Politecnico di Torino

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Configurazione automatica di web-application firewall

Relatori
prof. Riccardo Sisto
dott. Daniele Bringhenti
dott. Fulvio Valenza

Candidato
Michele Argentino

Anno Accademico 2021-2022

Sommario

Il paradigma Network Functions Virtualization (NFV) è una tecnologia di rete che, attraverso un disaccoppiamento tra le funzioni di rete e le apparecchiature hardware, consente di eleminare gli hardware specializzati, per eseguire una singola funzione di rete, e permette di installare processi software che agiscono da vere e proprie funzioni di rete su server generici. Uno dei vantaggi che questo paradigma da è che permette di avere una notrvole flessibilità nella creazione di un Service Graph.

Il problema che sorge, però, è che la creazione di un Service Graph di solito eseguita da un amministratore di rete. Questo amministratore, incaricato dell'allocazione e della configurazione delle Network Security Functions (NSF), come web application firewall, necessarie per proteggere la rete dagli attacchi alla sicurezza informatica, svolge queste operazioni manualmente, quindi sono soggette a errori umani.

Questa tesi ha contribuito allo sviluppo di VEREFOO (VErified REFinement and Optimized Orchestration), un framework che mira a fornire un approccio automatico nella sicurezza per evitare il problema presentato.

Lo scopo principale è quello di eseguire, su un Service Graph fornito in input, un'allocazione e configurazione ottimizzata automatica delle NSF necessarie per soddisfare una serie di requisiti di sicurezza della rete, che possono essere espressi dall'amministratore di rete sfruttando un linguaggio di alto livello.

L'approccio di VEREFOO consiste nella formulazione di un problema MaxSMT. Lo scopo è quello di soddisfare un insieme di clausole hard che richiedono di essere se,pre soddisfatte e, allo stesso tempo, di raggiungere la somma massima dei pesi che sono attribuiti alle clausole soft.

VEREFOO ha come obiettivi:

1. l'assegnazione del numero minimo di istanze NSF e ridurre il consumo di risorse grazie all'allocazione del virtuale corrispondente funzioni;

2. la riduzione delle regole che ne descrivono la configurazione per migliorare l'efficienza delle operazioni di filtraggio.

Il contributo fornito da questo lavoro di tesi è stato quello di estendere le funzionalità del framework per supportare anche la configurazione del Web Application Firewall.

Il Web Application Firewall (WAF) è molto importante al giorno d'oggi in quanto una grande quantità di traffico nella rete è generato dalle Web Application. Lo scopo ultimo della tesi è, quindi, stato quello di poter automatizzare la configurazione e allocazione di un web application firewall all'interno della rete in modo da proteggere le applicazioni dagli attacchi esterni, tra i quali i più noti sono attacchi XSS e SQL Injection.

L'implementazione è stata guidata dallo studio preliminare dei principali Web Application Firewall presenti in commercio. Sfruttando le conoscenza acquisite attraverso questa analisi sono stati modellizzati i parametri e le azioni principali per un WAF. Questa fase è stata importante perché necessaria per poter garantire ad un amministratore di rete le funzionalità necessarie per proteggere la propria Web Application dagli attacchi.

L'implementazione è stata infine testata in scenari di rete comuni e ha mostrato una certa scalabilità rispetto alla dimensione del Service Graph e al numero di requisiti di sicurezza dell'input; di conseguenza, questa tesi dimostra che l'approccio proposto è fattibile, ma che deve essere ottimizzato per poter garantire la fruibilità in una rete di dimensioni importanti.

