Alma Mater Studiorum · Università di Bologna

CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

MECCANISMI DI RISPOSTA RAPIDA AGLI INCIDENTI DI SICUREZZA MEDIANTE RICONFIGURAZIONE SEMI-AUTOMATIZZATA DEI SISTEMI DI PROTEZIONE

Tesi di Laurea in Sicurezza delle Reti

Relatore:
GABRIELE D'ANGELO

Presentata da:
EUGENIO SEVERI

Correlatore:
MASSIMILIANO BATTIGAGLIA

Sessione III
Anno Accademico 2017-2018

Introduzione

La sicurezza informatica ricopre un ruolo sempre più centrale a livello mondiale: di anno in anno gli attacchi ai danni di soggetti pubblici e privati sono in continua crescita, sia numericamente sia nella loro varietà e livello di sofisticatezza. Di conseguenza, anche l'impatto in termini economici è notevole. I trend degli ultimi anni hanno evidenziato un'impennata dei *ransomware* (software malevoli che cifrano i file delle vittime prendendoli "in ostaggio" e chiedendo un riscatto per riottenere l'accesso), dei furti di dati da grandi aziende pubbliche e private (come nel caso di *Equifax* nel 2017), dei *miner* (software che sfruttano la potenza di calcolo dei sistemi delle vittime per generare criptovalute) e degli attacchi legati ai dispositivi *Internet of Things* (IOT). [1]
In un mondo sempre più connesso e dipendente da Internet, è lecito ipotizzare che questo trend proseguirà ancora per anni. Ad essere colpiti non sono solo i soggetti di alto profilo, ma anche le piccole aziende e i privati cittadini. Le risorse che gli attaccanti impiegano per raggiungere il loro obiettivo sono direttamente proporzionali al guadagno che otterrebbero in caso di successo; procede di pari passo la sofisticatezza delle metodologie d'attacco applicate. Tuttavia, sebbene i soggetti di minori dimensioni risultino meno appetibili, è altrettanto vero che questi sono tendenzialmente meno disposti ad investire in sicurezza: gli attacchi nei loro confronti, per quanto meno sofisticati, vanno comunque a segno.
Per contrastare gli attacchi informatici è di primaria importanza la prevenzione, attraverso la corretta configurazione dei propri sistemi, l'installazione

di soluzioni per la sicurezza come firewall e antivirus e l'educazione degli utenti, sensibilizzandoli alle problematiche di sicurezza. Questo non è però sufficiente, in quanto anche la migliore delle configurazioni rimane pur sempre esposta ad eventuali vulnerabilità non ancora note e, poiché le configurazioni necessitano dell'intervento umano, non è da escludere che si commettano errori.
Per questi motivi, per tutelarsi adeguatamente, si richiede anche un continuo monitoraggio dei sistemi alla ricerca di eventuali attività malevole che non sono state bloccate. Infine, qualora un attacco in corso venisse rilevato, è fondamentale agire in maniera tempestiva per terminarlo ed impedirne la propagazione, in quanto i danni potenziali prodotti da un attaccante aumentano direttamente con il tempo di persistenza all'interno del sistema.
Date queste premesse, si è deciso di svolgere una tesi in azienda presso *VEM sistemi S.p.A.*, che da anni si occupa della difesa delle reti dei loro clienti, in collaborazione con *Certego S.r.l.*, la società del gruppo che si occupa di monitoraggio e *Incident Response*. In particolare, ci si è focalizzati sulla progettazione e lo sviluppo di un sistema robusto ed estensibile in grado di ridurre notevolmente i tempi tra il rilevamento di un attacco in corso da parte degli analisti dell'*Incident Response Team* e la sua terminazione, attraverso l'automatizzazione dell'inserimento di regole di blocco all'interno dei firewall dei clienti.
La tesi è strutturata in tre capitoli, secondo la seguente suddivisione:

- un'introduzione alle tecnologie utilizzate nell'ambito della tesi, comprendente le architetture firewall, le VPN (*Virtual Private Network*), gli IDS (*Intrusion Detection System*) e lo stile architetturale REST (*Representational State Transfer*);

- una trattazione sull'argomento della "sicurezza come processo", seguita dalle analisi dei requisiti e del problema che si intende trattare, comprensive di *test plan*;

- la descrizione del processo di progettazione, implementazione e testing del sistema.

Un'appendice contiene le istruzioni operative per il *deployment* su alcuni degli apparati di sicurezza presi in considerazione nell'ambito di questo progetto di tesi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Panoramica delle tecnologie utilizzate

In questo capitolo verranno presentate le tecnologie coinvolte nell'ambito del progetto realizzato, con particolare enfasi sugli aspetti inerenti alla sicurezza informatica. In particolare, saranno illustrate le architetture firewall, i metodi di comunicazione sicura attraverso una rete intrinsecamente insicura come Internet utilizzando la crittografia e REST (uno stile architetturale pensato per la realizzazione di servizi web, attraverso la definizione di vincoli architetturali, che sarà approfondito in seguito nella sezione 1.4) applicato alla progettazione di servizi web.

1.1 Le architetture firewall

Un firewall (figura 1.1), nella sua accezione più generica, è un dispositivo atto a proteggere una rete informatica dagli attacchi veicolati tramite la rete stessa, provenienti sia dall'esterno sia dall'interno. Questa funzione viene assolta ponendo in essere limitazioni sul traffico di rete ammesso, che altrimenti sarebbe inoltrato indiscriminatamente. Per questo motivo, di norma, un firewall è implementato su un router, con lo scopo di proteggere tutta la rete. È altresì possibile implementarlo sui singoli host della rete: in questo

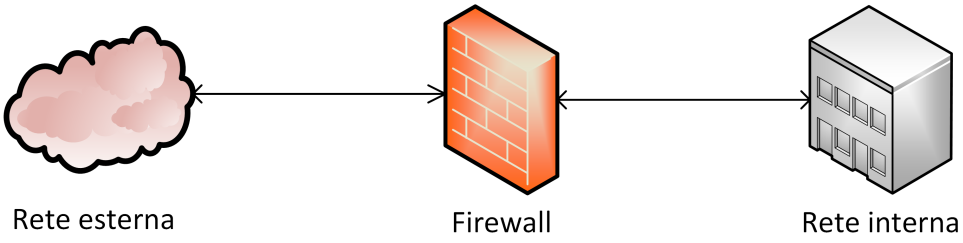

Figura 1.1: Schema di un firewall generico

caso la protezione sarà specifica per il dispositivo in questione [2].
Internet è oramai onnipresente, sia a livello personale che aziendale. Trattandosi di una rete ad accesso pubblico, non si può dare per scontato che i dati in transito su di essa non siano malevoli. Per questo motivo si rende necessario non rendere la propria rete trasparente a tutti i tipi di traffico, ovvero restringerli a quelli strettamente necessari.
Più le regole di filtraggio saranno restrittive, maggiore, almeno in teoria, sarà la sicurezza della rete. Impostare regole complesse può, tuttavia, risultare oneroso e richiedere competenze specialistiche. Sarà quindi importante analizzare attentamente il *threat model*, ovvero identificare le vulnerabilità potenziali sulla rete presa in esame e valutarne l'impatto, in modo da ordinarle in base alla loro priorità e stabilire se vale o meno la pena di mitigarle nel caso specifico.
Storicamente i firewall sono nati come meccanismo di difesa esclusivamente perimetrale, con l'idea che gli host interni fossero fidati ed eventuali minacce provenissero esclusivamente dall'esterno. Questa assunzione si è dimostrata eccessivamente limitativa, poiché le minacce possono provenire anche dall'interno tramite comportamenti scorretti da parte degli utenti, attacchi inizialmente provenienti dall'esterno non filtrati correttamente dal firewall o vulnerabilità nel firewall stesso.
Le buone norme di progettazione di un firewall prevedono che tutto il traffico in ingresso e in uscita dalla rete passi attraverso il firewall e, per impostazione predefinita, tutto il traffico debba essere bloccato ad eccezione di quanto specificato tramite apposite regole (approccio a *whitelist*). Inoltre, il firewall

dovrebbe essere un sistema fidato, virtualmente inattaccabile: questo obiettivo non può essere raggiunto formalmente ma si possono utilizzare tecniche di *hardening* del sistema operativo e dei software che lo costituiscono. [3]

1.1.1 Tipologie di firewall

Le policy di un firewall possono essere applicate al traffico a differenti livelli: dagli strati più bassi della rete (livello 3 del modello ISO/OSI) a quelli applicativi. ISO/OSI è uno standard che suddivide la struttura logica delle reti in una pila a sette livelli, ognuno dei quale si occupa di una funzione specifica, passando dal mezzo fisico ai dati finali (figura 1.2). Nella pratica, i tre livelli più alti vengono condensati in uno solo. In base a ciò che si

Figura 1.2: Modello ISO/OSI

vuole ottenere dal singolo firewall, può essere opportuno monitorare ad un livello piuttosto che un altro. Ad esempio, risalendo la pila ISO/OSI, i dati ricavabili si fanno sempre più ricchi di contenuto informativo e presentano informazioni di livello sempre più alto, a fronte però di maggiori oneri computazionali (che, nei casi dei sistemi ad alte prestazioni, possono facilmente costituire un collo di bottiglia limitando il throughput della rete) e una crescente probabilità di incorrere in falsi positivi.

Verranno ora analizzate le tipologie di firewall, procedendo in ordine cronologico e di complessità crescente.

Packet filter

Un packet filter firewall utilizza le informazioni presenti al livello di rete ed eventualmente al livello di trasporto. Ogni datagramma è trattato separatamente, senza correlarlo agli altri: non è quindi possibile tracciare le connessioni. Per questo motivo è anche detto *stateless*. [4] Nel caso del protocollo IP, i dati utilizzabili sono:

- l'interfaccia di rete sul firewall da cui proviene il pacchetto;
- l'interfaccia di rete sul firewall a cui il pacchetto è destinato;
- l'indirizzo IP sorgente;
- l'indirizzo IP di destinazione;
- il tipo di protocollo di livello superiore utilizzato;
- la porta sorgente, come indicato nel livello di trasporto;
- la porta di destinazione, come indicato nel livello di trasporto.

Le regole di filtraggio impostate vengono processate in cascata, fino a quando una non corrisponde al singolo pacchetto in esame. Nel caso nessuna regola dia luogo ad una corrispondenza, verrà intrapresa un'azione predefinita, ad esempio di blocco del pacchetto.
Il vantaggio di questa tipologia di firewall è la semplicità, che si traduce in una minore complessità sia di progettazione che computazionale durante l'esercizio. Tuttavia, tale semplicità comporta alcune vulnerabilità intrinseche [5]:

- non analizzando il livello applicativo, questi firewall non proteggono da vulnerabilità specifiche delle applicazioni;

- la capacità di produrre log è limitata dai pochi dati disponibili ai livelli a cui il firewall opera;
- di norma non sono supportati meccanismi avanzati di autenticazione degli utenti;
- eventuali vulnerabilità nel livello di trasporto, come ad esempio lo *spoofing* degli indirizzi IP da parte di un attaccante (l'invio di pacchetti con un indirizzo IP sorgente falsificato, in modo che al destinatario sembri provenire da una fonte attendibile), consentendo la trasmissione di eventuali pacchetti che non fanno parte di una connessione già attiva;
- essendo le regole impostate basate su pochi parametri, risulta più facile commettere errori durante la configurazione, dando origine a regole troppo permissive o limitative.

Nella tabella 1.1 è presente un semplice esempio di policy per un packet filter firewall che separa una rete locale da Internet. Viene consentito il solo traffico SMTP (Simple Mail Transfer Protocol) su porta 25, bidirezionalmente. Le regole sono applicate in cascata, dall'alto verso il basso.

Protocollo	**Src IP**	**Dest IP**	**Dest Port**	**Azione**
TCP	Esterno	Interno	25	Accetta
TCP	Interno	Esterno	$\geq$1024	Accetta
TCP	Interno	Esterno	25	Accetta
TCP	Esterno	Interno	$\geq$1024	Accetta
*	*	*	*	Rifiuta

Tabella 1.1: Esempio di policy per un packet filter firewall

Stateful

Un firewall stateful, a differenza di un packet filter firewall, è in grado di considerare non solo i pacchetti individualmente, ma è anche in grado di

analizzare il contesto in cui si trovano. [4] Questo è possibile per i protocolli *connection oriented* come TCP, in cui esiste il concetto di connessione ed è possibile identificare i pacchetti che ne fanno parte, attraverso l'intestazione del pacchetto. [6]
Nel caso dei protocolli client-server la connessione viene avviata dal client utilizzando un numero di porta compreso tra 1024 e 65535, generato dal sistema operativo e non predicibile. In assenza di informazioni sulla connessione, un packet filter firewall dovrebbe consentire il traffico in ingresso su tutte queste porte, nonostante sia effettivamente necessario solo sulle singole porte coinvolte in una connessione in quel momento.
Un firewall stateful è in grado di tenere traccia delle connessioni aperte e consentire dinamicamente il traffico in ingresso solo sulle porte strettamente necessarie, riducendo notevolmente la superficie di attacco.
Alcuni firewall di questo tipo tengono conto anche dei numeri di sequenza TCP, con lo scopo di ostacolare attacchi di tipo *session hijacking*.
Rispetto ai packet filter firewall richiedono una maggiore capacità di elaborazione [7] che, sebbene in passato costituisse un potenziale ostacolo, ad oggi non rappresenta più un problema: qualunque firewall moderno, anche a livello domestico, offre funzionalità stateful.
Pur risolvendo alcune delle vulnerabilità intrinseche dei packet filter firewall, rimane la mancata capacità di analisi del livello applicativo [4]. Inoltre, essendo presente una tabella delle connessioni, si introducono scenari di attacchi di tipo DoS (Denial of Service) che puntano alla saturazione di tale tabella. [8]
In tabella 1.2 è presente un esempio di traccia delle connessioni TCP attive.

Application-level

Un application-level firewall (o anche "application-level gateway" o "application proxy") agisce come intermediario a livello applicativo, filtrando tutto il traffico relativo ad uno stesso servizio/applicazione. [7, 9]

Src IP	Src Port	Dest IP	Dest Port	Stato
10.0.1.127	2304	10.0.2.15	25	Established
10.0.1.132	1897	10.0.2.15	80	Established
10.0.4.40	45060	10.0.3.27	443	Established
10.0.5.16	8660	10.0.4.200	80	Established

Tabella 1.2: Esempio di tabella delle connessioni

Poiché questa funzionalità è dipendente dal singolo servizio, solo i servizi esplicitamente supportati possono sfruttarla. Un firewall di questo tipo è anche potenzialmente in grado di limitare le funzionalità di un protocollo ad un suo sottoinsieme.
A differenza dei firewall precedentemente descritti, un application-level gateway è in grado di analizzare tutti i livelli della pila ISO/OSI, rendendolo potenzialmente più sicuro, a fronte di maggiori costi computazionali e complessità di gestione delle regole che, se non impostate correttamente, possono facilmente causare falsi positivi nel riconoscimento delle minacce. Anche le capacità di logging sono migliorate dalle informazioni di alto livello estrapolate dai pacchetti.
Uno dei punti cardine del funzionamento di un application-level gateway è quello di operare come un *man-in-the-middle* (ovvero intercettare ed eventualmente modificare il traffico in transito), al fine di scansionare il traffico. Il client perde quindi la possibilità di contattare direttamente il server nel caso di connessioni crittografate, ad esempio tramite SSL/TLS. Infatti, a stabilire la connessione sicura verso il server destinatario sarà il firewall, in modo da poter analizzare i dati in transito. Questo problema può essere aggirato o rinunciando alle funzionalità di proxy per i protocolli crittografati o installando nell'archivio dei certificati dei client una copia di quello utilizzato dal firewall (tipicamente autogenerato) in modo che il browser dell'utente non segnali anomalie, accettando però che i dati sul firewall vengano processati in chiaro. Quest'ultima opzione è attuabile solo se si ripone nel firewall e

nella sua configurazione la massima fiducia.

Circuit-level

Un circuit-level firewall (o anche "circuit-level gateway" o "circuit-level proxy") è simile ad un application-level firewall in quanto non permette ad un client di comunicare direttamente con un server, ma si comporta da intermediario. La differenza principale è che non consente ad un client di contattare direttamente il server, bensì il firewall segmenta la connessione in due sotto-connessioni: una dal client al firewall e una dal firewall al sistema destinatario. [10]
La funzione principale di un firewall di questo tipo è quella di isolare gli host della rete interna da quelli esterni, stabilendo quali connessioni consentire e quali no.
In base alle implementazioni è possibile integrare o meno anche le capacità di analisi del traffico. Per questo motivo nella pratica, spesso, non si ha una netta distinzione tra un circuit-level e un application-level firewall, coesistendo entrambe le funzioni su un unico prodotto. Naturalmente, rinunciando alla funzione di ispezione del traffico, il carico sul sistema si riduce. Una configurazione di questo tipo può essere giustificata nei casi in cui gli utenti di un sistema siano sufficientemente fidati da non richiedere un continuo monitoraggio.
Un'implementazione molto diffusa di circuit-level gateway è SOCKS. [11]

Next-generation

Con il termine next-generation firewall si intende un prodotto che include svariate tecnologie finalizzate ad implementare complessivamente la sicurezza di una rete. Proponendosi come prodotti unificati per la sicurezza, l'obiettivo è quello di rendere la configurazione più agevole rispetto alla gestione di molti dispositivi separati, oltre ad ottenere una maggiore efficienza computazionale. [4]
Sono quindi presenti le funzionalità firewall già citate, dal semplice filtrag-

gio dei packet filter firewall al monitoraggio del livello applicativo. [8] Sono inoltre presenti funzionalità come il supporto alle VPN (Virtual Private Network), ai NAT (Network Address Translation), a policy QoS (Quality of Service) e *traffic shaping*, ad *URL filtering*, ad abilità di IDS/IPS (Intrusion Detection/Prevention System), a sistemi di autenticazione integrati con i domini aziendali e all'integrazione con fonti di intelligence di terze parti per prevenire gli attacchi.

1.1.2 Implementazioni dei firewall

Come accennato precedentemente, un firewall di norma è implementato su un sistema hardware dedicato, sia per motivi di sicurezza (un sistema che offre molti servizi ha una superficie di attacco maggiore), sia per motivi prestazionali (nel caso dell'elaborazione di grandi quantitativi di traffico, l'overhead dovuto a tali elaborazioni non è trascurabile).
Da un punto di vista strettamente tecnico, un firewall è un componente software, per cui lo si può implementare potenzialmente su qualunque tipo di computer, eventualmente anche in ambienti virtuali. Di norma un firewall è basato su un sistema operativo come Linux (tramite *Netfilter*) o UNIX (tramite *pf*), opportunamente configurato in modo da includere i pacchetti necessari a fornire le funzionalità richieste e su cui è stato eseguito un processo di *hardening* (irrobustimento del sistema attraverso la rimozione di servizi non necessari, corretta configurazione di quelli restanti e impostazione di policy di accesso restrittive) volto a migliorarne la sicurezza. Sono anche possibili implementazioni proprietarie.
Alcuni vendor distribuiscono l'hardware e il software dei loro firewall in maniera congiunta e indivisibile (sotto forma di *appliance*), mentre altri consentono di installare il solo software su hardware di propria scelta. Talvolta il sistema può essere open source e gratuito per uso personale, come pfSense [12].
Verranno ora analizzate alcune tipiche implementazioni di firewall a livello concettuale.

