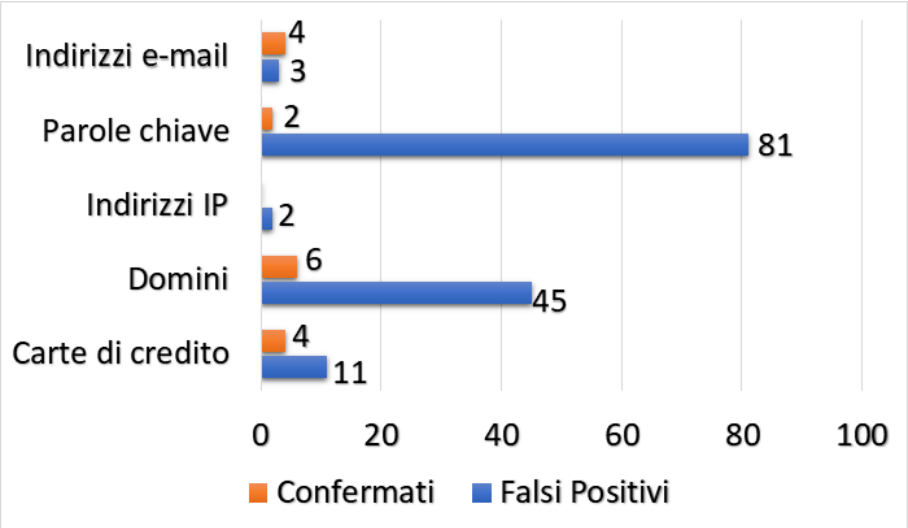

Figura 4.10. Pastebin - Cause allarmi

4.5 Gestione vulnerabilità

Il modulo è attivo per un cliente, che ci ha fornito un totale di 95 CPE.

La priorità degli allarmi generati è mostrata in figura 4.11. Quando il cliente riceve una segnalazione, solitamente corregge la vulnerabilità e ci notifica il caso in cui aggiorni l'asset, così che a nostra volta aggiorniamo il relativo CPE. In questo caso, non sono presenti falsi positivi.

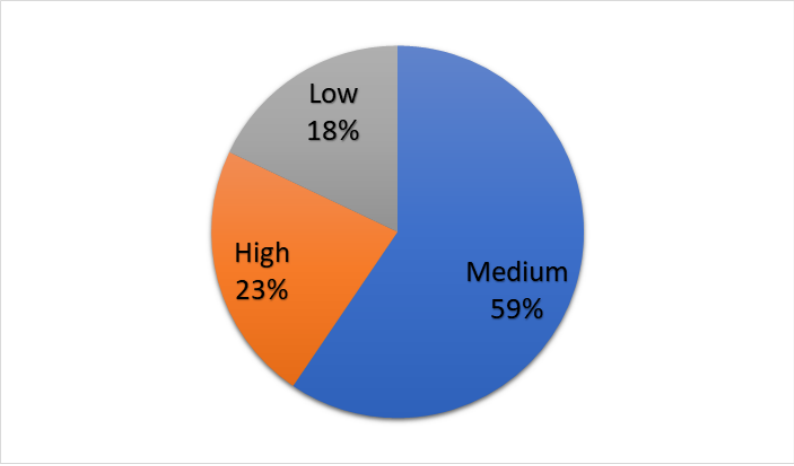

Figura 4.11. Vulnerabilità - Priorità allarmi

4.6 Controllo botnet

Il modulo è attivo per un solo cliente.

Dato un singolo log del DNS, la ricerca tramite query Elasticsearch sugli indicatori indicizzati impiega in media 2 millisecondi ad ottenere una risposta, con un valore di picco di 9 millisecondi. Gli indicatori archiviati sono più di 6 miliardi. Grazie a questo modulo siamo stati in grado, sebbene sia stato attivato da pochi giorni, di individuare molte interazioni anomale con il server DNS, ad esempio domini usati da applicazioni Android installate sui dispositivi per contattare pagine di pubblicità potenzialmente malevole. E' stato individuata l'interazione con indirizzi IP usati per la distribuzione e Command & Control di noti malware. Per ogni segnalazione è necessario al momento effettuare analisi sull'indicatore rilevato, attraverso piattaforme di condivisione di threat intelligence, per verificare quanto recentemente l'indicatore sia stato aggiunto, o se l'ultima associazione malevola risale a molto tempo prima. Tuttavia sono presenti falsi positivi relativi agli indirizzi IP associati ai server NTP (Network Time Protocol). Le macchine del cliente sono infatti configurate per sincronizzare l'orologio di sistema attraverso il protocollo NTP, usando un nome DNS associato ad un "pool" di macchine, che ha la funzione di bilanciare le richieste. Nel 78% dei casi, come mostrato in figura 4.12, è infatti capitato che l'indirizzo IP della macchina scelta dal bilanciatore sia un indicatore di compromissione, associato a server C2 o considerato come *malware site* da vari prodotti antivirus. Questo però non dovrebbe incidere sulla pericolosità dell'interazione NTP sulla porta 123, ma solo nel caso si contatti l'indirizzo su altre porte, quali ad esempio la 80 dell'HTTP. Segnaliamo in ogni caso l'allarme in quanto non abbiamo visibilità di come l'interazione con quell'IP sia stata svolta (se sulla porta dell'NTP o su altre), in modo che dal lato del cliente possano fare ulteriori verifiche.