Uno spunto per poter ottimizzare la configurazione di un WAF è dato all'interno della tesi: in futuro potrà e dovrà essere necessaria l'estensione delle funzionalità per supportate il carattere jolly "*" per le variabili di tipo stringa nel modello z3, funzionalità importante per poter ridurre il numero di configurazioni presenti all'interno di un WAF.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Obiettivo della tesi

Questoprimo capitolodescrivealcune tecnologie di retechesonoallabasedellavoro svolto in questa tesi. Negli ultimi anni sono emerse nuove tecnologie di rete tra cui la virtualizzazione delle funzioni di rete (NFV) e le Software-Defined Networks (SDN):

1. *Virtualizzazione delle funzioni di rete (NFV)*. Il concetto di NFV consiste nel sostituire l'hardware dell'applicazione di rete con macchine virtuali e processi sotto il controllo di un hypervisor; Ciò consente un'implementazione flessibile delle funzioni di rete.

2. *Software Defined Networking (SDN)*:si propone di centralizzare l'intelligenza di rete nei singoli componenti separando il processo di inoltro dei pacchetti (piano dati) dal processo di instradamento (piano di controllo).

Prima di questi paradigmi, come mostrato nella Figura 1.1, un Service Graph(SG), una generalizzazione di una catena di funzioni in cui possono esistere percorsi diversi tra qualsiasi coppia di endpoint, era un dispositivo hardware che implementava un particolare servizio. L'analisi di alcune metriche prestazionali mostra l'impatto di

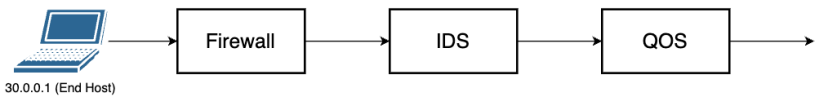

Figura 1.1. Service Graph senza SDN

questo tipo di catena:

- Tempo necessario per fornire nuovi servizi:le apparecchiature devono essere fisicamente collegate;

- Flessibilità: la connessione di un nuovo dispositivo di rete richiede un'interruzione temporanea nella catena, che porta all'interruzione della rete;
- Affidabilità: il guasto del dispositivo di rete interrompe la catena e interrompe la rete;
- Ottimizzazione: ogni dispositivo di rete può utilizzare una certa quantità di risorse, quinidi, ad esempio, nei picchi di lavoro non può sfruttare le risorse eventualmente lasciate libere in quel momento da un altro dispositivo.

Oggi, grazie alle Software Defined-Networks e alla virtualizzazione delle funzioni di rete, si tende sempre di più a virtualizzare le funzioni di rete poichè, in questo modo, si può sfruttare meglio il dispositivo fisico, l'hardware, in modo tale che più funzioni di rete possono essere installate su una stessa macchina, il server. L'utilizzo di una singola macchina fisica per creare più funzioni di rete virtualizzate consente di risparmiare. Pertanto, i costi vengono ridotti eliminando la necessità di acquistare un dispositivo fisico dedicato per ogni funzione di rete richiesta. Le funzioni di rete che possono essere virtualizzate includono, ad esempio, loadbalancer, NAT, VPN, packetfilter, firewall, web application firewall (WAF, l'argomento su cui si basa questa tesi).

Lo scopo della tesi è contribuire all'implementazione di VEREFOO (VERified Refinement and Optimized Orchestration), un framework il cui obiettivo principale è la distribuzione automatica e la configurazione ottimale delle funzioni di rete. In particolare, ci siamo concentrati sull'estensione delle funzionalità del modulo ADP della suddetta piattaforma per includere la configurazione e l'assegnazione automatica di web application firewall sulla rete. I web application firewall distribuiti sono configurati in base a una serie di requisiti espressi dagli amministratori di rete in un linguaggio di alto livello.

VEREFOO nasce per evitare gli errori di configurazione dovuti alla componente umana nel Service Graph. La configurazione è un punto molto delicato, in quanto un errore può rovinare l'intero grafo in termini di funzionalità e ottimizzazione. Gli errori più comuni che un amministratore può introdurre in una rete includono errori di ridondanza che portanoa problemi di ottimizzazione e errori più gravi che possono interrompere la corretta comunicazione tra i nodi della rete.

Si vedrà, infatti, nel corso della tesi come una configurazione automatica è utile per ottimizzare e migliorare le configurazioni dei servizi di rete.