Bastion host

Il termine "bastion host" deriva da un articolo del ricercatore Marcus J. Ranum, in cui lo descrive come un sistema critico per la sicurezza della rete e, come tale, deve essere configurato e gestito eseguendo hardening del sistema operativo e dei servizi. [13] Questo include l'utilizzo di un sistema operativo quanto più minimale possibile, rimuovendo o limitando tutti i servizi non strettamente necessari, in modo da ridurre la superficie di attacco e rendere più complicato un eventuale attacco.
Di norma, un bastion host (un esempio è visibile in figura 1.3) ospita un application-level o circuit-level firewall. Anche le funzionalità di logging e auditing sono cruciali, in quanto sono lo strumento che consente all'amministratore di identificare ed impedire gli attacchi, oltre a ripristinare un corretto funzionamento nel caso un attaccante avesse successo.

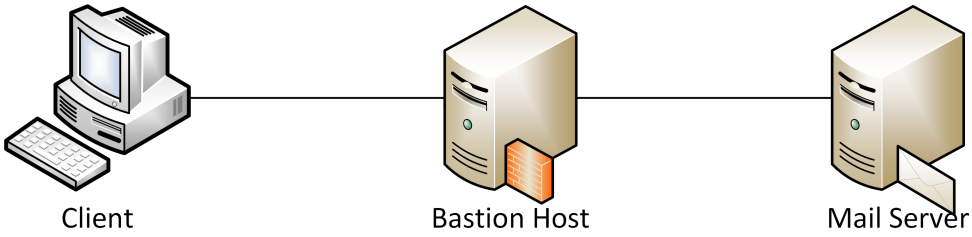

Figura 1.3: Bastion host

Host-based firewall

Un host-based firewall si occupa di proteggere un singolo host della rete, a differenza delle tipologie fino ad ora trattate, che proteggono genericamente tutta la rete.
I vantaggi di questa implementazione sono la possibilità di progettare regole di filtraggio basate anche su informazioni specifiche del singolo host (come ad esempio quale sia il software che sta avviando una data connessione) e di impostare regole diverse per ogni host.
È opportuno utilizzare questa tipologia di firewall non come alternativa ad

un firewall a livello di rete, ma in congiunzione ad esso, in modo da semplificare la configurazione di quest'ultimo (non richiedendo di inserire regole specifiche per ogni host) e aggiungere un ulteriore livello di filtraggio, migliorando la sicurezza complessiva. Inoltre, in caso di compromissione dell'host, è possibile che anche le regole di firewall impostate siano compromesse a loro volta.

1.1.3 Segmentazione

Nel caso di reti strutturate, è buona norma segmentare la rete in più sottoreti separate in base alla funzione degli host che le compongono, eventualmente avvalendosi di strumenti come le VLAN (Virtual Local Area Network), al fine di definire un insieme di regole specifico per ogni sottorete. Ad esempio, si possono separare le risorse che necessitano di essere utilizzate solo dall'interno della rete (file server interni e le workstation degli utenti) da quelle che devono essere accessibili anche dall'esterno (ad esempio il server web che ospita i servizi forniti al pubblico). In questo caso si parla di DMZ (demilitarized zone).
Poiché gli host di una DMZ sono esposti verso l'esterno, hanno una probabilità teoricamente maggiore di essere attaccati con successo. In assenza di segmentazione, l'eventuale compromissione di un host di questo tipo esporrebbe all'attaccante anche gli host interni, sebbene non siano direttamente raggiungibili dall'esterno. In figura 1.4 è presente lo schema di un'architettura firewall di questo tipo. Attraverso l'uso di VLAN è possibile condensare questa architettura utilizzando meno dispositivi fisici.

1.2 Virtual Private Network

Una virtual private network (VPN) è una soluzione pensata per connettere tra loro due reti distinte attraverso una rete insicura. Si pensi all'esempio di due sedi di un'azienda che necessitano di scambiarsi reciprocamente dati ma, poiché tale scambio avviene attraverso la rete Internet, non sono garan-

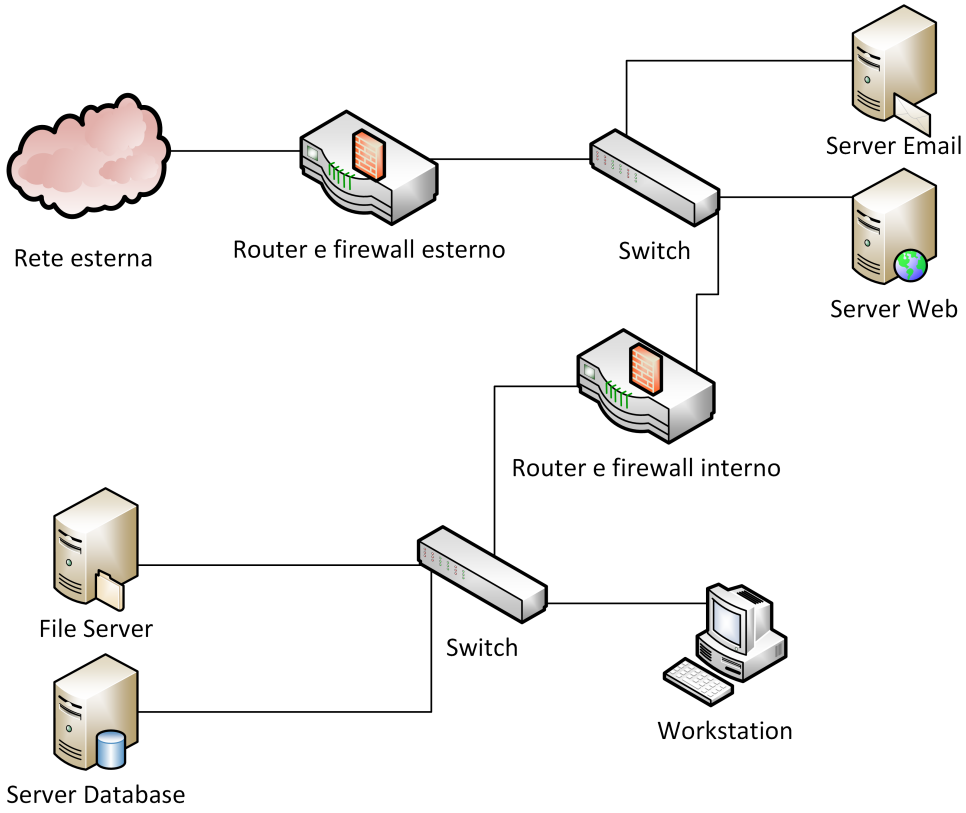

Figura 1.4: Schema di un'architettura firewall con DMZ

titi i requisiti di autenticazione, confidenzialità e integrità. La soluzione a questo problema è sfruttare la crittografia (simmetrica e asimmetrica) come strumento abilitante ad una comunicazione sicura, impedendo l'intercettazione e la modifica da parte degli attaccanti.

Per via del loro ambito di utilizzo, le VPN sono spesso implementate sui dispositivi che fungono da firewall, in modo da essere comprese nelle policy di filtraggio. Il loro utilizzo diventa quindi trasparente agli utenti, che possono accedere allo stesso modo tanto alle risorse locali quanto a quelle remote. Qualora le funzionalità VPN siano richieste da un singolo host, è anche possibile utilizzare l'host stesso come endpoint della connessione. Questa possibilità è spesso utilizzata nel caso del lavoro da postazioni remote o comunque nei casi in cui ad almeno una delle estremità del tunnel VPN si

trovi un sistema singolo invece che un'intera rete.
È importante considerare che è impossibile analizzare il traffico di una VPN se non alle estremità del collegamento, sfuggendo ad eventuali controlli firewall dei nodi intermedi.

1.2.1 Protocolli

Esistono svariati protocolli ideati per implementare una VPN. Tra questi spiccano:

- PPTP (Point to Point Tunneling Protocol), originariamente sviluppato da Microsoft, molto semplice da utilizzare, in quanto non prevede meccanismi avanzati di autenticazione ed è supportato nativamente da tutti i sistemi operativi maggiormente diffusi, ma considerato ad oggi insicuro [14];

- L2TP (Layer Two Tunneling Protocol), funzionante a livello di collegamento della pila ISO/OSI, non è dipendente da algoritmi crittografici predefiniti ma può essere utilizzato in maniera modulare, sebbene la prassi sia di utilizzarlo congiuntamente ad IPsec (IP Security) [15];

- OpenVPN, software open source basato su OpenSSL, utilizza tunnel SSL/TLS, supporta una grande varietà di algoritmi crittografici, metodi di autenticazione (chiavi precondivise, credenziali basate su nome utente e password, certificati), compressione del traffico, possibilità di operare sia a livello di rete che di collegamento, utilizzabile sia tramite TCP che UDP [16].

1.2.2 Site-to-site e remote-access

Le VPN sono generalmente classificate in due tipologie: *remote access* e *site-to-site*.
Le VPN remote access consentono ad un client di collegarsi ad una rete remota, accedendo alle risorse interne a quella rete come se si trovassero in locale.

Il caso d'uso tipico è quello del dipendente di un'azienda che ha necessità di lavorare remotamente: in questo caso la VPN consente al dipendente di accedere alle risorse dell'azienda come se si trovasse in sede. In figura 1.5 è presente lo schema di una VPN di questo tipo.

Le VPN site-to-site sono pensate per collegare tra loro due reti distinte

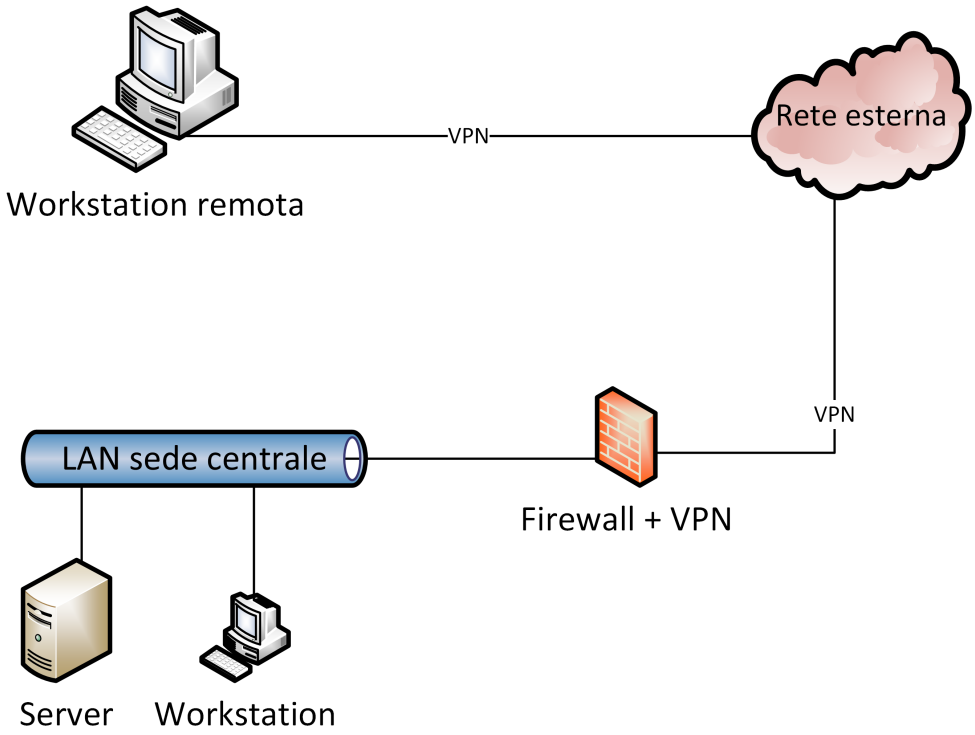

Figura 1.5: VPN Remote Access

(si pensi a due sedi di una stessa azienda). Questo tipo di VPN non viene gestito a livello del singolo client ma a livello di firewall da parte dell'amministratore della rete: dal punto di vista dell'utente, l'accesso alle risorse remote è trasparente. Oltre che per motivi di sicurezza, una VPN site-to-site può essere utilizzata anche per connettere tra loro due reti della stessa tipologia attraverso una rete di tipologia diversa (come ad esempio del caso di IPv6-over-IPv4). [17]

Un esempio di VPN site-to-site è presente in figura 1.6.

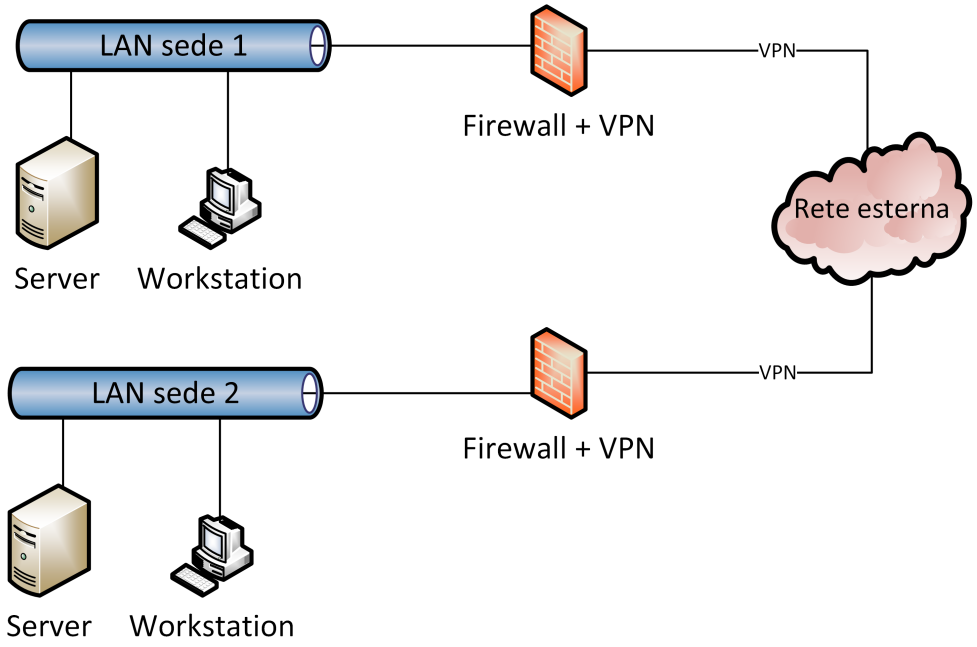

Figura 1.6: VPN site-to-site

1.3 Intrusion Detection and Prevention System

Un Intrusion Detection System (IDS) è un componente software di sicurezza che si occupa di rilevare eventuali intrusioni in un sistema. Nel momento in cui un'intrusione dovesse essere rilevata, l'amministratore ne viene messo a conoscenza, in modo che possa prendere adeguate contromisure. [18]
Un Intrusion Prevention System (IPS) è un IDS con, in più, la capacità di bloccare autonomamente le intrusioni. Per questo motivo, da qui in avanti si parlerà genericamente di IDS (salvo quando diversamente specificato), poiché quanto detto vale anche per gli IPS.
Come accennato precedentemente, un next-generation firewall integra funzionalità di IDS/IPS, sebbene sia possibile utilizzare un sistema dedicato a questa funzione.

1.3.1 Funzionamento

Un IDS è composto da una parte sensore, che si occupa di monitorare ogni parte di un sistema potenzialmente utile alla rilevazione di un'intrusione:

- il traffico di rete fino ai livelli più alti della pila ISO/OSI;
- il comportamento del software in esecuzione sul sistema;
- i file di log del sistema operativo.

Tutti i dati raccolti in questa prima fase vengono successivamente analizzati, verificando eventuali corrispondenze in un database di firme di comportamenti sospetti o noti per essere associati ad attacchi noti e tramite metodi euristici e statistici. In questa seconda fase i dati inizialmente raccolti vengono scremati, sia per ridurre lo spazio occupato, sia per presentare informazioni più utili ad un analista umano, che a sua volta analizzerà le informazioni filtrate per stabilire se effettivamente si sia verificata un'intrusione o se si tratti solo di un falso positivo.
In ogni caso, un IPS deve trovarsi in linea rispetto al flusso dei dati (tipicamente come componente di un next-generation firewall), mentre un IDS potrebbe trovarsi fisicamente separato: è sufficiente che il traffico gli sia inoltrato senza necessità di intervenire sulle regole di blocco. A questo fine è sufficiente uno switch managed con funzionalità di port-mirroring, che inoltri all'IDS una copia dei dati in transito sul router.

1.3.2 Analisi dei dati

Un IDS può utilizzare le seguenti strategie per rilevare le intrusioni: basate sulle anomalie, su firme ed euristiche. [19]
Le prime prevedono una fase iniziale di raccolta dati associati al normale comportamento degli utenti, prolungata nel tempo, al fine di produrre un modello di utilizzo legittimo. Successivamente, i dati relativi alle nuove sessioni vengono confrontati con quelli esistenti alla ricerca di divergenze sospette. La classificazione avviene mediante una varietà di approcci, che

possono essere sintetizzati in statistici, basati sulla conoscenza e basati su machine learning. [20]
Le strategie basate su firme, invece, prevedono la scansione dei dati alla ricerca di corrispondenze con dei pattern di dati noti per essere malevoli. Quelle euristiche, non si basano su sequenze di dati predefinite, bensì su regole comportamentali.

Metodi statistici

I metodi statistici monitorano il traffico in transito e ne creano un modello basandosi su metriche e serie temporali. In ogni momento, il traffico corrente viene confrontato con le serie temporali passate alla ricerca di divergenze significative, tali da indicare un potenziale attacco. Il vantaggio di questo approccio è il basso costo computazionale e la semplicità, non essendo necessario stabilire a priori un modello di traffico legittimo o malevolo; lo svantaggio è costituito dalla difficoltà nel gestire ogni profilo di traffico in maniera puramente statistica: non sempre è possibile individuare metriche adeguate.

Metodi basati sulla conoscenza

I metodi basati sulla conoscenza prevedono la definizione a priori delle regole che modellano i comportamenti legittimi e si verificano eventuali corrispondenze con i dati in esercizio. Il vantaggio di questo approccio è la maggiore flessibilità e robustezza nella definizione delle regole rispetto ai metodi statistici; lo svantaggio è nell'onerosità della fase iniziale di definizione delle regole, che deve essere necessariamente eseguita manualmente (almeno in parte) ed è suscettibile ad errori (destinati a diminuire nel tempo, attraverso il raffinamento continuo delle regole).