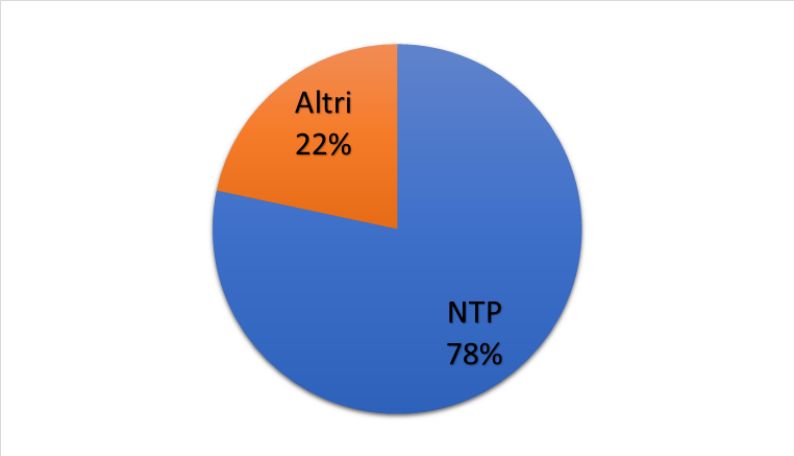

Figura 4.12. Botnet - Falsi positivi

Capitolo 5

Conclusioni

I moduli che ho sviluppato sono stati integrati nella piattaforma con successo, cominciando in seguito a numerosi test e raffinamenti a produrre allarmi per i clienti, riducendo man mano la percentuale di falsi positivi. Gli allarmi generati hanno permesso di mitigare molte minacce e situazioni di potenziale pericolo per i clienti.

Alla piattaforma manca la strutturazione e la possibilità di esportare i dati in formati specifici quali openIOC o STIX. Sarebbe utile anche configurare opportunamente il protocollo TAXII per ottenere intelligence da altre piattaforme o comunità.

Come ulteriore fonte di SOCMINT c'è in programma di utilizzare le API di Twitter, in quanto molti attacchi e informazioni di rilievo sono spesso pubblicate su questo social network. Monitorandolo permette di rilevare annunci e preavvisi di attacchi dai gruppi che sono soliti farlo.

Per il servizio di monitoraggio domini stiamo pensando di sviluppare un modulo che controlla l'acquisto di nuovi domini, grazie ad un servizio che permette di ottenere tutte le nuove registrazioni, in modo da estendere le funzionalità anti-cybersquatting già sviluppate e quelle di phishing, dato che in molti casi viene utilizzato anche un opportuno certificato. Combinato a questo esiste un progetto di Google, chiamato *Certificate Transparency*, che permette di ottenere in tempo (quasi) reale tutti i certificati emessi dalle Certification Authorities (CA) che aderiscono al progetto. In questo modo un'organizzazione può monitorare certificati potenzialmente pericolosi, in quanto registrati con riferimenti al suo dominio, ad esempio in seguito alla compromissione di una certification authority.

Si potrebbe infine integrare la ricezione di intelligence da tutte le piattaforme open-source accennate nella sezione *Esempi di piattaforme open-source* 1.2.6.

Di seguito le conclusioni che ho potuto trarre nel dettaglio da ogni modulo.