1.2 Descrizione e Organizzazione Tesi

In questa sezione vengono illustrati brevemente, capitolo per capitolo, i concetti chiave affrontati in questa tesi.

1. Capitolo 2: introduzione a VEREFOO. Il capitolo è diviso in tre sezioni. Nella prima sezione si fa un'introduzione delle tecnologie su cui si basa VEREFOO, SDN e VNF. Nella seconda sezione si parla invece di VEREFOO,

in particolar modo si fa riferimento all'approccio scelto per l'implementazione del framework. Infine, nella terza sezione, si fa una panoramica del framework e del modulo ADP su cui si è concentrato il lavoro della tesi.

2. Capitolo 3: dedicato all'analisi e alla descrizione degli attuali WAF in circolazione. Il capitolo è diviso in tre sezioni: nella prima si va a definire cosa è un Web Application Firewall; nella seconda si va a spiegare come funziona esattamente un Web Application Firewall; nella terza invece si entra più nel pratico e si va a mostrare l'analisi fatta sugli attuali Web Application Firewall in commercio. In particolar modo nell'ultima sezione si va definire quali sono le configurazioni principali dalle quali si è partiti per l'implementazione della nuova funzionalità all'interno di VEREFOO.

3. Capitolo 4: dedicato alla descrizione dell'approccio scelto per sviluppare la soluzione adottata. In questo capitolo viene spiegato l'obiettivo della tesi e verranno forniti più dettagli di come e cosa è stato modificato all'interno di VEREFOO. Infine, verrà descritto come l'approccio si basi su una formulazione del problema MaxSMT.

4. Capitolo 5: definizione della soluzione. Il capitolo è diviso in quattro sezioni diverse. Nella prima sezione si descrivono e si caratterizzano i requisiti che un amministratore di rete può formulare relativamente ai Web Application Firewall. Nella seconda sezione sono state modellizzate direttamente le caratteristiche principali configurabili del singono Web Application Firewall. Nelle utlime due sezioni viene mostrato, con l'aiuto di alcuni esempi, quali sono i requisti rappresentabili attraverso la modellizzazione mostrata nelle prime due sezioni e quali sono le limitazioni introdotte con il modello scelto.

5. Capitolo 6: dedicato alla descrizione e alla spiegazione della soluzione adottata. Questo capitolo viene diviso in tre sezioni. Nela prima sezione si spiega il modello adottato per la soluzione all'internon del framework. Nella seconda si dimostra la configurazione attraverso l'aiuto di un esempio. Infine, nell'ultima sezione si descrive il problema MaxSMT e quali sono i suoi obiettivi.

6. Capitolo 7: dedicato alla descrizione dell'implementazione e della validazione del modello adottato. Diviso in due sezioni. La prima descrive l'implementazione effettiva delle configurazioni previste dalla soluzione; la seconda invece è dedicata alla validazione del modello adottatto attraverso degli "use case" che mostrano come si comporta la soluzione all'interno del framework di VEREFOO.

7. Capitolo 8: dedicato alle conclusioni e ad una riflessione sui lavori futuri che potrebbero essere fatti a partire dai risultati ottenuti da questa tesi.

Capitolo 2

Verefoo

Questo capitolo analizza VEREFOO e la sua implementazione, che dovrebbe eliminare o almeno limitare il più possibile gli errori di configurazione associati a fattori umani e ottenere l'automazione e l'ottimizzazione della configurazione delle funzioni di rete.