Metodi basati su machine learning

I metodi basati su machine learning eseguono continuamente data mining sui dati in transito per allenare un'intelligenza artificiale che ha lo scopo di

classificare il traffico futuro in legittimo e anomalo. Il vantaggio principale è l'estrema adattabilità di questo approccio al riconoscimento di minacce future; gli svantaggi sono l'elevato onere computazionale nella fase di data mining, la dipendenza dalla definizione a priori di quale sia il corretto comportamento di un sistema e l'elevato quantitativo di falsi positivi: sebbene sia semplice allenare l'intelligenza artificiale utilizzando traffico legittimo, è molto più difficile reperire grandi quantitativi di traffico associato ad attività malevole.

Metodi basati su firme

Le strategie basate su firme prevedono la scansione delle fonti di informazione alla ricerca di corrispondenze in un database di firme associate ad attacchi noti. Tali database possono essere sia proprietari, forniti dai vendor di prodotti di sicurezza, sia open source. Il vantaggio è costituito dal basso costo computazionale. Gli svantaggi sono la possibilità di riconoscere solamente attacchi noti (lasciando quindi scoperte tutte le nuove minacce) e il continuo sforzo necessario per l'aggiornamento del database. Tale onere è giustificato dalla condivisione di ogni database tra molti utilizzatori (ad esempio il solo database di un vendor può essere fornito a tutti i clienti di quel vendor).

Metodi euristici

I metodi euristici utilizzano anch'essi un database, tuttavia basato su regole comportamentali che tentano di catturare i comportamenti normalmente tenuti dagli attaccanti durante lo sfruttamento di vulnerabilità note, ad esempio analizzando il funzionamento di alcuni strumenti di attacco noti. Un esempio molto conosciuto basato su questo approccio è l'IDS Snort. [21]

1.3.3 Architetture

Esistono tre fondamentali architetture di IDS che utilizzano le strategie di analisi sopra citate: host-based, network-based e ibride. [19]

IDS host-based

Gli IDS host-based sono installati su un singolo host e si occupano della protezione del singolo host, specialmente se contengono dati sensibili o sono considerati di importanza critica nel dominio. Costituiscono una protezione aggiuntiva (e non sostitutiva) ad una corretta configurazione del sistema e dei servizi. Essendo installati sul sistema da proteggere, sono in grado di accedere anche a dati interni non accessibili a livello di rete, come il contenuto dei file di log, l'integrità dei file considerati critici (attraverso algoritmi di checksum) e accessi sospetti a risorse (attraverso l'uso di chiamate di sistema potenzialmente dannose).

IDS network-based

Gli IDS network-based analizzano esclusivamente il traffico di rete, mirando a rilevare minacce provenienti dalla rete, quali malware, attacchi DoS (Denial of Service), scansioni delle porte aperte e attacchi basati sui protocolli di rete, di trasporto e applicativi. Per questo motivo si trovano generalmente in prossimità dei router o, nel caso degli IPS, sul router stesso o lungo il percorso dei dati (a fronte di una latenza di rete leggermente aumentata dall'elaborazione del traffico in tempo reale prima di stabilire se inoltrarlo o bloccarlo).
Non sono in grado di scansionare il traffico crittografato, per cui gli attacchi veicolati tramite canali di questo tipo non vengono rilevati, a meno che, come descritto relativamente agli application-level firewall, non operino come *man-in-the-middle*. Nel caso di una rete strutturata, può essere opportuno installare più di un IDS nei punti critici della rete.

IDS ibridi

Negli ambienti in cui si utilizzano più IDS, tanto di tipo host-based quanto network-based, è possibile centralizzarne la gestione, in modo da correlarne i dati e riconoscere eventuali pattern di attacco, migliorando l'efficacia nel riconoscimento delle minacce. Un esempio può essere costituito da un attacco che si diffonde gradualmente in una rete: senza avere una visione di insieme, l'attacco potrebbe non essere notato in tempo utile da mitigarne i danni e sarebbe più difficile conoscerne con esattezza la diffusione. Inoltre, se gli IDS comunicano tra loro, possono scambiarsi informazioni sull'esistenza di una minaccia: l'IDS che la rileva per primo può intervenire localmente e notificarla agli altri IDS. [22]
Un'architettura IDS ibrida può essere implementata o utilizzando dispositivi di uno stesso vendor che comunicano tra loro utilizzando metodi tipicamente proprietari, o utilizzando soluzioni SIEM (Security Information and Event Management) pensate per integrare le informazioni provenienti da più dispositivi di molteplici vendor.

1.3.4 Falsi positivi e negativi

Il problema dei falsi positivi, ovvero dell'errata classificazione di un comportamento come minaccia, è molto frequente, dato l'alto livello a cui un IDS opera. Questo è il motivo principale per cui in molteplici scenari si preferisce rinunciare alle capacità di filtraggio automatico di un IPS, preferendo un IDS: evitare il rischio di bloccare le attività legittime degli utenti. È anche possibile utilizzare un approccio ibrido, impostando un IPS in modo da bloccare solamente le attività a più alto rischio (che hanno bassa probabilità di incorrere in falsi positivi) e demandare ad un analista la valutazione dei casi più dubbi. Quest'ultima scelta è generalmente un buon compromesso, in base dallo scenario di utilizzo. L'obiettivo generale è quello di minimizzare sia i falsi positivi che i falsi negativi ovvero del mancato riconoscimento di una reale minaccia come tale. [23]

All'aumentare della sensibilità dell'IDS, si avrà un maggior numero di falsi positivi, uno minore di falsi negativi, e viceversa. Idealmente si vorrebbe portare entrambi questi valori a zero ma non è possibile: con impostazioni molto restrittive, i falsi positivi aumenteranno fino al punto di non essere gestibili dagli analisti che quindi tenderanno ad ignorarli; con impostazioni blande, si avranno pochissimi falsi positivi ma è molto probabile che alcune minacce effettive non saranno riconosciute.

1.4 Lo stile architetturale REST

Representational State Transfer (REST) è uno stile architetturale per il software pensato per la progettazione di servizi web, basato sul paradigma client-server. Il termine viene per la prima volta utilizzato nella tesi di dottorato di Roy Thomas Fielding, uno dei padri del protocollo HTTP e del web server Apache. [24]
Essendo REST uno stile architetturale e non un protocollo, non esiste uno standard universale per l'implementazione di un servizio cosiddetto *RESTful*; esistono tuttavia delle buone pratiche comunemente riconosciute, che si sono dimostrate efficaci nel raggiungimento degli obiettivi prefissati.

1.4.1 Vincoli

REST è costituito da una serie di vincoli, i quali limitano l'insieme delle interazioni teoricamente possibili: [25]

- l'architettura utilizzata è di tipo client-server, in cui il server si occupa di rendere disponibili i dati, i quali saranno fruiti dai client indipendentemente dalle loro specificità;

- la comunicazione avviene in maniera stateless, ovvero ogni richiesta deve essere indipendente e contenere tutti i dati necessari a servirla, evitando al server la necessità di memorizzare informazioni di sessione, migliorando la scalabilità;

- di base deve essere possibile memorizzare i dati contenuti nelle risposte in cache (eventualmente a più livelli), in modo da poterle riutilizzare senza doverle nuovamente reperire dai dati sorgente, aumentando la scalabilità e le prestazioni, a meno che questa operazione non abbia senso per i dati in questione;

- un sistema REST può essere organizzato in più livelli, dove i client non si collegano direttamente al server, ma ad un intermediario, al fine di migliorare la scalabilità tramite load balancing e incrementare le prestazioni implementando un livello di cache, riducendo il carico complessivo sul server centrale;

- deve essere fornita un'interfaccia comune a tutte le richieste ovvero tutte le richieste devono contenere l'identificativo della risorsa a cui necessitano di accedere e tutti i dati necessari a gestirle, le risorse vengono manipolate tramite loro rappresentazioni (che costituiscono un'astrazione delle risorse vere e proprie) e i client devono essere in grado di navigare dinamicamente i percorsi (URL) ipertestuali forniti dal server;

- opzionalmente, anche il codice eseguibile può essere parte dei dati trasferiti, con lo scopo di modificare le funzionalità di un client.

Quest'ultimo vincolo pone notevoli problemi in termini di sicurezza, in quanto l'esecuzione di codice arbitrario potrebbe essere sfruttata da un attaccante per veicolare codice malevolo. Nei sistemi in cui questa funzionalità non è espressamente richiesta, è opportuno non implementarla, in modo da eliminare questo vettore d'attacco. Nei casi in cui dovesse essere necessario, invece, si raccomanda di prendere le dovute precauzioni per mitigare il rischio: dotare il codice di firma digitale e consentire sui client l'esecuzione del solo codice firmato ed eseguirlo all'interno di una *sandbox* (un ambiente di esecuzione virtuale con accesso limitato al sistema sottostante).

1.4.2 Funzionalità e obiettivi

L'insieme dei vincoli sopra citati mira a raggiungere una serie di proprietà architetturali di particolare interesse negli ambienti distribuiti e in particolare in un contesto web-based: [24]:

- scalabilità, in quanto devono essere in grado di gestire quantità potenzialmente grandi di interazioni e di componenti contemporaneamente;
- semplicità, attraverso l'utilizzo di un'interfaccia comune per tutte le richieste;
- tolleranza ai guasti potenzialmente a qualsiasi livello dell'architettura;
- alte prestazioni, sia per massimizzare l'efficienza, sia per migliorare la reattività dal punto di vista dell'utente;
- possibilità di modificare i componenti in base alle necessità, anche durante l'esecuzione;
- visibilità della comunicazione tra componenti, consentendo ad altri componenti di monitorarla o fungere da intermediario;
- portabilità dei dati e, opzionalmente, anche del codice.

1.4.3 I servizi web

L'implementazione più comune di un servizio REST è basata sul protocollo HTTP, originariamente pensato e ad oggi ancora utilizzato principalmente per il trasferimento di pagine Web. Come formato per lo scambio di dati, comunemente si utilizza JSON o XML.
REST è regolarmente utilizzato come scelta progettuale per l'implementazione di Web-API, beneficiando così dei principi architetturali sopra citati.
Un'API REST basata su HTTP è basata sui seguenti elementi: [26]

- un Uniform Resource Identifier (URI), ovvero l'indirizzo a cui si trova la risorsa a cui si vuole accedere [27];

- uno dei metodi previsti dal protocollo HTTP (GET, PUT, POST, PATCH, DELETE) [28];
- un media type che specifichi il tipo dei dati in transito, in modo che il client possa utilizzare per la rappresentazione delle informazioni un formato supportato dal server (essendo i dati in sé separati dalla loro rappresentazione) e consentendo la componibilità delle richieste [29];

Ogni risorsa, identificata dal suo URI, supporta uno o più metodi HTTP, in base alle funzionalità che si vuole implementare per tale risorsa. In particolare:

- il metodo GET è un'operazione in sola lettura (quindi priva di *side-effects*) che ottiene una rappresentazione della risorsa specificata;
- il metodo PUT crea una nuova risorsa con i dati specificati e, se già esistente, la sovrascrive;
- il metodo POST crea una nuova risorsa in maniera non distruttiva, ovvero aggiunge i dati contenuti nella richiesta a quelli esistenti, a meno che non esistessero già, nel qual caso non ha effetto;
- il metodo PATCH aggiorna i dati relativi alla risorsa specificata in maniera simile a PUT, con la differenza che non è in grado di creare una nuova risorsa, ma solo di modificarne una esistente;
- il metodo DELETE elimina una risorsa, se esistente (in caso contrario non ha effetto).

1.4.4 Sicurezza

Nei casi in cui si dovesse utilizzare un servizio REST attraverso una rete inerentemente insicura come Internet, è possibile garantire la sicurezza della comunicazione utilizzando gli stessi meccanismi previsti da HTTP per raggiungere lo stesso obiettivo: il protocollo Secure Sockets Layer (SSL), attualmente in via di deprecazione, o il suo successore Transport Layer Security

(TLS). [30] In questo modo è possibile garantire autenticazione, confidenzialità e integrità della comunicazione.
Nella sua implementazione più comune, solo il server si autentica tramite il client attraverso l'utilizzo di un certificato digitale. Per una sicurezza ulteriore, è anche possibile autenticare a sua volta il client presso il server, sebbene questa implementazione sia meno utilizzata a causa della maggiore complessità nel distribuire i certificati per ogni client.

Capitolo 2

Analisi dei requisiti e del problema

In questo capitolo si tratterà il tema della sicurezza come processo, con particolare attenzione al ruolo dell'*Incident Response*. Successivamente si entrerà nel merito del contesto specifico del progetto realizzato nell'ambito di questa tesi, con l'analisi dei requisiti e del problema, con riferimento ai prodotti di tipo *Next-generation firewall* e *Access Control Systems* che si prevede di gestire, con particolare enfasi sulle API REST che i vendor hanno implementato sui loro prodotti.

2.1 La sicurezza come processo

Affrontare le minacce moderne alla sicurezza di una rete richiede non solo l'utilizzo di strumenti adeguati come firewall, IDS e VPN (discussi nel capitolo precedente), ma anche che questi siano inseriti all'interno di un processo ben definito, con lo scopo di gestire i flussi di informazioni, rendendoli fruibili. Non è pensabile, infatti, che, una volta implementate le policy di sicurezza, queste risultino definitive e immutabili: è pressoché certo che alcune casistiche di attacco non siano state contemplate, specialmente per quanto riguarda attacchi basati su nuove vulnerabilità o sottovalutati in fase di analisi. Inol-

tre, non è possibile acquistare un singolo prodotto in grado di gestire tutta la sicurezza di una rete, tantomeno in maniera completamente automatica: si rende necessario combinare tra loro sia prodotti adibiti a funzioni differenti, sia molteplici competenze professionali da parte degli amministratori. [31]

2.1.1 Il processo

Il processo di gestione della sicurezza di un'infrastruttura deve, pertanto, essere ciclico, prevedendo le seguenti fasi:

- analisi dei rischi con identificazione del *threat model* specifico per il caso in esame;
- implementazione sui sistemi di quanto evidenziato come necessario nella prima fase;
- durante l'esercizio del sistema, continuo monitoraggio alla ricerca di eventuali violazioni;
- qualora si verificasse una violazione, rispondere adeguatamente per neutralizzarla.

A questo punto il ciclo si ripete, includendo nella fase di analisi quanto appreso durante la gestione della violazione. In figura 2.1 è illustrato questo processo.

2.1.2 Incident Response

Un *Incident Response Team* (IRT) si occupa dell'ultima parte del processo sopra descritto e funge da anello di congiunzione per l'inizio del ciclo successivo. In base allo standard NIST SP 800-61 [32], un IRT è in grado di:

- gestire attraverso un opportuno processo ogni sopraggiunto problema di sicurezza;

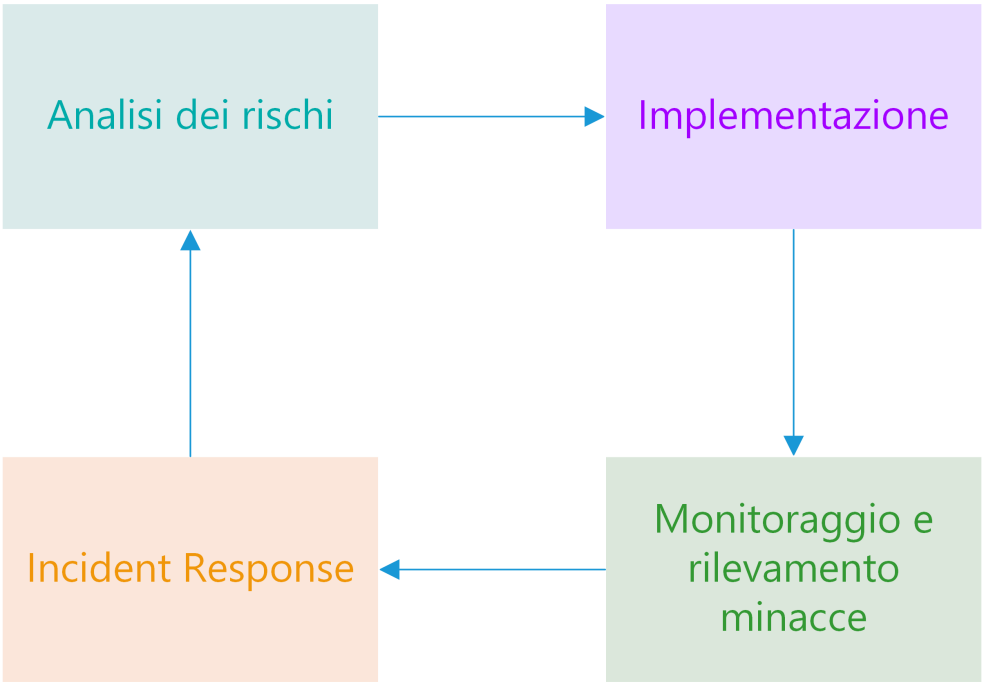

Figura 2.1: La sicurezza come processo

- ripristinare rapidamente ed efficientemente il corretto funzionamento dei sistemi, minimizzando i danni (perdita e/o furto di dati e/o interruzione dei servizi);
- utilizzare le informazioni raccolte durante tali fasi per una migliore gestione delle minacce future;
- gestire eventuali questioni legali che potrebbero sorgere a causa delle violazioni.

In particolare, è il terzo punto di questo elenco ad influire sul ciclo successivo. La parte di rilevazione delle minacce in corso viene effettuata tramite strumenti di analisi quali IDS e IPS (di cui si è discusso nella sezione 1.3) che attingono a molteplici fonti di informazioni, dal traffico di rete all'analisi dei log e della verifica dell'integrità dei software in esecuzione.
Nel momento in cui uno strumento automatico identifica una possibile violazione, spetta ad un analista approfondirla, stabilendone la legittimità e, in

caso positivo, la priorità di gestione. Le azioni di risposta ad una violazione di sicurezza includono procedure quali: [33]

- intraprendere azioni per proteggere i sistemi compromessi o minacciati dall'attaccante;
- fornire soluzioni o mitigazioni sulla base degli allarmi ricevuti (tanto da strumenti automatizzati quanto da fonti di *intelligence* terze);
- circoscrivere il perimetro della rete compromesso;
- impostare regole firewall per bloccare gli attacchi basati sulla rete;
- ripristinare i sistemi compromessi;
- sviluppare, in generale, qualunque altro tipo di risposta o workaround si ritenga opportuno.

2.2 Analisi dei requisiti

Essendo il processo sopra descritto articolato in più fasi, tutte richiedenti l'intervento umano, è lecito chiedersi se sia possibile automatizzare, almeno in parte, le interazioni uomo-uomo. L'opportunità è costituita dalle API REST (la cui architettura è stata discussa nella sezione 1.4) che i vendor di dispositivi di sicurezza hanno cominciato a sviluppare e mettere a disposizione degli amministratori negli ultimi anni, con lo scopo di consentire l'automatizzazione della configurazione dei loro apparati utilizzando strumenti terzi.
Ciò che ci si propone di realizzare nell'ambito di questa tesi è uno "strumento terzo" di questo tipo, che si interfacci con le soluzioni firewall in modo da fornire funzionalità aggiuntive.