5.1 Antidefacement

Lo script è sicuramente da ottimizzare e da migliorare. Gli script bash sono adatti a piccoli o semplici task, mentre la complessità del modulo di antidefacement è cresciuta in seguito all'introduzione di molte funzionalità. Il debugging, la gestione degli errori e la loro correzione risulta difficile e porta via molto tempo rispetto a quello che ci si impiegherebbe con un linguaggio più moderno. Per questo sto riscrivendo il modulo in Node.js. Per aumentare l'efficacia del modulo e per produrre degli allarmi più rilevanti, sarà necessario introdurre il confronto tra gli screenshot oltre che del codice sorgente. E' vero che l'aspetto di una pagina è determinato dal codice sorgente, ma sarebbe sufficiente cambiare soltanto il nome di un'immagine usata per la pagina e l'aspetto potrebbe cambiare drasticamente, avendo come effetto solamente un minimo cambiamento del fuzzy hash del codice. In seguito all'aggiunta della funzionalità di confronto tra immagini, si potrebbero correlare i due tipi di differenze, in modo da produrre nel caso un allarme più consistente.

L'espressione regolare usata per estrarre i link potrebbe essere migliorata, rilevando quelli formati via javascript con codifica esadecimale (https%3A%2F%2Fesempio.it). Si potrebbe aggiungere anche il controllo di nuovi link nei file ".js" per evitare invio di dati ad altri domini attraverso interazioni *cross domain*. In generale, il modulo produce molti falsi positivi e ho pensato ad alcune migliorie che potrebbero evitarlo. Anzichè concentrarsi sull'entità del cambiamento (fuzzy hash), bisognerebbe concentrarsi sul tipo di cambiamento. La sola differenza del codice HTML dovrebbe essere una condizione marginale, che solo combinata a qualcos'altro avrebbe un peso. Si potrebbe rilevare quando una pagina risponde con un errore temporaneo o con un HTML vuoto, così da non generare un allarme con priorità alta. Un altro punto importante riguardo alle migliorie del modulo è lo sviluppo di un'interfaccia di gestione. Ogni volta che una URL dev'essere rimossa o aggiunta, alla situazione attuale bisogna aggiungere la URL nel file di configurazione ed eseguire uno script che carica le informazioni necessarie nell'indice Elasticsearch. E' ancora più macchinoso quando devono essere aggiunti o modificati i link permessi o le occorrenze delle stringhe tipiche, che necessitano di query manuali su Elasticsearch. L'interfaccia quindi può essere sviluppata nella sezione "Antidefacement" della piattaforma, oltre alla funzionalità di mandare una URL in learning.

5.2 Controllo permutazioni

Il modulo di controllo permutazioni genera molti allarmi, soprattutto falsi positivi. Questo perchè la logica decisa per la generazione lo prevede. Molti domini rilevati infatti cambiano spesso l'indirizzo IP associato, aggiungono o rimuovono un name server o un mail server, producendo molti allarmi che potrebbero essere limitati ad esempio "riconoscendo" una precedente segnalazione. Un modo per limitare la generazione di falsi positivi si potrebbe avere ottenendo la lista dei "parked domains", in modo da evitare di analizzare le permutazioni che rientrano in questo insieme. Si potrebbe migliorare l'aspetto antiphishing del modulo. Lo script utilizzato calcola solo il fuzzy hash della permutazione e lo confronta con il dominio originale. Questo però può non essere sufficiente, dato che del codice HTML completamente diverso potrebbe produrre lo stesso aspetto della pagina originale. Per questo si potrebbe sostituire il comportamento standard dello script per la parte di riconoscimento similarità, ad esempio implementandone uno ad-hoc in quanto a volte lo script non ritorna un valore di fuzzy hash. Si ricercherebbero all'interno della pagina stringhe riferite al cliente o addirittura si potrebbe implementare un algoritmo di riconoscimento immagini per individuare loghi del cliente. Sarebbe utile anche riconoscere se la pagina associata ad una permutazione è legittimamente posseduta, con un certificato valido. Questo potrebbe essere usato per notificare un allarme con una priorità più bassa. Per facilitare l'analisi degli allarmi si potrebbe aggiungere, se la permutazione ha un indirizzo IP associato, lo screenshot della pagina, così che non si debba aprirla e potenzialmente finire vittima del typosquatting. Per evitare di calcolare ogni volta le permutazioni e per una più efficiente interazione si potrebbero archiviare le permutazioni per ogni dominio senza doverle calcolare ogni volta. In questo modo lo script python si userebbe solo al momento dell'aggiunta di un nuovo dominio per calcolarne le permutazioni. Molte volte succede inoltre che lo script non riesce ad ottenere il codice HTML della pagina in quanto alcune non la forniscono se non sono precedute da "www". Nel modulo da migliorare bisognerebbe quindi aggiungere questa opzione. Allo script per il calcolo delle permutazioni andrebbe aggiunto il tipo "swapping", cioè l'inversione di due lettere all'interno del dominio. Anche per questo modulo sarebbe molto utile l'interfaccia di gestione, una sezione della piattaforma dove è possibile aggiungere/rimuovere i domini monitorati.