2.1 Introduzione

Come descrittoin [1],la configurazione e la distribuzione delle funzionalità di sicurezza su una rete viene solitamente eseguita manualmente. Ciò porta spesso a violazioni della sicurezza, con conseguente aumento dei tempi di riconfigurazione della rete. Questo problema è aggravato se si utilizza una rete virtualizzata. La sua complessità e dinamicità rendono molto difficile una corretta configurazione manuale. La flessibilità e il dinamismo forniti dalla virtualizzazione della rete possono portare ad altri problemi che possono sorgere durante la configurazione. In questi contesti, i problemi che possono sorgere sono legati al mantenimento della sicurezza. Ci sono periodi di transizione durantela modifica delle configurazioni che possono indicare uno stato di transizione non sicuro. [10] cerca di trovare una soluzione a questo problema cercando una metodologia per evitare possibili errori umani, basata su una combinazione di tre caratteristiche principali: automazione, ottimizzazione e verifica formale. Pertanto, l'errata configurazione delle funzionalità di sicurezza della rete come firewall e Web application firewall è diventata uno dei problemi piùi mportanti. Questo è un problema insito nell'approccio manuale utilizzato dagli amministratori di rete. Le regole di filtraggio e/o protezione sono in genere assegnate alle funzioni di sicurezza della rete utilizzando criteri di buon senso non ottimali. È a causa di questo rischio che è necessaria una policy automatizzata. I criteri aiutano gli amministratori a creare e configurare i servizi di sicurezza. Ciò avviene attraverso un processo automatizzato che gestisce la creazione di policy per ogni funzione di rete, seguendo una serie di requisiti che rappresentano obiettivi da raggiungere. I vantaggi dell'utilizzo di questo approccio di "automazione della sicurezza" sono la prevenzione dell'errore umano, l'analisi automatica dei conflitti di policy e la verifica formale che siano effettivamente corretti. Uno dei principali vantaggi è la capacità di ricercare l'ottimalità. Per gli algoritmi automatizzati è più facile ottenere risultati ottimali rispetto agli algoritmi manuali.

La flessibilità è un requisito fondamentale per supportare l'approccio richiesto dai processi automatizzati. Da questo punto di vista, le nuove tecnologie di networking come NFV (Network Functions Virtualization) e SDN (Software Defined Networking) possono dare un contributo importante.

2.1.1 Software Defined Network

La teoria della SDN [2] introduce un disaccoppiamento tra le funzioni di elaborazione, che vengono virtualizzate come macchine virtuali o Docker, e i server fisici general-purpose su cui sono installati. I pilastri della teoria SDN originale sono tre:

- Una separazionetra il piano dati, che prende le decisioni sull'inoltro e gestisce le modifiche di basso livello dei pacchetti, e il piano di controllo, che coordina le azioni di inoltro costruendo strutture dati condivise, come il database di filtraggio per un bridge o una tabella di instradamento per un router;
- Tutte le funzioni del piano di controllo sono centralizzate in un singolo modulo, comunemente denominato controller SDN. Questo modulo racchiude tutta l'intelligenza di questa tecnologia. La centralizzazione è tecnicamente possibile a livello logico, ma la maggior parte delle implementazioni pratiche si basa sulla centralizzazione fisica;
- Definizione delle interfacce southbound e northbound per la comunicazione tra il controller SDN e, rispettivamente, l'infrastruttura di inoltro e le applicazioni di livello utente o i controller di livello superiore;

L'infrastruttura di rete è caratterizzata da elementi di inoltro dei dati che non devono necessariamente essere router complessi o dispositivi personalizzati dal fornitore. Il loro unico compito è inoltrare i pacchetti di input alla porta di output corretta lungo il percorso per raggiungere la loro destinazione finale il più rapidamente possibile, quindi sono costruiti con materiale semplice. In effetti, il trasporto efficiente è un concetto chiave della tecnologia SDN.