2.2.1 Requisiti

Al fine di valutare la fattibilità e la convenienza della realizzazione di uno strumento di questo tipo, è necessario analizzare i requisiti che un'ipo-

tetica soluzione dovrebbe rispettare, ancora prima di entrare nel merito del problema specifico:

- estensibilità, ovvero deve essere possibile modificare, aggiungere o rimuovere funzionalità al software;
- dinamicità, ovvero deve essere possibile modificare la struttura di una rete senza richiedere una corrispettiva modifica progettuale al software;
- scalabilità, in modo da poter gestire anche infrastrutture complesse;
- robustezza, gestendo i possibili scenari di malfunzionamento senza provocare errori irreversibili;
- qualità del servizio adeguata al tipo di applicazione, minimizzando i tempi di attesa da parte degli utenti;
- facilità d'uso da parte degli amministratori, attraverso una gestione centralizzata di infrastrutture distribuite e un interfacciamento uniforme e di immediata comprensione;
- sicurezza: data la natura del sistema, sarà opportuno progettarlo e implementarlo secondo opportune *best practices* di sicurezza;
- interoperabilità, al fine di garantire flessibilità tra diverse implementazioni.

Questi requisiti sono in parte insiti nello stile architetturale REST. La soluzione che si andrà a progettare non dovrà essere in contrasto con essi.

2.2.2 Approfondimento dei requisiti

Estensibilità

Il software deve essere progettato in modo da consentirne agevolmente una successiva modifica. È altamente probabile che in futuro possano essere

implementate nuove funzionalità in aggiunta a quelle esistenti; altre potrebbero essere rimosse o modificate a causa di cambiamenti nelle API stesse dei vendor, tali da alterarne l'interfaccia o la semantica. La produzione di una buona documentazione sarà necessaria per consentire tali modifiche. Sarà quindi necessario sviluppare una serie di interfacce condivise, sul modello del paradigma *object oriented*, in modo da favorire l'estensibilità attraverso lo sviluppo di classi specifiche basate su esse. Inoltre, il software stesso deve essere progettato come modulo importabile all'interno di un software più grande.

Dinamicità

Poiché il software sviluppato dovrà interfacciarsi con reti eterogenee e potenzialmente molto estese, non deve dipendere dalla loro topologia per il funzionamento. Inoltre, dovrà essere adattabile a cambiamenti nell'infrastruttura di rete successivi alla prima installazione senza richiedere modifiche sul software stesso, bensì sulla sua sola configurazione.

Scalabilità

Le reti gestite spazieranno dalle poche decine alle migliaia di host. Nel caso di reti più estese, queste saranno protette tramite molteplici firewall, potenzialmente di vendor diversi. Per quanto riguarda i prodotti dello stesso vendor, possono essere presenti strumenti di gestione del vendor stesso che mirano a centralizzare la gestione di ambienti firewall distribuiti. Anche queste soluzioni dovranno essere dotati delle relative API REST per consentirne l'integrazione.

Robustezza

Trattandosi di integrazione tra sistemi diversi, sarà necessario garantire un certo livello di tolleranza ai guasti. In particolare, sarà utile porre attenzione al mantenimento della consistenza delle operazioni effettuate tramite

il software, adottando un modello transazionale analogo a quello utilizzato nei database relazionali. Un obiettivo sarà quello di garantire quanto più possibile le proprietà *ACID*:

- atomicità della transazione (esecuzione delle sotto-transazioni nella loro interezza o per nulla);
- consistenza dei dati (tutti i dati devono rispettare i vincoli di formattazione e non essere incompatibili con la loro semantica);
- isolamento di ogni transazione (esecuzione indipendente dalle altre e supportando l'accesso mutualmente esclusivo quando necessario);
- durabilità delle transazioni (i cambiamenti sono considerati validi una volta resi persistenti).

Qualità del servizio adeguata

Si deve cercare di minimizzare i tempi necessari a servire le richieste, sia per ridurre il tempo percepito dall'utente (considerato accettabile fintanto che rimane inferiore alla decina di secondi), sia per consentire il trasferimento dei pacchetti dati attraverso una rete senza che le connessioni scadano.

Facilità d'uso

Un software di questo tipo è pensato per essere utilizzato da un pubblico di analisti di sicurezza con competenze di programmazione. Sarà quindi necessario sviluppare un'interfaccia di utilizzo rispettosa delle buone pratiche di progettazione del software, in modo da renderne l'utilizzo intuitivo. Il numero di parametri richiesti per la configurazione sarà ridotto al minimo, ricorrendo anche a valori predefiniti (quando applicabili). Anche la produzione della documentazione ricoprirà un ruolo importante per la facilitazione della comprensione.

Sicurezza

Il sistema dovrà essere progettato e implementato utilizzando tecniche di *security by design*, quali una robusta crittografia nelle comunicazioni basata su certificati e autorità di certificazione e VPN, con lo scopo di ridurre la superficie di attacco, non esponendo i servizi direttamente sulla rete Internet. Solo gli utenti autorizzati dovranno essere in grado di modificare le configurazioni dei firewall e tutte le operazioni effettuate dovranno essere tracciate all'interno di opportuni log, in modo da consentire un successivo *auditing*. Si dovrà mitigare l'impatto degli eventuali danni in caso di intrusione attraverso la minimizzazione dei privilegi necessari all'esecuzione.

Interoperabilità

Si prevede che il software realizzato sarà installato all'interno di molteplici reti. Per questo motivo è ipotizzabile che, in seguito agli aggiornamenti che sicuramente coinvolgeranno il prodotto, coesisteranno più versioni del programma. Non si esclude che sarà installato su sistemi operativi diversi, sia Linux che Windows. Di conseguenza, sarà necessario garantire l'interoperabilità sia tra versioni diverse del software, sia relativamente all'esecuzione su più sistemi operativi. Il sistema dovrà essere progettato in modo da evitare quanto più possibile il ricorso a soluzioni proprietarie, favorendo invece standard riconosciuti. In questo modo sarà possibile adattare il sistema a scenari molteplici, integrandolo con strumenti di terze parti, qualora sia necessario.

2.3 Analisi del problema

Verrà ora analizzato il processo di sicurezza specifico per il dominio applicativo del progetto di tesi.

2.3.1 La situazione di partenza

All'interno del gruppo VEM, relativamente alla gestione del processo di sicurezza, sono presenti più team, ognuno dei quali si occupa di un aspetto specifico. In particolare, la parte di implementazione delle configurazioni sui firewall e la loro manutenzione è gestita dal *NOC* (Network Operation Center); quella di monitoraggio, rilevamento delle minacce e Incident Response è gestita dall'*IRT* (Incident Response Team) di Certego.
Al fine della gestione del processo sono necessarie continue interazioni tra i due team: nel momento in cui l'IRT individua un attacco in corso basato sulla rete, come ad esempio un malware penetrato all'interno della rete che contatta un server sotto il controllo dell'attaccante, è importante bloccarlo tempestivamente. Tuttavia, l'IRT non gestisce direttamente i firewall, per cui deve contattare il NOC, affinché quest'ultimo si incarichi di inserire una regola di blocco sul firewall interessato. In questa operazione sono coinvolti due utenti umani, che devono interagire compatibilmente con le priorità e gli impegni pianificati.
Si potrebbe incaricare l'IRT dell'inserimento delle nuove regole senza passare dal NOC ma questa strada non è percorribile, sia per ragioni amministrative (si cerca di limitare il numero di persone responsabili della gestione di un determinato apparato), sia per un problema di competenze (i membri dell'IRT non conoscono i dettagli delle configurazioni firewall dei clienti e le loro specificità). Inoltre, a complicare ulteriormente lo scenario, vi è il fatto che il panorama delle soluzioni installate presso i clienti è sempre più eterogeneo e integrato, imponendo una crescente specializzazione.
Di conseguenza, tra il rilevamento di un attacco in corso e la sua effettiva neutralizzazione può trascorrere parecchio tempo, specialmente in frangenti di carico di lavoro elevato per il NOC e/o al di fuori del normale orario di lavoro. La tempestività è invece di primaria importanza quando si tratta di Incident Response, in quanto all'aumentare del tempo durante il quale un attaccante ha accesso ad un sistema aumentano anche i danni potenziali che può compiere; un attacco bloccato sul nascere avrà maggiori probabilità di

essere inconcludente o di ridotto impatto.

2.3.2 Il software da sviluppare

Dati i requisiti e la situazione di partenza descritti, si intende sviluppare un'API (nome in codice **kAPI-royal**) che consenta all'IRT di applicare regole di blocco sui firewall dei clienti utilizzando un'interfaccia semplificata e uniforme, indipendente dai vendor, che a sua volta sfrutti le potenzialità delle API REST implementate dai vendor sui loro prodotti. In questo modo si eliminerà non solo la necessità di conoscere le specificità degli svariati prodotti di ogni vendor, ma anche quella di conoscere nel dettaglio la topologia delle reti dei clienti.
Le regole di blocco saranno basate alternativamente su indirizzo IP (ad esempio `192.168.1.120`), di sottorete (ad esempio `192.168.2.0/24`) o FQDN (*Fully Qualified Domain Name*, ad esempio `test.org`) del dominio che si intende bloccare.
In figura 2.2 è presente il diagramma UML dei casi d'uso che esemplifica le principali operazioni a cui si vuole abilitare gli analisti dell'IRT. Contestualmente viene visualizzato il ruolo del NOC, che si occuperà della prima installazione e della manutenzione ordinaria, senza necessità di intervenire manualmente ogni volta che l'IRT desidera agire sulle regole di blocco.

2.3.3 I dispositivi da supportare

Nell'ambito di questa tesi, ci si focalizzerà sul supporto dei dispositivi utilizzati attualmente dai vari clienti, tralasciando gli altri. La struttura modulare del progetto dovrà comunque consentire l'estensibilità, aggiungendo in futuro il supporto a nuovi prodotti e vendor.
In generale, saranno gestiti dispositivi classificabili in due categorie: i next-generation firewall (descritti nella sezione 1.1.1) ed i sistemi per il controllo degli accessi. Verranno ora analizzati i prodotti specifici che saranno presi in esame.

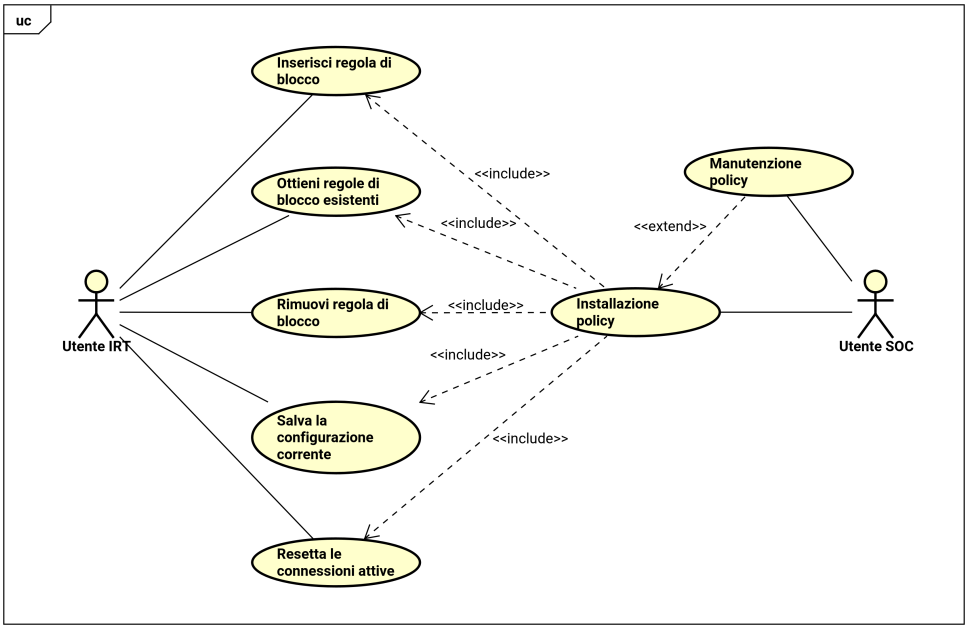

Figura 2.2: Diagramma dei casi d'uso

Cisco ASA

Cisco *Adaptive Security Appliance* (ASA) [34] è la prima soluzione next-generation firewall proposta dall'azienda californiana. Si tratta di un prodotto solido, anche se la curva di apprendimento per utilizzarlo completamente è piuttosto ripida, in quanto manca un'interfaccia grafica web, rendendo necessario destreggiarsi con la linea di comando. È presente un plugin aggiuntivo per implementare le API REST, che consentono la gestione delle funzioni principali, come alternativa alla linea di comando.

Cisco Firepower

Cisco *Firepower* [35] è il nuovo sistema operativo eseguibile su hardware ASA. Introduce una nuova interfaccia grafica e la possibilità di centralizzare la gestione di un'infrastruttura basata su di esso tramite *Firepower Management Center* (FMC). I firewall controllati sono chiamati *Firepower Threat Defense* (FTD). Si tratta di un prodotto ancora non completamente maturo,

in quanto alcune funzionalità, come le API REST, sono implementate solo in parte e le prestazioni non sono ancora ottimali.

Fortinet FortiManager

FortiGate [36] è la soluzione next-generation firewall di Fortinet. Può essere utilizzato come dispositivo a sé stante o, nel caso di infrastrutture firewall distribuite, in maniera centralizzata tramite *FortiManager* [37]. Nel caso di studio preso in esame ci si limiterà a quest'ultimo caso d'uso. Sono presenti API REST piuttosto complete nelle funzionalità offerte, anche se non del tutto rispettose degli standard de facto nella progettazione di servizi web *RESTful*.

Check Point R80

Anche Check Point offre un next-generation firewall di sua produzione, al momento giunto alla versione R80. [38] Basato su *Red Hat Enterprise Linux*, è anch'esso pensato per infrastrutture firewall distribuite. Le API REST sono piuttosto complete e ben documentate.

Cisco ISE

A differenza dei prodotti descritti finora, Cisco *Identity Services Engine* (ISE) [39] non è un next-generation firewall, bensì uno strumento scalabile e adattabile di controllo degli accessi per reti di dimensioni medio-grandi e grandi basato sul contesto (i privilegi sono assegnabili agli host dinamicamente in base al ruolo, alla locazione e alla configurazione software). A differenza di quanto progettato per i firewall, qui non si bloccherà il traffico in transito, ma si opererà per mettere in quarantena i client infetti della rete, in modo da contenere la diffusione dei malware ed evitare che questi inviino dati ai loro creatori.

2.3.4 Stato dell'arte

Dato il problema presentato, è opportuno esplorare il mercato alla ricerca di eventuali soluzioni a questo problema: se già ne esistesse una adeguata, non sarebbe necessario progettare una soluzione ad hoc.
In primo luogo, si nota come il problema sia piuttosto specifico e legato al contesto applicativo dell'azienda: questa difficoltà di comunicazione si verifica solo nei casi in cui esistano più team che si occupano di aspetti diversi dello stesso cliente. Per questo motivo, eseguendo ricerche estese sia a soluzioni proprietarie, sia a soluzioni open source, non è stato possibile individuare un prodotto completo in grado di adempiere alle funzionalità richieste.
Si è quindi reso necessario estendere il raggio della ricerca, mirando ad individuare non più soluzioni complete, ma, più in generale, prodotti parziali che interagiscono con le API REST dei vendor, senza essere necessariamente pensati per il contesto applicativo preso in esame, ai quali potersi ispirare. Fortunatamente, sono stati individuati alcuni progetti open source relativamente a Cisco Firepower, Fortinet FortiManager, Check Point R80 e Cisco ISE. [40, 41, 42, 43, 44, 45] Si potrà attingere a tali fonti per valutare alcuni esempi di utilizzo delle API.

2.3.5 Test plan

Considerato l'ambito applicativo, è necessario che il software sia ben testato, in modo da prevenire malfunzionamenti inattesi durante l'utilizzo in produzione che potrebbero causare danni ai clienti: poiché si interagirà direttamente con le policy dei firewall, nel caso peggiore si potrebbe mandare *offline* tutta la rete. Si rende, quindi, necessaria la definizione di un processo di controllo della qualità volto a mitigare questo rischio.
Sarà opportuno definire *unit tests* che andranno a verificare la correttezza del comportamento delle singole componenti del sistema e *integration tests* che esamineranno le interazioni dei componenti tra loro. Tali test saranno eseguiti in maniera automatica prima del rilascio di ogni nuova versione del

software, in modo da intercettare eventuali bug.

Per quanto riguarda gli ambienti di test (schematizzati in figura 2.3), si utilizzerà nelle prime fasi un ambiente dedicato e virtuale, separato dai sistemi in produzione e simile ad essi, in modo da non causare danni in caso di problemi. Quando il comportamento del software in questo ambiente sarà considerato corretto, si passerà ai test in produzione presso alcuni clienti selezionati. Questo passaggio è necessario poiché nell'ambiente virtuale non è possibile riprodurre efficacemente la complessità dei sistemi di tutti i clienti, che in alcuni casi arrivano a contemplare migliaia di regole sui firewall. Infine, se anche questo test avrà successo, si rilascerà il prodotto presso tutti i clienti.

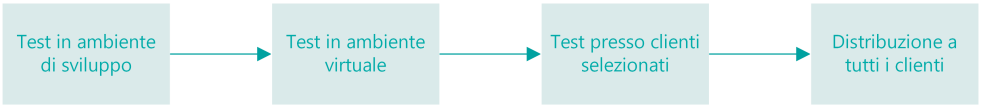

Figura 2.3: Ambienti di test

Capitolo 3

Progettazione e implementazione

A fronte del problema descritto nel capitolo precedente, saranno ora esaminate le fasi di progettazione e implementazione del sistema software descritto (**kAPI-royal**), motivando le scelte intraprese.

3.1 Progettazione

In questa sezione saranno analizzate le scelte progettuali effettuate, tenendo conto delle specificità dei prodotti dei vari vendor.

3.1.1 Paradigmi di sviluppo

Il software da realizzare deve fornire un'interfaccia che medi l'accesso ai dispositivi controllati. Considerata l'affidabilità richiesta nelle comunicazioni e la necessità di ottenere i risultati di un'elaborazione prima di proseguire con le attività successive, si è scelto di utilizzare uno stile di comunicazione sincrono. Sarà fondamentale l'integrazione con il software preesistente, mantenendo uno stile modulare: è consigliabile adottare lo stesso paradigma del software principale, ovvero quello *object-oriented*. Questo implica lo svolgimento di un'attività di mediazione con lo stile architetturale REST

utilizzato dalle API dei firewall. Ad ogni modo, internamente può essere utilizzato qualsiasi paradigma si ritenga opportuno, purché l'interfaccia esposta sia di tipo *object-oriented*.
In particolare, dove possibile, si è valutato di utilizzare il paradigma funzionale, poiché le funzioni definite in questo modo sono prive di *side-effect*, rendendo più semplice verificare la presenza di bug e migliorando l'efficienza.