5.3 Pastebin

Il modulo ha generato molti allarmi verificati come minaccia, di documenti contenenti username/mail e password di utenti, provenienti da "dump" di database di siti anche minori su cui i dipendenti si erano registrati con credenziali aziendali. Ha individuato codici di carte di credito con tutti i dati per poter essere utilizzate (processo di carding). Al tempo stesso per determinate espressioni regolari vengono generati molti falsi positivi. Ci sarebbero alcune migliorie che si potrebbero fare al modulo, sia per evitare falsi positivi, sia per migliorare la qualità dei match. Si

potrebbe aggiungere il concetto di *blacklist* per i paste già analizzati come falsi positivi, tramite fuzzy hash del paste o manualmente con regex che, se contenute nel documento, prevengono la generazione dell'allarme. Capita infatti che lo stesso paste venga pubblicato più volte, magari con una piccola modifica, che provocherebbe la generazione di allarmi multipli. Si potrebbe migliorare la divisione a blocchi dei paste più grandi, per evitare che la divisione coincida proprio con un match. Infine, si potrebbe migliorare l'assegnazione della priorità dell'allarme, in quanto ora sono tutti notificati con priorità bassa. Si potrebbe quindi far scattare un allarme con una priorità maggiore se ad esempio in un documento che contiene match con regex di tipo email è presente anche la keyword "password", oppure per regex di tipo carte di credito è presente la keyword "CVV". Anche per questo modulo l'interfaccia di gestione sarebbe molto comoda per aggiungere/modificare/rimuovere espressioni regolari, in quanto al momento sono generate ed inserite manualmente in Elasticsearch.

5.4 Gestione vulnerabilità

Siccome la notifica di aggiornamento di un asset in seguito alla segnalazione di una sua vulnerabilità è ancora un processo manuale, sarebbe utile anche per questo modulo un'interfaccia di gestione sulla piattaforma, che permetta di mostrare gli asset configurati e di modificarli, anche da parte del cliente. Potrebbe essere utile inoltre integrare altri database di vulnerabilità.

5.5 Controllo botnet

Il modulo potrebbe essere migliorato in quanto il match per ora è solo esatto. Ad esempio, se la query fosse *sub.polito.it* e l'indicatore fosse *polito.it* (o viceversa), il match non si avrebbe. Per questo stiamo pensando ad un algoritmo per confrontare anche sotto-domini e generare allarmi con priorità più alta se il match è esatto. Potremmo inoltre integrare altre sorgenti di indicatori, così da espandere la lista di IP e domini potenzialmente pericolosi che sono contattati dalla rete del cliente. Il confronto è "a posteriori": perchè ci sia un match, l'indicatore dev'essere già stato analizzato dalla comunità e dev'essere stato reso disponibile nella piattaforma usata. Questo può impiegare qualche giorno, se non mesi. Per questo si potrebbe pensare di archiviare le query DNS per un po' di tempo per rielaborarle in seguito, dando tempo alla piattaforma di raccogliere nuovi indicatori. La priorità dell'allarme potrebbe essere infine differenziata a seconda che l'indicatore trovato sia la risposta (indirizzo IP) o la query (il dominio). Se l'indicatore fosse il dominio la priorità sarebbe più alta, soprattutto se il dominio ricercato è generato con algoritmi DGA. Per un allarme più preciso, bisognerebbe incrociare l'indicatore segnalato con l'effettiva interazione con esso. Come descritto nella sezione risultati (4.6), l'indirizzo IP può essere usato con un protocollo che non è pericoloso (NTP).