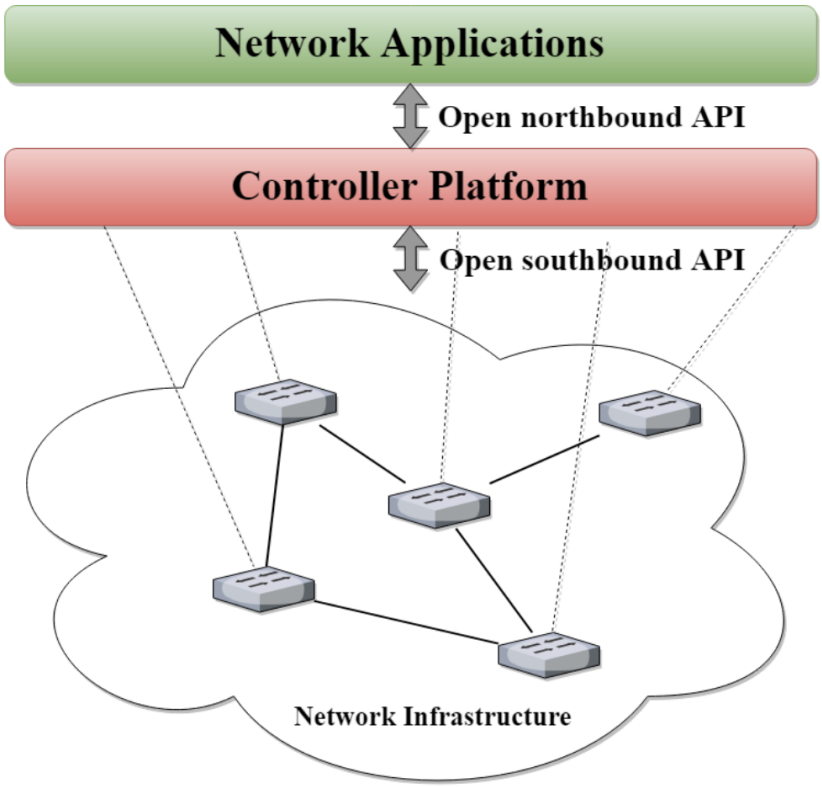

Figura 2.1. Architettura SDN

2.1.2 Network Functions Virtualization

L'idea principale di questa nuova tecnologia è che le funzioni di rete non richiedono hardware speciale, sono implementate da processi software e possonoessere eseguite su server ordinari. Mentre SDN si concentra sulla creazione di percorsi per le comunicazioni tramite software, NFV si concentra sulla virtualizzazione dell'intera infrastruttura.

Ci sono diversi modi per implementare le funzioni di rete in modo vistualizzato: un primo modo per farlo è basato sulle macchine virtuali. Queste ultime sono in grado di fornire un forte isolamento. Ogni volta che viene creata una nuova macchina virtuale, vengono definiti un nuovo profilo hardware e un diverso sistema operativo che può essere installato su di esso, indipendentemente dall'hypervisor; è, di conseguenza, la soluzione migliore per scenari dove la protezione dagli accessi ai propri servizi è una questione fondamentale.

Tuttavia, questo vantaggio va a discapito di grandi requisiti di memoria e spazio su disco.

L'approccio moderno alla virtualizzazione si basa su Docker. Il successo di Docker è dovuto alla sua portabilità. Una volta creato Docker, può essere trasferito su un'altra macchina senza problemi. Il potenziale problema dei file system indipendenti che occupano spazio su dischi di grandi dimensioni può essere risolto anche con file system a più livelli. Il file system a più livelli consente a più docker di condividere parti del file system tra loro e con l'hypervisor. Il limite principale è la mancanza di isolamento tra i contenitori, che può portare a potenziali problemi di sicurezza. Per questo motivo, utilizzare Docker in ambienti in cui ogni contenitore è di proprietà di un utente diverso potrebbe non essere la soluzione migliore.

Indipendentemente dalla tecnologia utilizzata, il vantaggio principale del paradigma NFV è la capacità di scalare automaticamente le funzioni di rete virtualizzate (VNF) quando sono necessarie più risorse. In questo contesto si possono utilizzare due approcci.

- Assegnazione di più risorse alla stessa VNF. Con questo approccio, i limiti di allocazione delle risorse sono determinati dal numero massimo di risorse disponibili per il server su cui è installato VNF. Inoltre, ottenere più risorse potrebbe non essere sempre vantaggioso.

- Installazione di più istanze della stessa VNF. Il secondo approccio è più flessibile e offre la possibilità di parallelizzare le operazioni. Tuttavia, questa strada potrebbe richiedere un Load Balancer che tenga conto del carico effettivo su ogni istanzae per instradare meglio il traffico.