3.1.2 Astrazioni e formato dei dati

Un problema fondamentale quando si tratta l'integrazione tra servizi diversi è quello del formato utilizzato per rappresentare i dati: ogni vendor utilizza quello che ritiene più opportuno, impedendo un accesso unificato. Inoltre, ogni prodotto utilizza internamente delle astrazioni software proprie. L'API che si intende realizzare dovrà fronteggiare questo problema e distillare le molte specificità in un'interfaccia condivisa.
In figura 3.1 è mostrato il diagramma UML di sequenza che illustra il ruolo di mediazione del software realizzato con le API REST dei vendor.

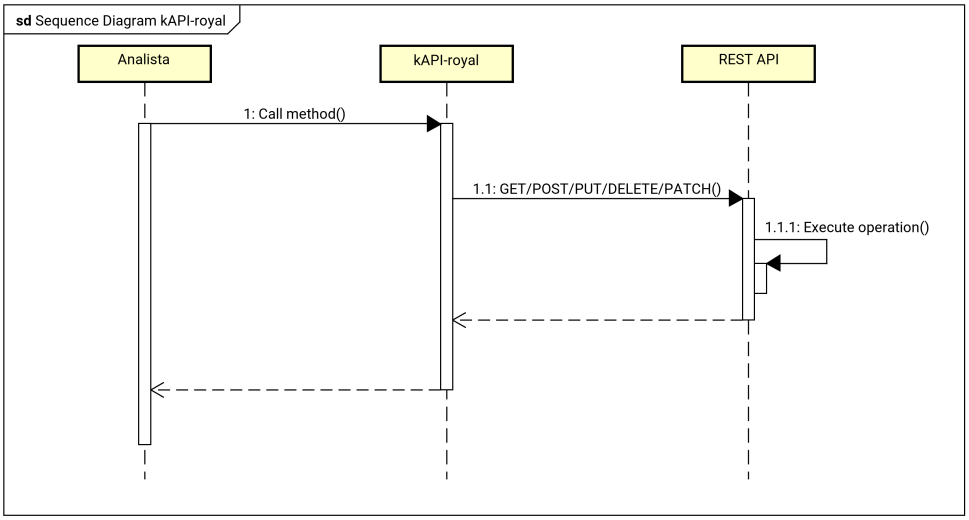

Figura 3.1: Ruolo di mediazione del software realizzato

Per quanto riguarda le regole di blocco, si è scelto di sfruttare l'astrazione a

"oggetti" e "gruppi di oggetti" implementata da tutti i firewall: un oggetto è l'elemento base di costruzione delle regole all'interno di un firewall e rappresenta un dato considerato elementare. Appartengono a questa categoria gli indirizzi IP, di sottorete, FQDN (*Fully Qualified Domain Name*) e, dipendentemente dal singolo vendor, anche altri. Questi possono essere raggruppati, per convenienza, all'interno di uno o più gruppi. I vendor stabiliscono non solo quali elementi possano costituire un oggetto ma anche quali di questi siano in grado di esistere solo come parte di un gruppo o a sé stanti e quali oggetti possano coesistere all'interno di uno stesso gruppo e quali no.
Al fine di unificare le divergenze tra i vari prodotti, si è reso necessario rendere trasparenti agli utenti questi dettagli implementativi, dando origine ad un'ulteriore astrazione nella rappresentazione dei dati, questa volta unificata. Il software sviluppato si occuperà di tradurre di volta in volta le istruzioni dell'utente nel formato appropriato.

3.1.3 Definizione delle interfacce

Come anticipato nella sezione 2.3, i dispositivi supportati apparterranno essenzialmente a due categorie: i next-generation firewall (Cisco ASA, Cisco Firepower, Fortinet FortiManager, Check Point R80) e i sistemi di controllo degli accessi (Cisco ISE). Di conseguenza, serviranno due interfacce distinte, al cui interno saranno definiti i metodi comuni a tutte le classi che le implementeranno. In figura 3.2 sono presenti i diagrammi UML delle due interfacce, chiamate rispettivamente `KAPIFirewall` e `KAPIAccessControl`. Le classi che implementeranno le funzionalità dei singoli prodotti estenderanno da esse.
Relativamente ai firewall, le operazioni da implementare sono:

- `push_block_list`: imposta una serie di indirizzi IP o di sottorete per essere bloccati dal firewall;
- `push_block_list_fqdn`: imposta una serie di FQDN per essere bloccati dal firewall;

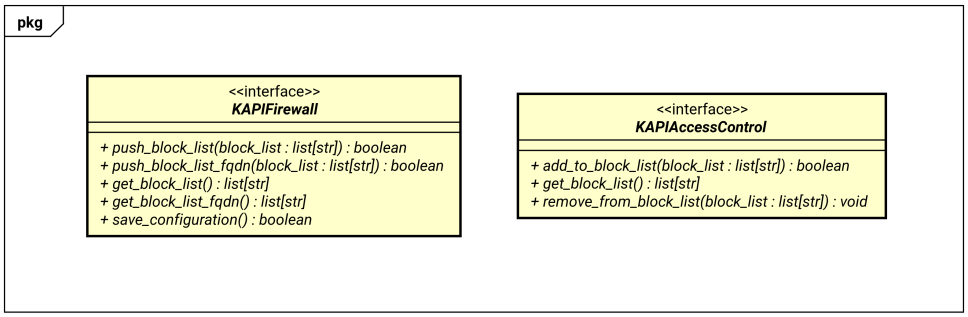

Figura 3.2: Diagramma UML delle interfacce

- `get_block_list`: ottiene la lista degli indirizzi IP o di sottorete attualmente bloccati dal firewall;
- `get_block_list_fqdn`: ottiene la lista degli FQDN attualmente bloccati dal firewall;
- `save_configuration`: salva la configurazione attualmente impostata, rendendola persistente e distribuendola ai firewall associati.

Relativamente ai sistemi per il controllo degli accessi, le operazioni da implementare sono:

- `add_to_block_list`: aggiunge indirizzi IP o MAC alla lista di quelli da bloccare;
- `get_block_list`: ottiene la lista di indirizzi MAC attualmente bloccati;
- `remove_from_block_list`: rimuove una lista di indirizzi MAC dalla lista di quelli bloccati.

I dati contenenti un numero variabile di elementi sono sempre trasferiti come liste. La notazione utilizzata per indicare le sottoreti è *CIDR* (l'indirizzo di base della rete seguito dalla *subnet mask*, ad esempio `192.168.2.0/24`).

3.1.4 Specificità dei singoli prodotti

Le funzioni definite nelle interfacce non sono le sole disponibili nei dispositivi: quelle specifiche per i singoli prodotti saranno implementate nelle relative classi. In alcuni casi, come in Cisco Firepower, alcune funzionalità non sono ancora presenti ma è noto che saranno implementate in futuro. A volte, funzionalità apparentemente equivalenti su più prodotti sono in realtà diverse. Nella tabella 3.1 è presente uno schema delle funzionalità implementate sui firewall, divise per dispositivo.

Funzione	**ASA**	**FMC**	**Check Point**	**FortiManager**
push_block_list	✓	✓	✓	✓
push_block_list_fqdn	✓	×	✓	✓
get_block_list	✓	✓	✓	✓
get_block_list_fqdn	✓	×	✓	✓
save_configuration	✓	✓	✓	✓
reset_connections	✓	×	✓	×

Tabella 3.1: Funzionalità comuni implementate sui firewall

Reset delle connessioni aperte

Una funzione degna di nota, ma presente solo in alcuni prodotti, è il reset delle connessioni TCP aperte: nel momento in cui si inserisce una nuova regola per impedire la comunicazione verso un dato indirizzo, le connessioni precedentemente stabilite rimangono valide fino alla loro chiusura. Per questo è utile poterne forzare la terminazione. Su alcuni prodotti (come Cisco ASA) è possibile effettuare questa operazione a sé stante, richiamando l'apposita funzione, mentre su altri (Check Point R80) l'operazione è eseguita automaticamente all'installazione delle policy.

Feed Intelligence

Alcuni prodotti includono una funzione di *Feed Intelligence* (o *Threat Feed*) che consiste nell'importazione automatica di un elenco di risorse da bloccare tramite un file esterno a piacere che gli analisti possono popolare, senza necessità di intervenire sulle policy e rimuovendo il bisogno di accedere ai firewall. [46, 47, 48]
Tuttavia, il supporto è al momento molto frammentario, poiché solo alcuni dispositivi lo implementano e la semantica varia da un vendor all'altro (ad esempio Fortinet applica queste regole esclusivamente al traffico HTTP e HTTPS). Inoltre, l'aggiornamento del *feed* avviene a intervalli di tempo prestabiliti e non sempre configurabili, rendendolo poco adatto alla risposta tempestiva ad un attacco, quanto piuttosto ad un'attività di prevenzione (popolandolo periodicamente con indirizzi noti per distribuire malware).
Si è comunque deciso di esplorare la funzionalità, il cui attuale supporto è indicato in tabella 3.2.

Funzione	**ASA**	**FMC**	**Check Point**	**FortiManager**
Feed con IP/subnet	×	✓	✓	✓
Feed con DNS	×	✓	×	✓

Tabella 3.2: Funzionalità "Feed Intelligence" implementate sui firewall

Applicazione delle modifiche

Tutti i prodotti in esame includono l'operazione di "salvataggio delle modifiche effettuate", ma il funzionamento può essere molto diverso a seconda dei casi: nei firewall *stand-alone* (non distribuiti, ad esempio Cisco ASA), quando si applica una modifica, questa è immediatamente attiva e l'operazione di salvataggio ha lo scopo di rendere tali modifiche persistenti in caso di riavvio; nei firewall distribuiti, le modifiche vengono inviate all'interfaccia di *management* (ad esempio FortiManager) ma non risultano attive fino a

quando non vengono installate sui firewall associati. In quest'ultimo caso, è importante applicare le modifiche in mutua esclusione con eventuali altri amministratori, per evitare l'installazione di policy inconsistenti.

Monitoraggio dei task

Le operazioni lanciate tramite le API REST dei vendor adottano uno stile di comunicazione sincrono ovvero il chiamante rimane in attesa fino a quando non viene restituita una risposta. Normalmente questo non costituisce un problema, poiché vengono eseguite rapidamente. Costituiscono un'eccezione quelle la cui durata non è trascurabile, come ad esempio l'installazione delle policy (che può arrivare a richiedere anche diversi minuti). In questi casi l'API restituisce immediatamente un identificativo associato al task vero e proprio invece dell'esito dell'operazione: la chiamata non risulta bloccante, ma, per conoscere l'esito, è necessario interrogare successivamente il firewall tramite *polling*.

3.1.5 RADIUS Change of Authorization e 802.1X

Poiché Cisco ISE non è un firewall, bensì un sistema di controllo degli accessi a supporto di un'infrastruttura di rete, relativamente alle regole di blocco non si può utilizzare la stessa strategia adottata per gli altri prodotti. Viene quindi in aiuto la funzionalità *Adaptive Network Control* (ANC) [49], che ha lo scopo di mettere in quarantena i dispositivi delle reti gestite interagendo con i singoli apparati di rete che le compongono (switch e access point). Questi, ogni volta che un dispositivo desidera connettersi alla rete, si affidano ad ISE per l'accettazione della richiesta e l'assegnazione di privilegi di accesso. In aggiunta a questo, viene in aiuto *RADIUS Change of Authorization* (CoA), un'estensione proprietaria di Cisco al protocollo RADIUS che consente ad ISE di modificare gli attributi di *autenticazione*, *autorizzazione* e *accounting* (AAA) di una sessione attiva. [50, 51] Per utilizzare queste tecnologie, tutti i dispositivi coinvolti devono essere compatibili tra loro.

IEEE 802.1X è uno standard basato su EAP (*Extensible Authentication Protocol*) [52] che consente l'autenticazione sicura dei client in una rete, oltre a proteggere i dati in transito mediante crittografia. Gli attori coinvolti sono il *supplicant* (entità che desidera connettersi), l'*authenticator* (dispositivo di rete attraverso il quale il client accede alla rete) e l'*authentication server* (server RADIUS). Poiché non tutti i dispositivi supportano 802.1X, esiste un estensione proprietaria di Cisco denominata MAB (*MAC Authentication Bypass*) che, limitatamente a questi casi, ripiega su un'autenticazione alternativa basata su nome utente, password e indirizzo MAC. [53]

3.1.6 Eccezioni

In aggiunta alle eccezioni predefinite del linguaggio di programmazione e delle librerie associate, sarà utile definirne alcune personalizzate per meglio rappresentare le situazioni di errore specifiche del dominio applicativo:

- `UnexpectedStatusCodeError`, generata quando una chiamata HTTPS all'interfaccia REST di un apparato restituisce un codice di errore;
- `AddressParsingError`, generata nel caso l'utente non utilizzi il formato corretto per indicare indirizzi IP, di sottorete, FQDN o MAC;
- `TooWideSubnetError`, generata quando si tenta di bloccare una sottorete più ampia rispetto ad una soglia prefissata (per evitare modifiche accidentali nocive);
- `GroupNotFoundError`, generata quando sui firewall non esiste un gruppo in cui inserire gli elementi da bloccare (può succedere quando il nome del gruppo specificato non è corretto o se l'apparato non è stato predisposto per l'amministrazione tramite il software);
- `TimestampError`, utilizzata dagli apparati che richiedono un *timestamp* in ogni richiesta, nel caso questo non fosse valido (ad esempio se troppo vecchio o riferito ad un tempo futuro).

3.1.7 Sicurezza

Il software interagirà con elementi critici dell'infrastruttura di sicurezza, potenzialmente in grado di rendere inagibili le reti dei clienti. Questo rende necessario adottare particolari accorgimenti di *security by design*, al fine di mitigare questo rischio:

- limitare l'uso di librerie software di terze parti a quelle strettamente necessarie e, anche tra queste, privilegiare le più note e testate;
- tutti gli accessi dovranno essere registrati e si dovrà mantenere traccia di tutte le operazioni eseguite;
- per l'accesso si utilizzeranno account utente ad hoc e non quelli generici per l'amministrazione dell'apparato;
- i privilegi assegnati a tali account saranno limitati a quelli strettamente necessari per svolgere le attività richieste (modifica di oggetti sui firewall e installazione delle policy);
- l'accesso da parte degli analisti ai server in cui sarà installato il software avverrà tramite VPN per evitare attacchi di tipo *man-in-the-middle*;
- le comunicazioni tra il software e le API dei dispositivi avverranno attraverso un canale sicuro, tramite crittografia (*HTTPS*) e fisicamente all'interno della rete del cliente;
- dovranno essere mantenuti backup regolari delle configurazioni degli apparati per ripristinare il funzionamento in seguito ad un attacco o un malfunzionamento.

3.1.8 Deployment

Per quanto riguarda l'installazione del software, sarà eseguita sugli IDS, attualmente utilizzati dagli analisti come punto di accesso alle reti dei clienti. In questo modo non sarà necessario utilizzare un ulteriore server dedicato. In figura 3.3 è presente lo schema di deployment progettato. Il NOC

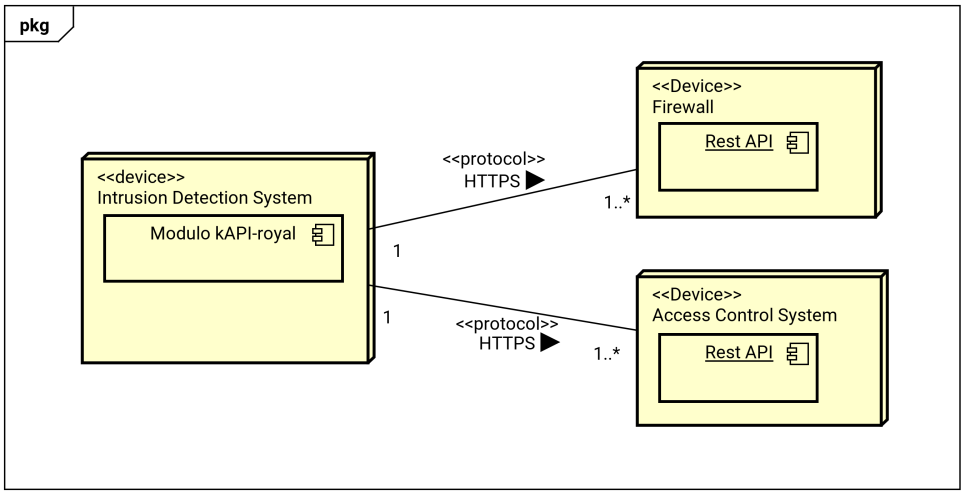

Figura 3.3: Diagramma UML di deployment

si occuperà della prima configurazione degli apparati da controllare mentre l'IRT dell'installazione del modulo software sull'IDS e dell'attività di *Incident Response*.

3.2 Implementazione

In questa sezione sarà descritto il processo di implementazione, partendo dal linguaggio di programmazione, passando dall'implementazione delle classi associate ai dispositivi da controllare e quelle ausiliarie, per concludere con alcuni esempi di utilizzo del software sviluppato.

3.2.1 Linguaggio di programmazione

Per lo sviluppo, si richiede l'utilizzo di un linguaggio di programmazione che offra un buon supporto ai paradigmi citati e sia facilmente integrabile con il software preesistente.
La scelta è ricaduta su **Python** [54], linguaggio *object-oriented* di alto livello che rispetta i requisiti e consente di sviluppare moduli software facilmente importabili all'interno di altri programmi. I vantaggi di Python includono

anche il supporto nativo a tipi di dati complessi come i dizionari e le tuple. Inoltre, è multi-piattaforma ovvero è possibile sviluppare, testare ed eseguire il software su molteplici sistemi operativi.
Python è al momento uno dei linguaggi più utilizzati ed è in ascesa da anni. Nel momento in cui viene scritto questo documento (Febbraio 2019) è al terzo posto dopo Java e C. [55]
La documentazione del codice sarà prodotta con l'ausilio di *Sphinx*, un generatore di documentazione basato sulla sintassi *reStructuredText*. [56]

3.2.2 Implementazioni delle interfacce

Poiché in Python non esiste la distinzione tra "interfacce" e "classi astratte", esistendo solo queste ultime, le interfacce definite in sezione 3.1.3 sono state implementate direttamente come classi astratte, includendo i campi comuni alle classi figlie. Come si può notare, è sempre richiesto l'indirizzo del server a cui collegarsi (`server_url`) e le credenziali di accesso, basate su `username` e `password`. In aggiunta, è presente un riferimento ad un oggetto di tipo `Logger` (realizzato con il pattern *Singleton*), che avrà lo scopo di mantenere traccia di tutte le operazioni effettuate tramite l'API e gli eventuali errori, per consentire un successivo *auditing*.
In figura 3.4 è presente il diagramma delle classi relativo ai firewall, mentre in figura 3.5 quello relativo ai sistemi per il controllo degli accessi.
Ogni apparato ha le proprie specificità: in alcuni l'autenticazione è basata solo su nome utente e password, in altri su *token* generato dinamicamente; nelle infrastrutture firewall distribuite esistono i concetti di "dominio amministrativo", cioè un gruppo di firewall logicamente connessi che condividono un set di regole, e di "sessione" ovvero più amministratori possono operare contemporaneamente mantenendo separate le loro modifiche; alcune API REST utilizzano un ampio set di comandi HTTP, mentre alcuni si limitano ad usare "POST", demandando al corpo della richiesta la rappresentazione dei dati. Nelle singole classi degli apparati sono stati aggiunti anche metodi per fornire l'accesso ad alcuni dati potenzialmente utili a scopo di debug.