Bibliografia

[1] National Council of ISACs, https://www.nationalisacs.org/
[2] Tor Project, https://www.torproject.org/
[3] Rete anonima I2P, https://geti2p.net/it/
[4] CPNI, "Threat Intelligence Infographic", https://www.cpni.gov.uk/advice/cyber/Threat-Intelligence
[5] STIX Project, https://oasis-open.github.io/cti-documentation/stix/intro
[6] STIX, http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part3-core.html
[7] TAXII Project, https://taxiiproject.github.io/
[8] TAXII, "Introduction to TAXII", https://oasis-open.github.io/cti-documentation/taxii/intro.html
[9] CSIRT Gadgets Foundation, "Collective Intelligence Framework", http://csirtgadgets.org/collective-intelligence-framework/
[10] MITRE, "Collaborative Research Into Threats' - CRITs', https://crits.github.io/
[11] The MANTIS Cyber-Intelligence Management Framework, http://django-mantis.readthedocs.org/en/latest/
[12] Malware Information Sharing Platform MISP - A Threat Sharing Platform, http://www.misp-project.org/
[13] SOLTRA, http://www.soltra.com/
[14] Recorded Future, "Threat Intelligence and SIEM", https://www.recordedfuture.com/siem-threat-intelligence-part-1/
[15] P. Mell and T. Grance, "The nist definition of cloud computing", National Institute of Standards and Technology, vol. 53, no. 6, Article 50, 2009 DOI 10.6028/NIST.SP.800-145
[16] Nginx, https://www.nginx.com/resources/wiki/
[17] AngularJS, https://angularjs.org/
[18] Node.js, https://nodejs.org/
[19] Elasticsearch https://www.elastic.co/products/elasticsearch
[20] DB-Engines, "Ranking of Search Engines" https://db-engines.com/en/ranking/search+engine
[21] CURL, Command line tool and library https://curl.haxx.se/
[22] Apache, Hadoop http://hadoop.apache.org/
[23] "Web Hacking Terms: What is Website Deface/Defacement", https://www.pinoyhacknews.com/web-hacking-terms-what-is-website-defacedefacement
[24] H. Baier, F. Breitinger, "Introduction to Context Triggered Piecewise Hashing", https://www.fbi.h-da.de/fileadmin/personal/h.baier/Lectures-summer-11/SS-11-Seminar-T/introduction-ctph-110331.pdf
[25] ssdeep Project https://ssdeep-project.github.io/ssdeep/index.html
[26] JQ, "A lightweight and flexible command-line JSON processor", https://stedolan.github.io/jq/
[27] A. Dinaburg, "Bitsquatting: DNS Hijacking without Exploitation", https://media.blackhat.com/bh-us-11/Dinaburg/BH_US_11_Dinaburg_Bitsquatting_WP.pdf
[28] Pastebin, https://pastebin.com/
[29] ISO/IEC, "Information technology – Security techniques-Information security risk management 27005:2008"
[30] R. Shirey, "Internet Security Glossary", RFC-2828, May 2000, DOI 10.17487/RFC2828

[31] Gary Stoneburner (NIST), Alice Goguen (BAH), Alexis Feringa (BAH), "SP 800-30 Risk Management Guide for Information Technology Systems", DOI 10.6028/NIST.SP.800-30
[32] CNSS, CNSS Instruction No. 4009, https://www.ecs.csus.edu/csc/iac/cnssi_4009.pdf
[33] MITRE, "The standard for information security vulnerability names", http://cve.mitre.org/
[34] MITRE, "A structured naming sheme for IT systems, platforms and packages", https://cpe.mitre.org/
[35] ThreatConnect, "What is a Threat Intelligence Platform", https://www.threatconnect.com/threat-intelligence-platform-2-2/
[36] P. Kampanakis, "Security automation and threat information-sharing options", J. IEEE Security & Privacy. 12, 42-51 (2014)
[37] Nginx, "What is a reverse proxy server", https://www.nginx.com/resources/glossary/reverse-proxy-server/
[38] Yunchuan Sun, Junsheng Zhang, Yongping Xiong, Guangyu Zhu, "Data Security and Privacy in Cloud Computing", DOI 10.1155/2014/190903
[39] J. Connolly, "The trusted automated exchange of indicator information (taxii)", http://taxii.mitre.org/about/documents/Introduction_to_TAXII_White_Paper_July_2013.pdf
[40] FIRST, "Common Vulnerability Scoring System SIG", https://www.first.org/cvss/
[41] Dave Shackleford, SANS, "Who's Using Cyberthreat Intelligence and How?", https://www.sans.org/reading-room/whitepapers/analyst/who-039-s-cyberthreat-intelligence-how-35767
[42] G. Farnham, K. Leune, SANS, "Tools and Standards for Cyber Threat Intelligence Projects" https://www.sans.org/reading-room/whitepapers/warfare/tools-standards-cyber-threat-intelligence-projects-34375
[43] J. Steinberger, A. Sperotto, M. Golling, H. Baier, "How to exchange security events? Overview and evaluation of formats and protocols", IEEE International Symposium on Integrated Network Management (IM), pp. 261-269. IEEE, Los Almitos (2015)