2.1.3 Applicazione della teconlogia NFV con SDN

Come discusso in precedenza, la tecnologia NFV consente di abbandonare l'hardware dedicato alle singole funzioni e sostituirlo con processi software che possono essere installati su server regolari (mostrato anche nella Figura2.2). Laddove NFV

fornisce questo vantaggio, il paradigma SDN può fare affidamento sulle regole applicate dal controller SDN per determinare il flusso dei pacchetti. Pertanto, NFV e SDN sono nuovi paradigmi che cambiano le regole del networking e prestano attenzione al dinamismo e alla programmabilità [3]. Queste due tecnologie sono utilizzate come strumenti complementari in questo approccio. Oltre ai vantaggi già forniti da SDN, la NFV applicata alla catena dei servizi di rete aggiunge altri vantaggi:

- Maggiore flessibilità per implementare nuove funzionalità tramite software invece di acquistare, connettere e configurare nuovi dispositivi hardware;
- Un soft switch configurato come switch SDN consente la comunicazione dinamica con funzioni sullo stesso server;
- Tutte le funzioni virtualizzate su un singolo server condividono risorse di elaborazione che possono essere allocate dinamicamente secondo necessità.
- Si possono facilmente creare regole di inoltro nello switch in modo proattivo, non reattivo, con una comunicazione minima tra lo switch e il controller SDN.

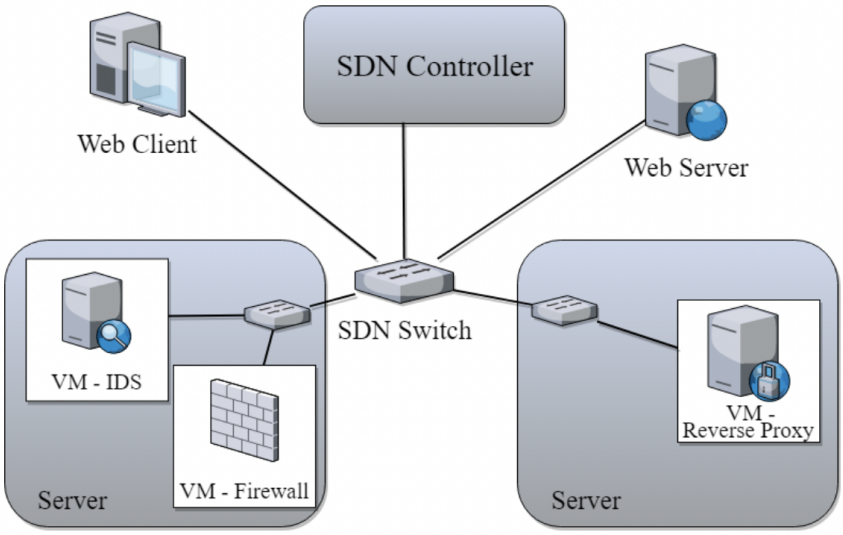

Figura 2.2. Architettura NFV applicata a SDN

2.1.4 Verefoo

La virtualizzazione ha sicuramente aumentato la flessibilità nella protezione della rete, rendendo più semplice lo sviluppo di nuove implementazioni delle funzioni di sicurezza. Tuttavia, comediscusso in [4], lo svantaggio di questa opportunità è che gli amministratori di rete responsabili della configurazione e della distribuzione delle funzionalità di sicurezza sulle proprie reti devono scegliere l'opzione migliore. Vari lavori sono stati fatti in letteratura per automatizzare completamente la configurazione delle funzioni di rete [5][6][9]. Tutta la ricerca fatta in letteratura è