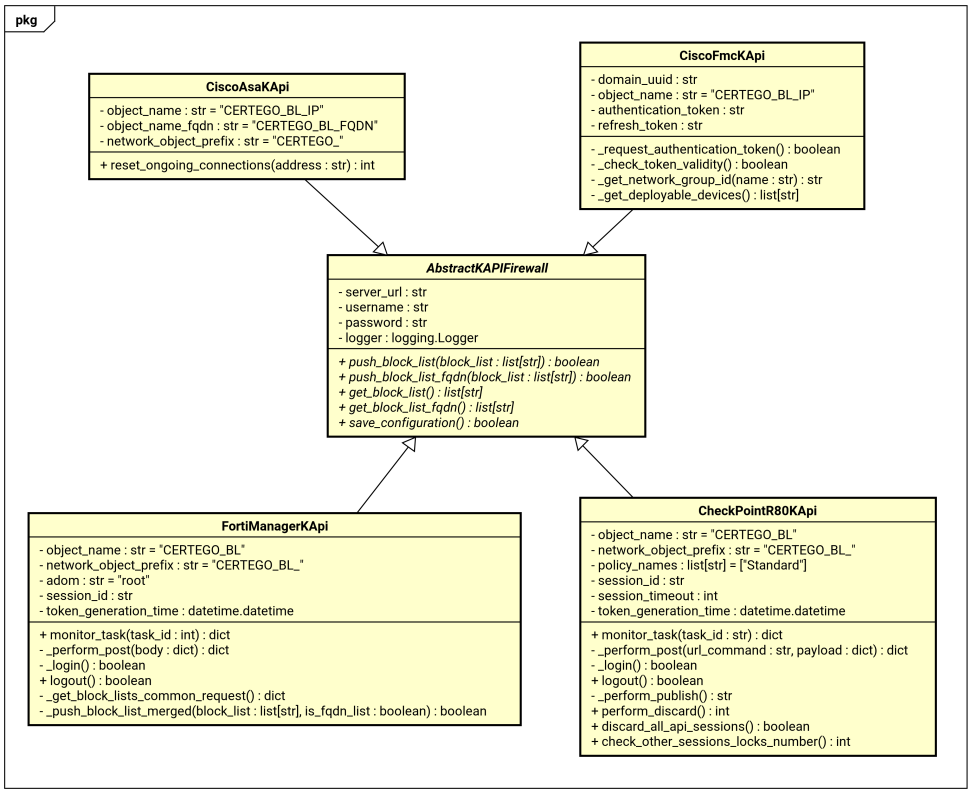

Figura 3.4: Diagramma UML dei firewall

Un valore di default è stato impostato per tutti i parametri per i quali è possibile, richiedendo di modificarli solo in circostanze straordinarie, in accordo con il requisito di facilità d'uso.

3.2.3 Metodi di utilità

Separatamente dalle classi associate ai singoli prodotti, è stata definita una serie di metodi di utilità comuni a più prodotti:

- `is_subnet`, che esegue il *parsing* di un indirizzo per stabilire se è valido e se si tratti di un singolo indirizzo IP o una sottorete;
- `is_mac_address`, analogo al precedente, ma pensato per gli indirizzi MAC;

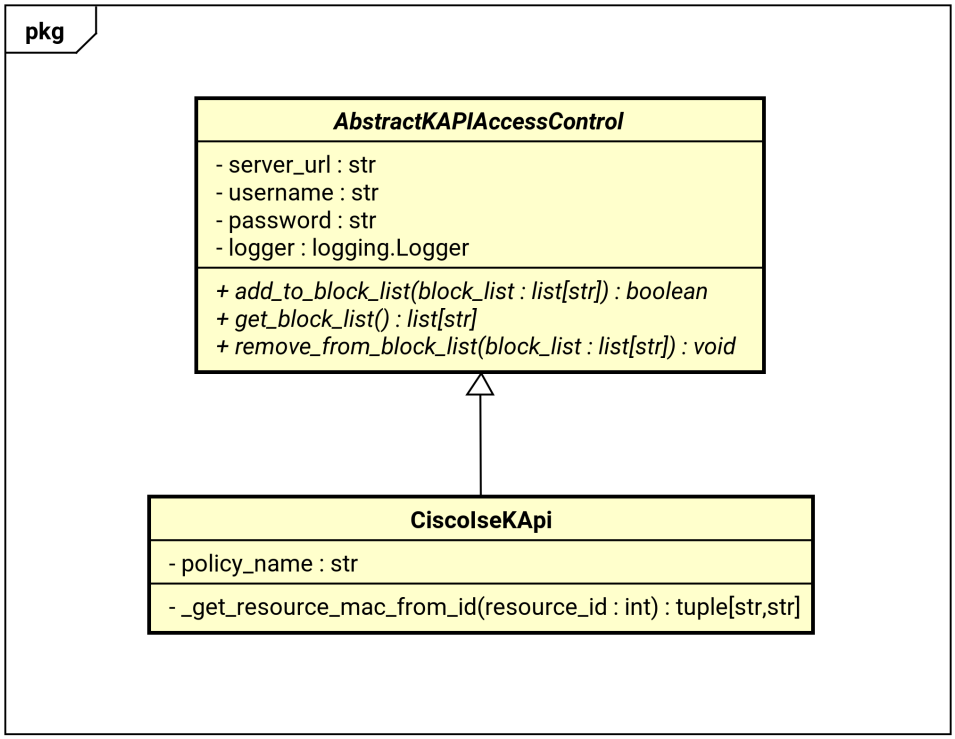

Figura 3.5: Diagramma UML dei sistemi per il controllo degli accessi

- `unsupported_method`, utilizzato come decoratore per i metodi che, pur facendo parte di un'interfaccia, non sono al momento supportati dal produttore;
- `write_block_list_network_feed_file`, che consente di generare una lista di indirizzi IP e/o di sottorete per le funzionalità di *Feed Intelligence* (sezione 3.1.4);
- `write_block_list_dns_feed_file`, analogo al precedente, ma pensato per nomi di dominio.

È stato implementato anche un semplice server HTTP con capacità di filtraggio degli indirizzi dei client pensato per pubblicare eventuali liste di indirizzi da bloccare con la funzionalità di *Feed Intelligence*, per gli apparati che la utilizzano.

3.2.4 Esempi di utilizzo del software

In questa sezione verranno mostrati alcuni esempi di utilizzo di **kAPI-royal**, richiamabili tramite una console Python o uno script.
In questo primo esempio la lista corrente di indirizzi IP e di sottorete bloccati su un firewall Cisco ASA sarà sovrascritta con un singolo indirizzo IP. Successivamente saranno terminate le connessioni attualmente aperte verso quell'indirizzo e la configurazione sarà resa persistente.

```
from cisco.CiscoAsaKApi import CiscoAsaKApi
kapi_asa = CiscoAsaKApi("https://192.168.1.1", "user", "pass")
kapi_asa.push_block_list(["10.0.0.1"])
kapi_asa.reset_ongoing_connections("10.0.0.1")
kapi_asa.save_configuration()
```

Il secondo esempio è simile al primo, ma il nuovo elemento è una sottorete e sarà aggiunto agli elementi esistenti invece di sovrascriverli.

```
from cisco.CiscoAsaKApi import CiscoAsaKApi
kapi_asa = CiscoAsaKApi("https://192.168.1.1", "user", "pass")
previous_block_list = kapi_asa.get_block_list()
previous_block_list.append("10.0.0.0/24")
kapi_asa.push_block_list(previous_block_list)
kapi_asa.reset_ongoing_connections("10.0.0.0/24")
kapi_asa.save_configuration()
```

Nel terzo esempio verranno visualizzati gli FQDN attualmente bloccati su un firewall Check Point. Si noti che, utilizzando il costrutto `with`, al termine dell'operazione la sessione sarà automaticamente chiusa. In alternativa è possibile richiamare il metodo `logout` manualmente.

```
from checkpoint.CheckPointR80KApi import CheckPointR80KApi
with CheckPointR80KApi("https://192.168.1.1/", "user", "pass",
    policy_names=["Standard"]) as kapi_chkp:
  print(kapi_chkp.get_block_list_fqdn())
```

Nel quarto e ultimo esempio due indirizzi MAC saranno aggiunti a quelli da bloccare su un'installazione Cisco ISE.

```
1  from cisco.CiscoIseKApi import CiscoIseKApi
2  kapi_ise = CiscoIseKApi("https://192.168.1.1:9060", "user", "pass",
       "CERTEGO_ANC_QUARANTINE")
3  kapi_ise.add_to_block_list(["33-33-33-33-33-33", "55:55:55:55:55:55"])
```

Conclusioni

Nell'ambito di questa tesi è stata realizzato un software che consente di ridurre i tempi di risposta agli attacchi informatici automatizzando le interazioni tra gli analisti dell'*Incident Response Team* e gli apparati degli utenti (firewall e dispositivi di controllo degli accessi). In base ai requisiti e al dominio applicativo, sono state inizialmente approfondite le tecnologie associate per poi passare alla progettazione della soluzione che è stata curata in tutte le sue fasi, fino alla messa in produzione sui sistemi dei clienti. Le funzionalità implementate saranno commercializzate con il nome di *Tactical Response* e sarà prevista l'integrazione con *PanOptikon*, la piattaforma di *Incident Response* attualmente utilizzata.

Per quanto riguarda gli sviluppi futuri, sono pensabili scenari in cui persino gli utenti finali, attraverso un'interfaccia grafica opportunamente semplificata, saranno in grado di intervenire sulle regole di blocco impostate all'interno delle loro reti. La struttura modulare del progetto consente la successiva introduzione di nuove opzioni, come il supporto a nuovi apparati e la modifica di quelli esistenti.

Potranno eventualmente essere implementate anche funzionalità che in questa fase si è volutamente deciso di tralasciare, essendo ritenute poco utili per il contesto applicativo, tra cui il *certificate pinning* e la validazione dei nomi di dominio. Il primo consiste nell'accettazione di uno specifico certificato SSL/TLS da parte del server, per ridurre la probabilità di attacchi *man-in-the-middle* e si è deciso di non implementarlo poiché per l'accesso alle reti dei clienti si passa attraverso percorsi considerati sicuri, tramite VPN (de-

scritte nella sezione 1.2) e VLAN (di cui si è parlato nella sezione 1.1.3); la seconda prevede un'attività di verifica dell'esistenza dei nomi di dominio da bloccare prima di inserirli in lista, per prevenire errori accidentali durante la digitazione: questa opzione è stata scartata per evitare di doversi affidare a server DNS esterni, considerando anche che un eventuale errore in questa fase difficilmente può provocare danni e può essere corretto in modo rapido.
Sulle infrastrutture firewall distribuite è stato riscontrato un problema nella gestione degli accessi concorrenti da parte di più amministratori: tramite le API REST non è sempre possibile stabilire con esattezza quali altre modifiche siano in attesa di essere installate sugli apparati. Di conseguenza, se un amministratore umano dimenticasse di accedere in modalità esclusiva o lasciasse alcune modifiche alle configurazioni in sospeso, esisterebbe il rischio concreto che il software, al momento dell'inserimento o della rimozione di regole di blocco, installi involontariamente anche queste. La raccomandazione attuale è di evitare l'installazione automatica delle policy qualora esistano potenziali conflitti.
L'obiettivo della tesi è stato raggiunto con successo: ora gli analisti dispongono di uno strumento che li svincola dalla mediazione continua da parte del NOC ogni volta che necessitano di terminare un attacco in corso basato sulla rete. Da parte del NOC è sufficiente una configurazione iniziale degli apparati per predisporli all'amministrazione remota.

Appendice A

Guide alla configurazione degli apparati

In questa appendice sono presenti le guide alla prima configurazione di alcuni degli apparati supportati nell'ambito della tesi per poter essere amministrati tramite il software sviluppato. Nelle procedure descritte saranno utilizzati dei dati di esempio impiegando, dove possibile, quelli predefiniti: in questo modo sarà minimizzato il numero di parametri necessari agli analisti dell'IRT.

A.1 Cisco ASA

Su Cisco ASA è innanzitutto necessario installare le API REST tramite un modulo software aggiuntivo, poiché la funzionalità non è presente di default. Non tutte le versioni di ASA sono supportate. [57, 58]
Verranno ora elencati i passaggi per l'installazione.

1. Attivare il modulo "REST API" seguendo le istruzioni nel documento di installazione ufficiale. [59]

2. Tramite la console Cisco, abilitare l'accesso amministrativo HTTPS e limitarlo all'indirizzo IP del sistema sui cui sarà installato il software

di controllo (in questo esempio `10.0.0.1`) sull'interfaccia appropriata (in questo caso `inside`):

```
http server enable
http 10.0.0.1 255.255.255.255 inside
```

3. Creare le credenziali amministrative per consentire l'accesso tramite l'API. Nel caso sia impostata l'autenticazione locale, devono essere definite nella configurazione di ASA tramite il comando `username`:

   ```
   username certego password Passw0rd
   ```

   Se l'autenticazione è mediata da un server *RADIUS* o *TACACS+*, devono essere definite sull'opportuno *identity store*. Inoltre, nel caso di *TACACS+*, l'API REST di ASA rivolgerà al server AAA (*autenticazione*, *autorizzazione* e *accounting*) delle richieste di autorizzazione comandi per conto di un utente fittizio denominato `enable_1`: sarà necessario definire anch'esso in un opportuno *identity store* (la password non sarà mai utilizzata) e autorizzarlo all'interno delle policy. In figura A.1 è presente un esempio di questo tipo su ISE 2.2.

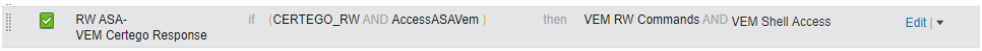

Figura A.1: Esempio di policy di autorizzazione su ISE

4. Creare i gruppi di oggetti per ospitare le destinazioni da bloccare (uno per indirizzi IP/sottorete e l'altro per FQDN) e inserire all'interno di entrambi un oggetto provvisorio, poiché i gruppi non possono essere vuoti:

   ```
   object-group network CERTEGO_BL_IP
   network-object host 169.254.0.1
   exit
   ```

```
object network CERTEGO_BL_dummydomain.local
fqdn v4 dummydomain.local
exit
object-group network CERTEGO_BL_FQDN
network-object object CERTEGO_BL_dummydomain.local
exit
```

5. Affinché il blocco per FQDN sia utilizzabile, è necessario che i server DNS siano specificati in configurazione, come mostrato nell'esempio sottostante (i parametri varieranno in funzione dell'installazione specifica):

```
DNS server-group DefaultDNS
name-server 8.8.8.8
name-server 8.8.4.4
domain-name example.com
```

6. Applicare i gruppi di oggetti creati alle opportune liste di accesso sul firewall per fare in modo che gli oggetti all'interno di essi vengano bloccati bidirezionalmente sulle interfacce desiderate:

```
access-list INSIDE line 1 extended deny ip any
    object-group CERTEGO_BL_IP
access-list INSIDE line 2 extended deny ip any
    object-group CERTEGO_BL_FQDN
access-list OUTSIDE line 1 extended deny ip
    object-group CERTEGO_BL_IP any
access-list OUTSIDE line 2 extended deny ip
    object-group CERTEGO_BL_FQDN any
```

7. Comunicare agli analisti dell'IRT i seguenti parametri: nome del cliente, URL per l'accesso all'API, nome utente e password. Eventuali altri parametri sono richiesti solo nel caso divergano da quelli predefiniti.

8. Verificare il corretto esito della procedura chiedendo agli analisti di popolare la lista degli indirizzi bloccati tramite l'API. Le modifiche applicate sono visibili dal contenuto dei gruppi sopra definiti, tramite i comandi:

```
show run object-group id CERTEGO_BL_IP
show run object-group id CERTEGO_BL_FQDN
```

A.2 Check Point R80

Nel caso di Check Point R80 non è necessario installare software aggiuntivo, poiché il supporto alle API REST è presente nativamente. Verranno ora elencati i passaggi per l'installazione eseguiti dall'interfaccia di gestione.

1. Creare un utente e un profilo amministrativo dedicati dal menù `SmartConsole` → `Manage & Settings` → `Permissions & Administrators` e impostarne i privilegi, come mostrato nelle figure A.2 e A.3. I permessi da assegnare al profilo sono elencati in tabella A.1, divisi per sezione.

2. Attivare le API REST dal menù `SmartConsole` → `Manage & Settings` → `Blades`, come mostrato in figura A.4 e verificare dall'interfaccia web che l'indirizzo IP del sistema su cui sarà installato il software di controllo sia presente tra quelli abilitati nella sezione `User Management` → `GUI Clients` (figura A.5). A questo punto riavviare il servizio dell'API dall'interfaccia a linea di comando (l'operazione può durare alcuni minuti):

```
fwtest> api restart
2018-Nov-28 10:07:23 - Stopping API...
2018-Nov-28 10:07:31 - API stopped successfully.
2018-Nov-28 10:07:31 - Starting API...
. . . . . .
```

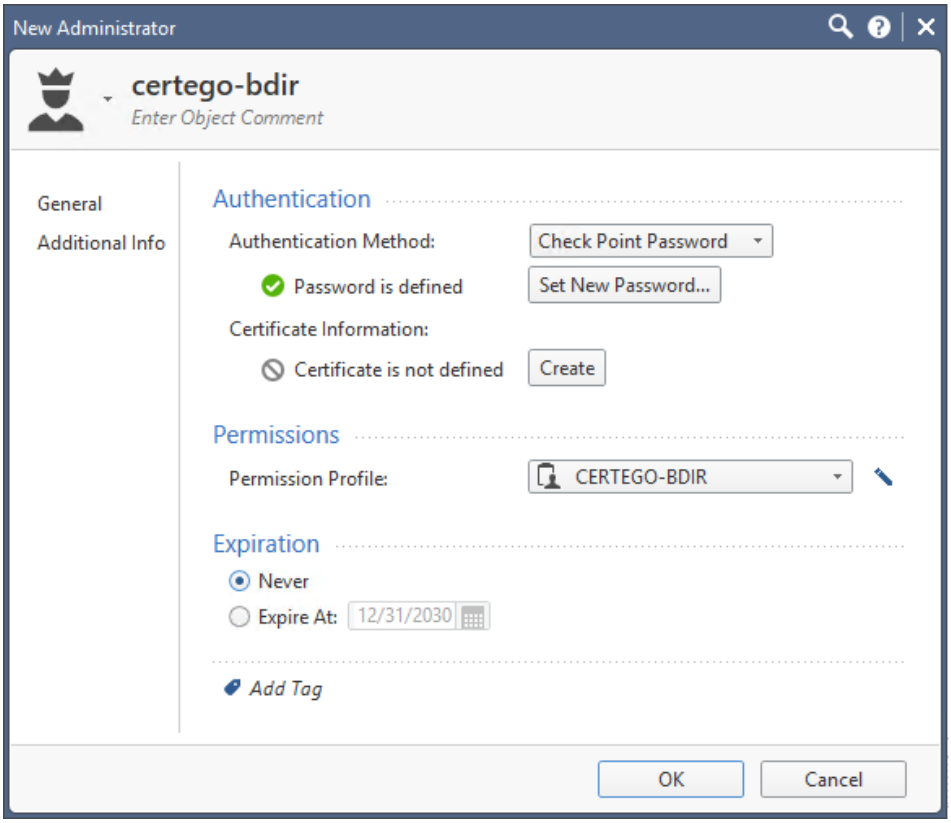

Figura A.2: Check Point - Creazione utente

```
2018-Nov-28 10:10:07 - API started successfully.
```

In caso di problemi di interfacciamento con l'API, è possibile verificare lo stato del server tramite il comando `api status`.

3. Creare un oggetto di tipo "Network Group" denominato `CERTEGO_BL` (figure A.6 e A.7), inserire al suo interno un oggetto provvisorio (un gruppo non può essere vuoto) denominato `CERTEGO_BL_169.254.0.1` contenente l'indirizzo IP `169.254.0.1` (figure A.8 e A.9).

4. Definire, in cima ai policy package rilevanti, due regole che blocchino il gruppo precedentemente creato, rispettivamente come sorgente e come destinazione (su modello dell'esempio in tabella A.2).

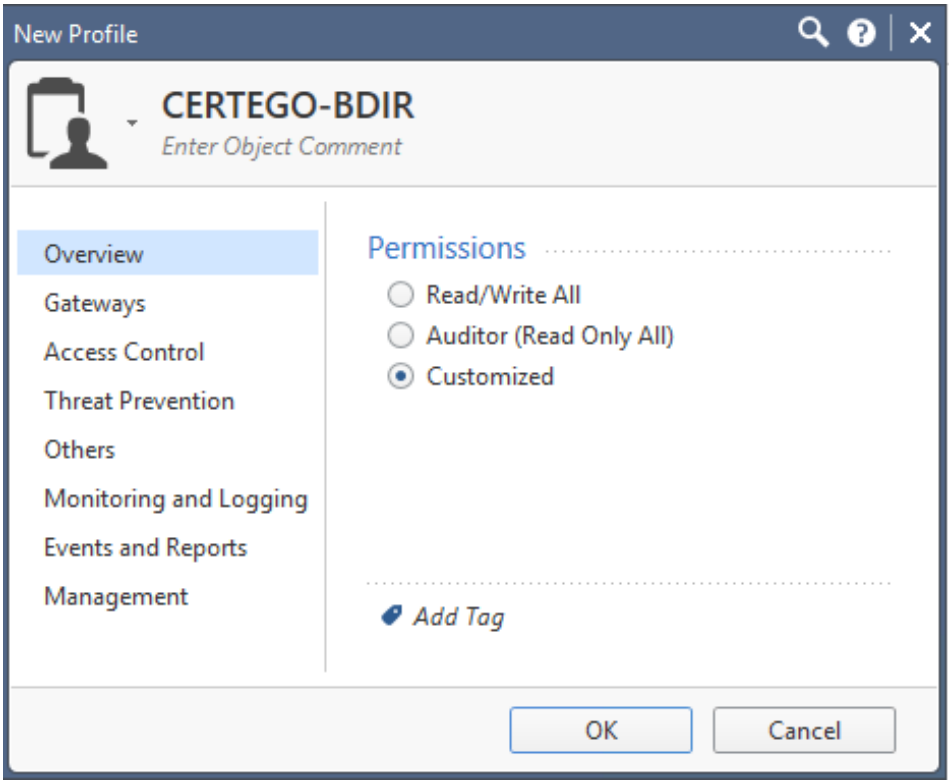

Figura A.3: Check Point - Creazione profilo amministrativo

5. Comunicare agli analisti dell'IRT i seguenti parametri: nome del cliente, URL per l'accesso all'API, nome utente e password. Eventuali altri parametri sono richiesti solo nel caso divergano da quelli predefiniti.

6. Verificare il funzionamento dell'integrazione chiedendo agli analisti di popolare la lista di blocco. Le modifiche applicate sono visibili dal contenuto dei gruppi sopra definiti, attraverso il menù `Object Explorer`.

Permesso	**Valore**
Overview → Permissions	Customized
Access Control → Show Policy → Firewall	✓
Access Control → Access Control Objects and Settings	Write
Access Control → Install Policy	✓
Threat Prevention → Policy Rules	Read
Threat Prevention → Policy Exceptions	Read
Threat Prevention → Profiles	Read
Threat Prevention → Protections	Read
Threat Prevention → Settings	Read
Others → Common Objects	Write
Others → Check Point Users Database	Read
Management → Management API Login	✓

Tabella A.1: Permessi minimi per il profilo amministrativo su Check Point

Name	**Source**	**Destination**	**Action**	**Track**
Certego TR Outbound	Any	CERTEGO_BL	Drop	Log
Certego TR Inbound	CERTEGO_BL	Any	Drop	Log

Tabella A.2: Check Point - Definizione regole

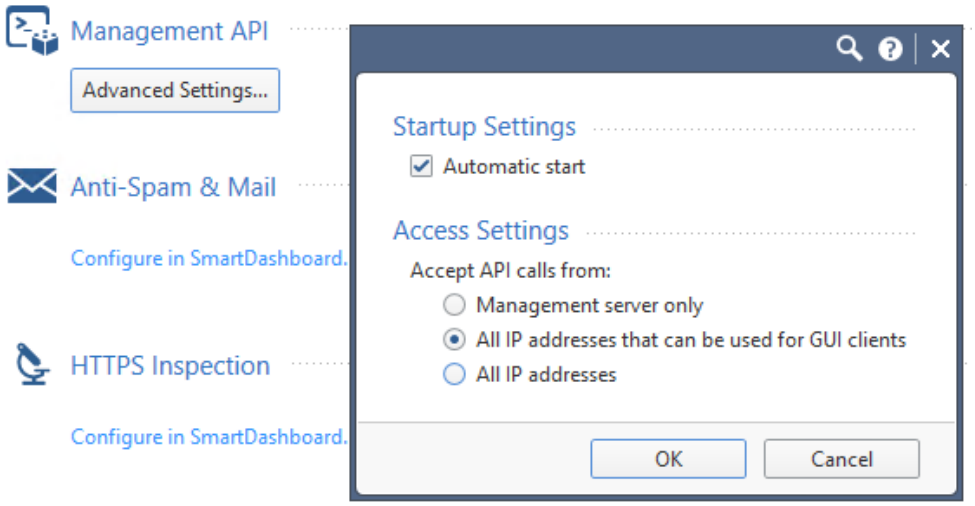

Figura A.4: Check Point - Attivazione API (1)

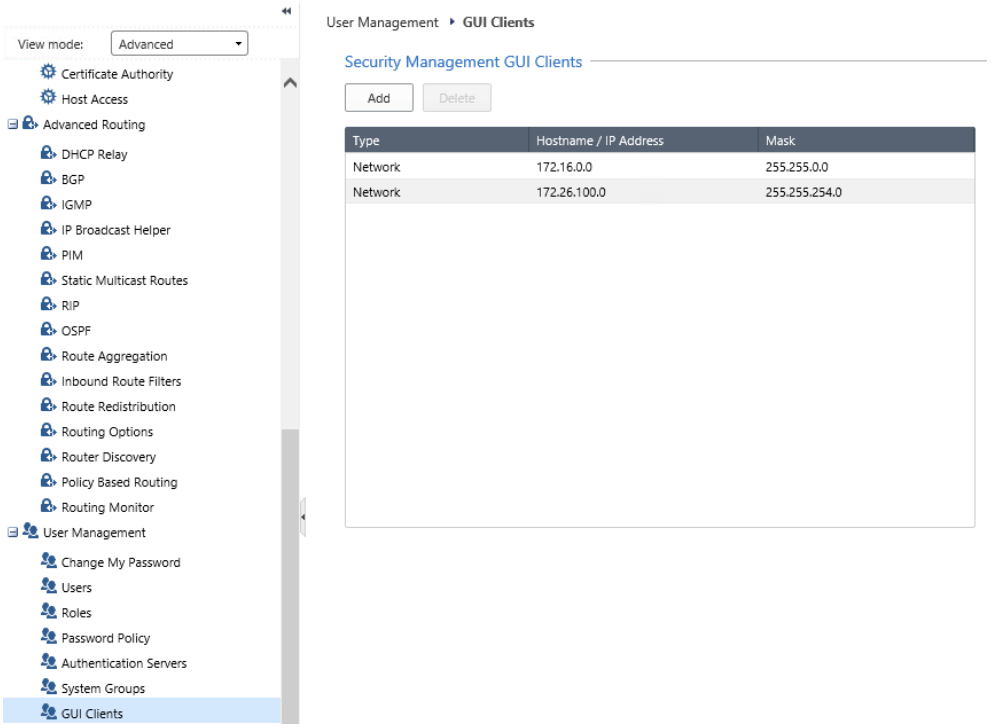

Figura A.5: Check Point - Attivazione API (2)

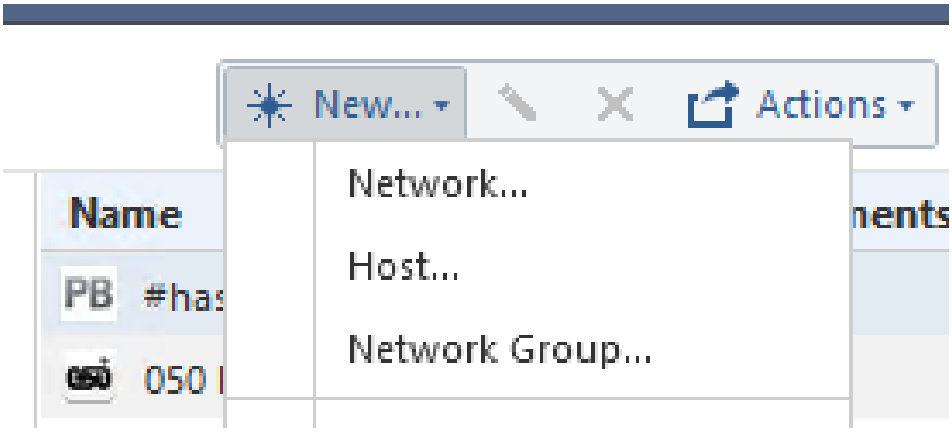

Figura A.6: Check Point - Creazione gruppo (1)

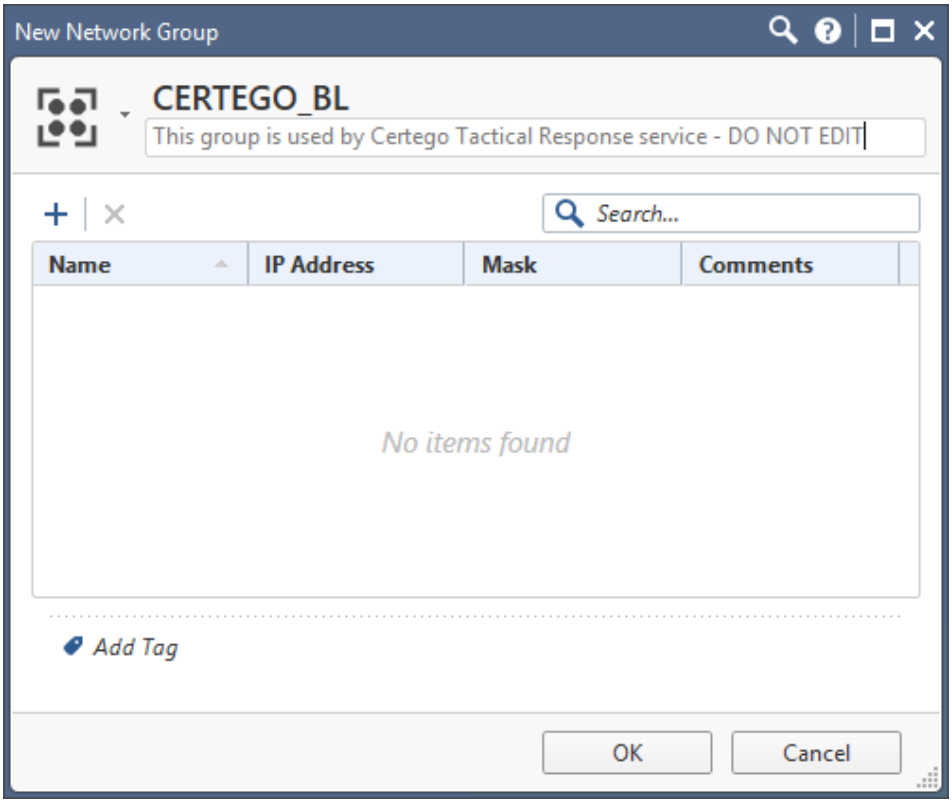

Figura A.7: Check Point - Creazione gruppo (2)

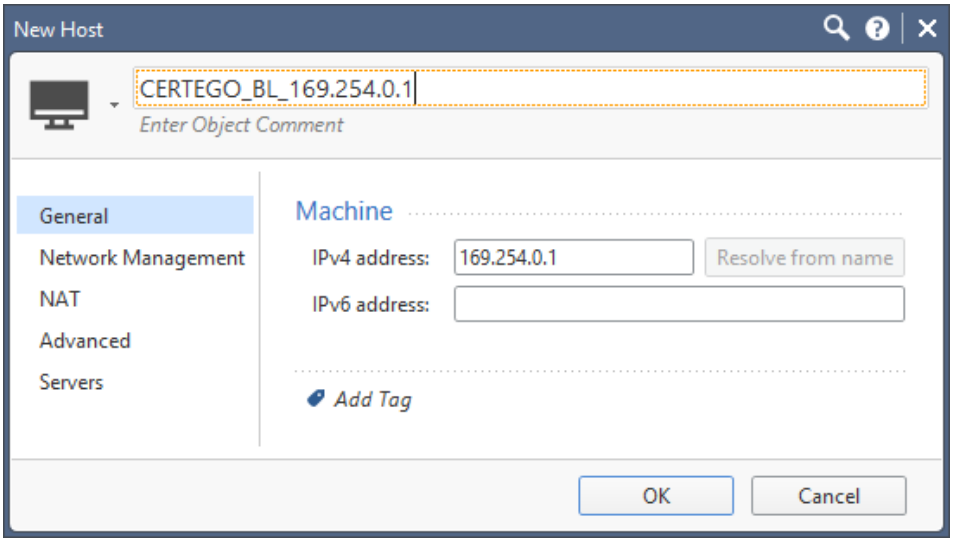

Figura A.8: Check Point - Popolamento gruppo (1)

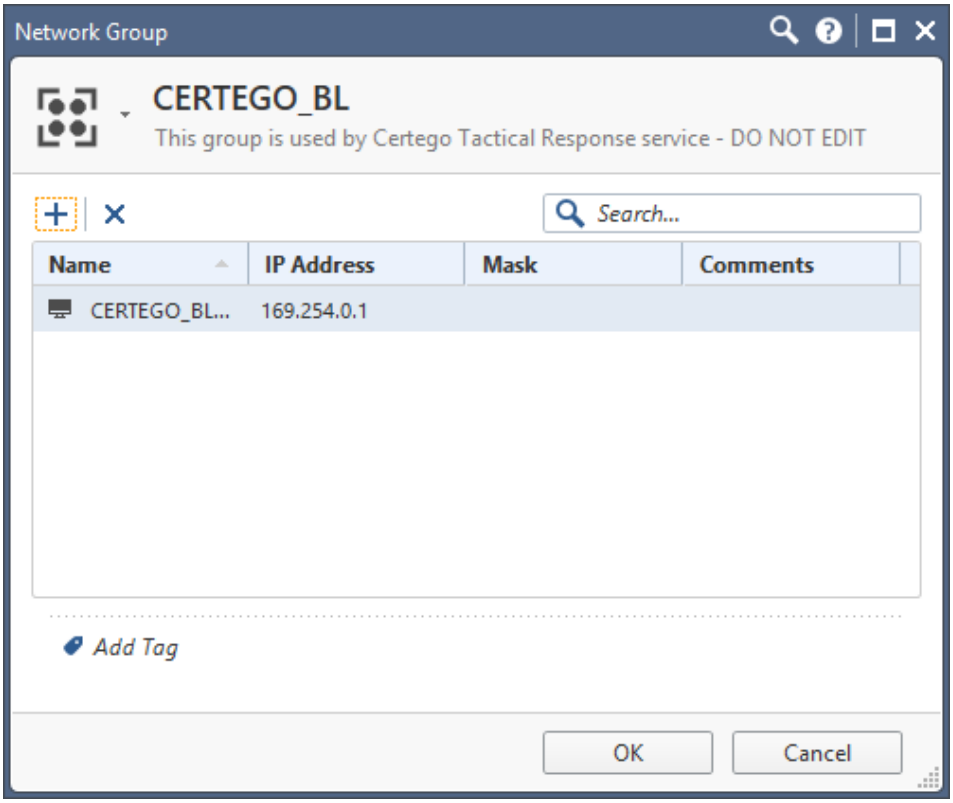

Figura A.9: Check Point - Popolamento gruppo (2)

A.3 Cisco ISE

L'integrazione con Cisco ISE è disponibile dalla versione 2.0 e successive. Per poter utilizzare le funzioni del software sviluppato è necessaria una licenza di tipo *Plus*. Inoltre, l'installazione deve essere integrata con i *Network Access Device* dell'infrastruttura ed è necessario il supporto alla funzionalità *RADIUS CoA* (descritta in sezione 3.1.5). Verranno ora elencati i passaggi per l'installazione eseguiti dall'interfaccia di gestione.

1. Attivare l'interfaccia *External RESTful Services* (ERS) dal menù `Administration` → `System` → `Settings` → `ERS Settings`, come mostrato in figura A.10. Tale operazione non richiede il riavvio di ISE.