sempre finalizzata a limitare l'interazione umana con la configurazione e ad ottimizzare le soluzioni per migliorare la sicurezza. Tenendo presenti queste considerazioni, l'approccio utilizzato in questa tesi prevede un framework completamente automatizzato chiamato VEREFOO [7], che sta per Verified Refinement and Optimized Orchestration. Il suo scopo è quello di affinare i requisiti di sicurezza di alto livello, espressi con un linguaggio semplice per l'uomo, attraverso uno schema ottimale di allocazione e configurazione delle funzinoi di sicurezza su un Service Graph (SG). Questa fase viene eseguita in modo da garantire la correttezza della costruzione; i risultati calcolati sono ottimali rispetto ad un insieme di funzioni di costo, come la minimizzazione del numero di funzioni installate o il numero di regole configurate. puòimplementareeconfigurare Questo framework può implementare e configurare le funzioni virtuali e di configurarle attraverso un'interazione diretta con alcuni orchestratori, senza alcuna operazione manuale.

2.2 L'approccio

VEREFOO gestisce la creazione, la configurazione e l'orchestrazione di complessi servizi di sicurezza di rete secondo l'approccio modulare che si riflette nel framework stesso. Innanzitutto, VEREFOO esegue automaticamente la distribuzione delle funzioni di rete (NSF) e ottimizza la configurazione richiesta per soddisfare i requisiti di sicurezza (NSR) impostati per il Service Graph in input. Il Service Graph di input è costituito da funzioni di rete senza alcuna capacità di sicurezza; pertanto, deve essere innanzitutto trasformato automaticamente in una topologia logica, chiamata Allocation Graph, dove tra ogni coppia di funzioni virtuali viene creato un Allocation Place. Ogni Allocation Place è una posizione segnaposto in cui potrebbe essere allocato un NSF, se necessario, per soddisfare i vincoli di sicurezza in ingresso. Un esempio è mostrato nella Figura 2.3, dove vale la pena sottolineare che ogni elemento, dal NAT al bilanciatore di carico, è in realtà una VNF piuttosto che un dispositivo fisico.

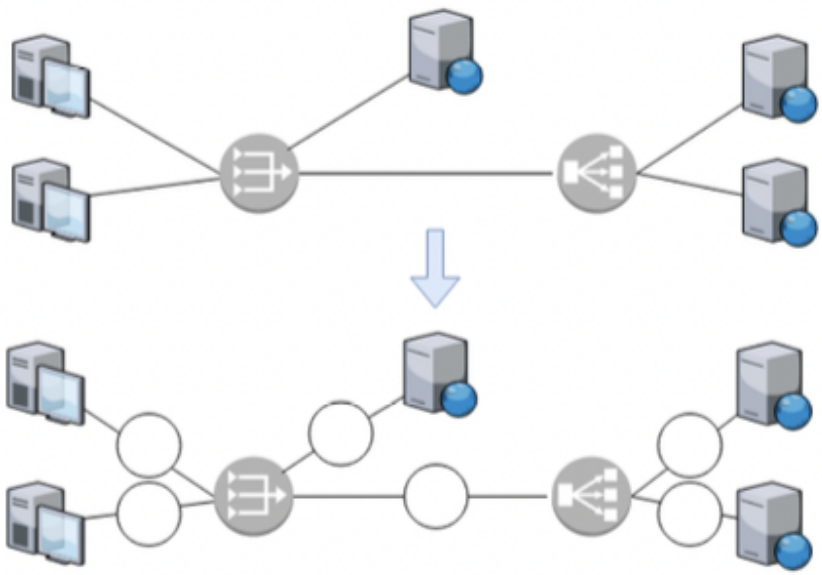

Figura 2.3. Esempio di Allocation Graph

2.2.1 Maximum Satisfiability Modulo Theories

L'approccio di VEREFOO alla configurazione e all'assegnazione automatica delle funzioni di rete si basa sulla risoluzione del problema MaxSMT. Il Maximum Satisfiability Modulo Theories (MaxSMT) ha il compito di selezionare automaticamente le funzioni di rete necessarie e la loro posizione; in questo modo, viene fornita anche una corretta garanzia formale. La ricerca di soluzioni ottimali è senza dubbio l'elemento centrale che è stato considerato per raggiungere questi obiettivi. Uno biettivo secondario dopol a creazione di un servizio di sicurezza virtuale è il posizionamento ottimale delle singole funzioni sui server fisici che compongono la rete.