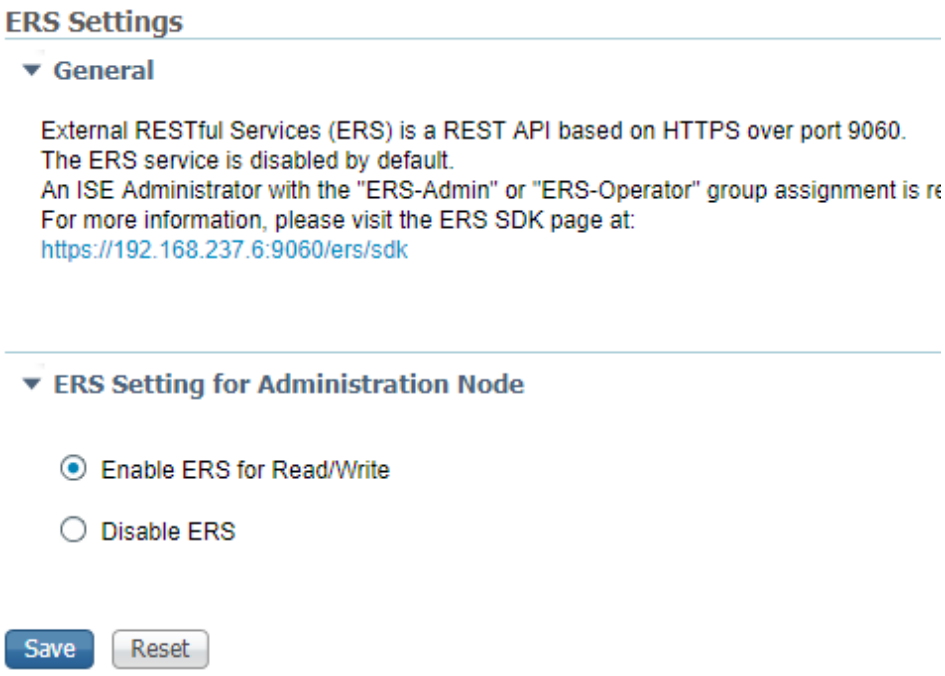

Figura A.10: Configurazione di ISE - Abilitazione ERS

2. Creare un utente amministrativo per l'accesso alle API dal menù `Administration` → `System` → `Admin Access` → `Admin Users` e assegnarlo al gruppo `ERS Admin` (figura A.11).

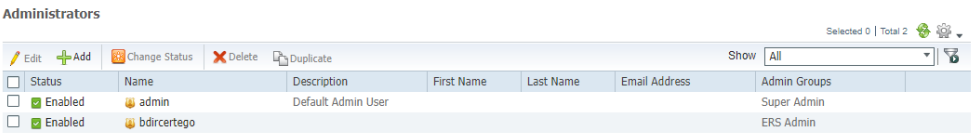

Figura A.11: Configurazione di ISE - Creazione utente

3. Creare una policy ANC (funzionalità descritta in sezione 3.1.5) di quarantena dal menù `Operations` → `Adaptive Network Control` → `Policy List` e chiamarla `CERTEGO_ANC_QUARANTINE` (figura A.12).

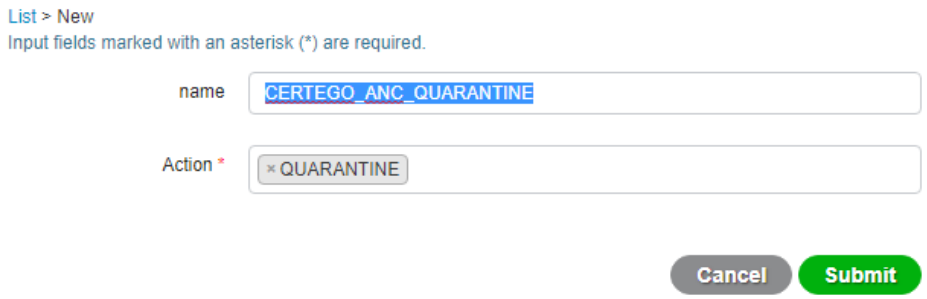

Figura A.12: Configurazione di ISE - Policy ANC

4. Creare una "global exception" all'interno delle policy di autorizzazione, che indichi come trattare le richieste di accesso dei client in quarantena. A titolo di esempio, le alternative possono essere: rifiutarle o presentare al cliente una notifica di blocco mediante un *captive portal*. La prima opzione è più semplice, poiché è sufficiente creare una policy di autorizzazione di tipo "Exception"; la seconda sarà dettagliata a parte, in sezione A.3.1.

5. Comunicare agli analisti i seguenti parametri: nome del cliente, URL per l'accesso all'API, nome utente e password. Eventuali altri parametri sono richiesti solo nel caso divergano da quelli predefiniti.

6. Eseguire un test chiedendo agli analisti dell'IRT di bloccare un indirizzo tramite l'API. Su ISE, l'elenco dei client bloccati è visibile

dal menù `Operations` → `Adaptive Network Control` → `Endpoint Assignment`.

A.3.1 Implementazione di un captive portal

Verranno ora descritti i passaggi per implementare un *captive portal su ISE* da utilizzare congiuntamente alla funzionalità ANC.

1. Creare un portale di tipo "Hotspot", abilitando la sola pagina di successo e personalizzandone il contenuto grafico dal menù `Work Centers` → `Guest Access` → `Portals & Components` → `Guest Portals` (figure A.13 e A.14).

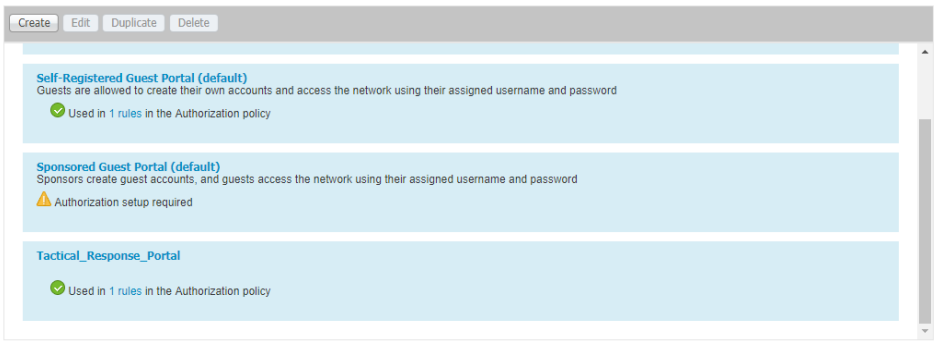

Figura A.13: Configurazione di ISE - Captive Portal (1)

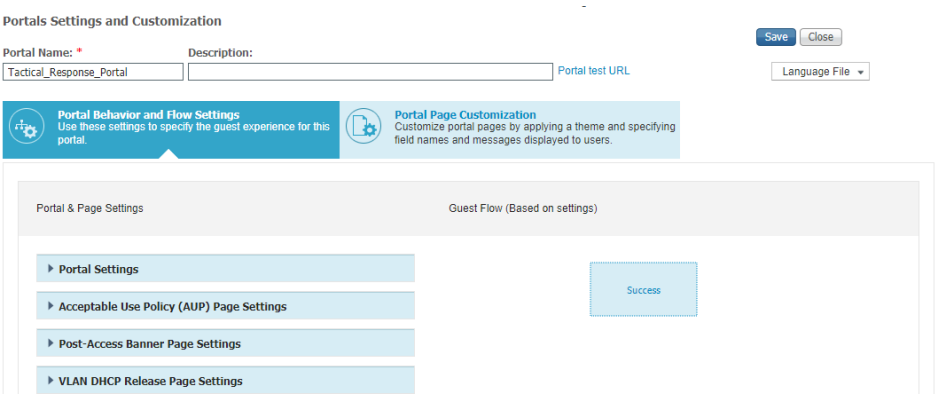

Figura A.14: Configurazione di ISE - Captive Portal (2)

2. Creare un "Authorization Profile" che imponga la ridirezione sul portale appena creato dal menù `Policy` → `Policy Elements` → `Results` → `Authorization` (figura A.15). La "pre-authentication-ACL", denominata `TACTICAL_RESPONSE_ACL`, deve permettere il traffico DNS (porta 53/UDP) e verso il portale hotspot di ISE (di default, sulla porta 8443/TCP).

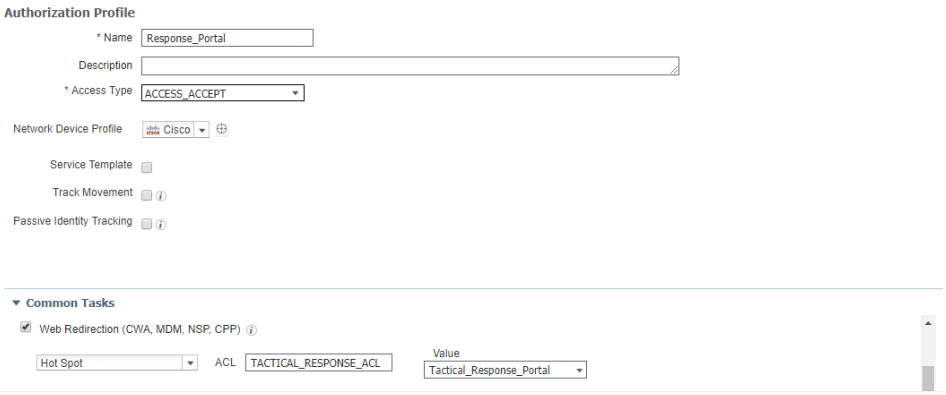

Figura A.15: Configurazione di ISE - Captive Portal (3)

3. Applicare al portale un certificato valido per il FQDN contattato dai client che saranno rediretti.

4. Creare una policy di autorizzazione di tipo "Exception" per redirigere i client in quarantena sul portale (figura A.16).

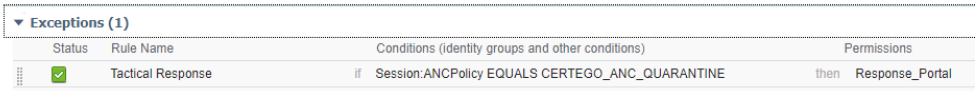

Figura A.16: Configurazione di ISE - Captive Portal (4)

Bibliografia

[1] Rob Sobers, *60 Must-Know Cybersecurity Statistics for 2019*, Varonis blog, sito web, `https://www.varonis.com/blog/cybersecurity-statistics/`.

[2] Bradley Mitchell, *The Definition and Purpose of a Network Firewall*, 19 Dicembre 2018, sito web, `https://www.lifewire.com/definition-of-firewall-817568`.

[3] S. Bellovin e W. Cheswick, *Network Firewalls*, IEEE Communication Magazine, Settembre 1994.

[4] Judy Thompson-Melanson, *Learn About Firewall Evolution from Packet Filter to Next Generation*, Juniper Networks, 2015, ISBN 978-1-941441-00-8.

[5] K. Scarfone e P. Hoffman, *Guidelines on Firewalls and Firewall Policy*, NIST Special Publication SP 800-41-1 - Settembre 2009.

[6] Tony Northrup, *Firewalls*, Microsoft Technet, 29 Giugno 2009, sito web, `https://technet.microsoft.com/it-it/library/cc700820.aspx`.

[7] *How Firewalls Work*, TechWeb, Boston University, sito web, `https://www.bu.edu/tech/about/security-resources/host-based/intro/`.

[8] Liviu Arsene, *The Evolution of Firewalls: Past, Present & Future*, 27 Gennaio 2015, sito web, `https://www.informationweek.com/partner-perspectives/bitdefender/`

the-evolution-of-firewalls-past-present-and-future/a/d-id/1318814.

[9] *What is Firewall Security?*, SecureWorks, 15 Novembre 2016, sito web, https://www.secureworks.com/blog/firewall-security.

[10] Del Smith CCNA, *Understand the evolution of firewalls*, 13 agosto 2002, sito web, https://www.techrepublic.com/article/understand-the-evolution-of-firewalls/.

[11] IETF RFC 1928, *SOCKS Protocol Version 5*, Marzo 1996, sito web, https://www.ietf.org/rfc/rfc1928.txt.

[12] pfSense, sito web, https://www.pfsense.org/.

[13] Marcus J. Ranum, *Thinking About Firewalls*, 1993, sito web, http://www.vtcif.telstra.com.au/pub/docs/security/ThinkingFirewalls/ThinkingFirewalls.html.

[14] Jürgen Schmidt, *A death blow for PPTP*, 26 Settembre 2012, sito web, http://www.h-online.com/security/features/A-death-blow-for-PPTP-1716768.html.

[15] IETF RFC 2661, *Layer Two Tunneling Protocol "L2TP"*, Agosto 1999, sito web, https://www.ietf.org/rfc/rfc2661.txt.

[16] OpenVPN, sito web, https://openvpn.net/.

[17] Joseph Davies, *IPv6 traffic over VPN connections*, Microsoft Technet, Luglio 2007, sito web, https://web.archive.org/web/20120615203602/http://lab.technet.microsoft.com/en-us/magazine/cc138002.

[18] IETF RFC 2828, *Internet Security Glossary*, Maggio 2000, sito web, https://www.ietf.org/rfc/rfc2828.txt.

[19] William Stallings, Lawrie Brown, *Computer Security: Principles and Practice*, third edition, 2014, ISBN 978-1292066172.

[20] P. Garcia-Teodoro e altri, *Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges*, Computers & Security, Vol. 28, 2009.

[21] Snort, sito web, `https://www.snort.org/`

[22] J. Agosta e altri, *Towards Autonomic Enterprise Security: Self-Defending Platforms, Distributed Detection, and Adaptive Feedback*, Intel Technology Journal, 9 Novembre 2006.

[23] A. Lazarevic, V. Kumar e J. Srivastava, *Intrusion Detection: A Survey*, tratto da *Managing Cyber Threats: Issues, Approaches and Challenges*, Springer, 2005.

[24] Roy Thomas Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, 2000, sito web, `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

[25] Thomas Erl, Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian, *SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST*, Upper Saddle River, New Jersey, Prentice Hall, 2012, ISBN 978-0-13-701251-0.

[26] Leonard Richardson, Mike Amundsen, *RESTful Web APIs*, O'Reilly Media, 2013, ISBN 978-1-449-35806-8.

[27] IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, Agosto 1998, sito web, `https://www.ietf.org/rfc/rfc2396.txt`.

[28] IETF RFC 1945, *Hypertext Transfer Protocol – HTTP/1.0*, Maggio 1996, sito web, `https://www.ietf.org/rfc/rfc1945.txt`.

[29] Roy Thomas Fielding, *REST APIs must be hypertext driven*, 20 Ottobre 2008, sito web, http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven.

[30] IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*, Agosto 2008, sito web, https://www.ietf.org/rfc/rfc5246.txt.

[31] Bruce Schneier, *The Process of Security*, Aprile 2000, sito web, https://www.schneier.com/essays/archives/2000/04/the_process_of_secur.html.

[32] P. Cichonsky e altri, *Computer Security Incident Handling Guide*, NIST Special Publication 800-61, Agosto 2012, sito web, https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final.

[33] Carnegie-Mellon Software Engineering Institute, *Handbook for Computer Security Incident Response Teams (CSIRTs)*, CMU/SEI-2003-HB-002, Aprile 2003.

[34] Cisco Adaptive Security Appliance (ASA), sito web, https://www.cisco.com/c/it_it/products/security/adaptive-security-appliance-asa-software/index.html.

[35] Cisco Firepower Next-Generation Firewall (NGFW) Data Sheet, sito web, https://www.cisco.com/c/en/us/products/collateral/security/firepower-ngfw/data_sheet-c78-736661.html.

[36] Fortinet FortiGate: Next Generation Firewall (NGFW), sito web, https://www.fortinet.com/products/next-generation-firewall.html.

[37] Fortinet FortiManager, sito web, https://www.fortinet.com/products/management/fortimanager.html.

[38] Check Point Next Generation Firewall, sito web, https://www.checkpoint.com/products/next-generation-firewall/.

[39] Cisco Identity Services Engine, sito web, https://www.cisco.com/c/en/us/products/security/identity-services-engine/index.html.

[40] *fmcapi*, progetto su Github, https://github.com/daxm/fmcapi.

[41] *FortigateApi*, progetto su Github, https://github.com/DavidChayla/FortigateApi.

[42] *fortipy*, progetto su Github, https://github.com/pschmitt/fortipy.

[43] *fortiosapi*, progetto su Github, https://github.com/fortinet-solutions-cse/fortiosapi.

[44] *cp_mgmt_api_python_sdk*, progetto su Github, https://github.com/CheckPointSW/cp_mgmt_api_python_sdk.

[45] *ise*, progetto su Github, https://github.com/bobthebutcher/ise.

[46] Cisco Firepower Management Center Configuration Guide, Version 6.1, Chapter: Reusable Objects, *Security Intelligence Lists and Feeds*, sito web, https://www.cisco.com/c/en/us/td/docs/security/firepower/610/configuration/guide/fpmc-config-guide-v61/reusable_objects.html#ID-2243-00000135.

[47] Fortinet Technical Documentation, *Blocking malicious domains using threat feeds*, sito web, https://cookbook.fortinet.com/blocking-malicious-domains-using-threat-feed-connectors-60/.

[48] Check Point Support Center, *How to block traffic coming from known malicious IP addresses*, sito web, https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk103154.

[49] Cisco Identity Services Engine Administrator Guide, Release 2.4, *Chapter: Setup Adaptive Network Control*, sito web, https://www.cisco.

com/c/en/us/td/docs/security/ise/2-4/admin_guide/b_ise_admin_guide_24/b_ise_admin_guide_24_new_chapter_01101.html.

[50] IETF RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*, Giugno 2000, sito web, https://www.ietf.org/rfc/rfc2865.txt.

[51] Authentication, Authorization, and Accounting Configuration Guide, Cisco IOS Release 15SY, *Chapter: RADIUS Change of Authorization*, sito web, https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_usr_aaa/configuration/15-sy/sec-usr-aaa-15-sy-book/sec-rad-coa.html.

[52] IETF RFC 2284, *PPP Extensible Authentication Protocol (EAP)*, Marzo 1998, sito web, https://www.ietf.org/rfc/rfc2284.txt.

[53] Cisco Systems, Inc., *MAC Authentication Bypass Deployment Guide*, Maggio 2011, sito web, https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/identity-based-networking-services/config_guide_c17-663759.html.

[54] Python, sito web, https://www.python.org/.

[55] TIOBE Index for February 2019, sito web, https://www.tiobe.com/tiobe-index/.

[56] Sphinx, sito web, http://sphinx-doc.org/.

[57] Cisco Systems, Inc., *About the ASA REST API v1.3.2*, Luglio 2018, sito web, https://www.cisco.com/c/dam/en/us/td/docs/security/asa/api/asapedia_rest_api_132.pdf.

[58] *Cisco ASA Compatibility* Febbraio 2019, sito web, https://www.cisco.com/c/en/us/td/docs/security/asa/compatibility/asamatrx.html#id_65991.

[59] *Cisco ASA REST API Quick Start Guide*, sezione *Install and Configure the ASA REST API Agent and Client*, Giugno 2018, sito web, `https://www.cisco.com/c/en/us/td/docs/security/asa/api/qsg-asa-api.html#pgfId-61941`.