Questo passaggio considera altre funzioni di costo come il ritardo nter-VNF e la riduzione al minimo del consumo di risorse. Inoltre, poiché ogni NSF è caratterizzata da policy (espresse con un linguaggio di medio livello che fornisce un'astrazione dalle diverse implementazioni), è necessaria una traduzione per ottenere la configurazione specifica del fornitore di ogni funzione virtuale. Infine, il servizio viene configurato attraverso un'integrazione con gli orchestratori cloud con lo scopo di fornire le proprietà di sicurezza per la comunicazione tra i punti finali delle reti. È importante sottolineare però che questo risultato è stato ottenuto grazie agli algoritmi completamente automatici che vengono sfruttati nell'approccio complessivo, partendo da un Service Graph e da un insieme di requisiti di sicurezza come input.

2.3 Panoramica del Framework

Secondo la visione di VEREFOO, gli utenti degli orchestratori NFV, ovvero i progettisti di servizi, possono inserire come input:

- Un set di requisiti di sicurezza (NSR) che esprime i vincoli da soddisfare utilizzando un linguaggio di livello medio/alto, a seconda del livello di esperienza dell'utente, attraverso una policy GUI che facilita la creazione di requisiti più semplici.
- un Service Graph (SG) o, in alternativa, un Allocation Graph (AG) attraverso una Service GUI, che fornisce l'accesso a un Network Functions Catalog (Catalogo NF) dal quale l'utente può decidere quali funzioni di rete, semplici o anche NSF, allocare immediatamente sul grafo.

La Figura 2.4 presenta una panoramica completa del framework.

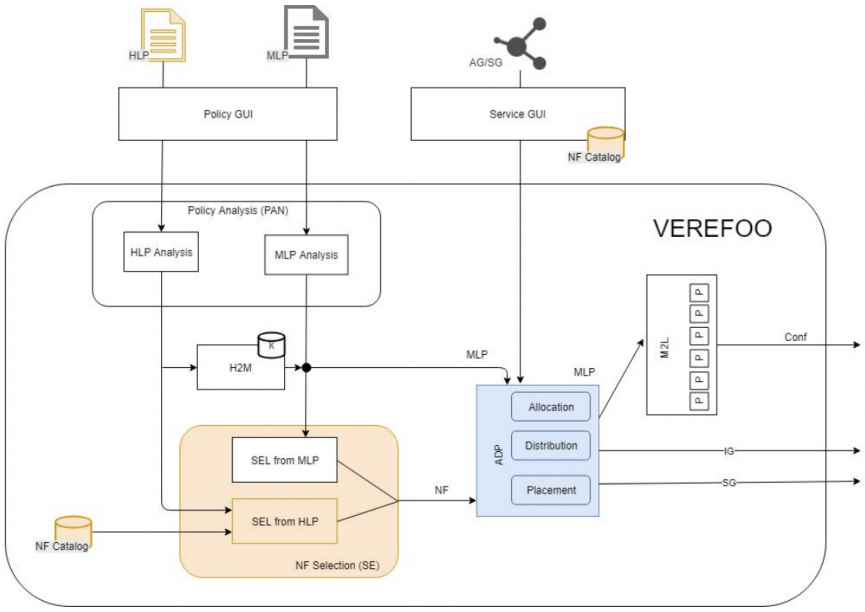

Figura 2.4. Architettura di VEREFOO

Ricevuti gli input dal progettista, una prima fase preliminare all'interno del frameworl è rappresentata dalla PAN, Policy ANalysis. L'obiettivo principale di questo passaggio è iniziare con l'input ricevuto e utilizzare diversi metodi per eseguire l'analisi dei conflitti per vedere se ci sono requisiti e vincoli contrastanti che devono essere soddisfatti nella rete. A questo punto è possibile ottenere in output un failure report se i requisiti espressi in input contengono errori o conflitti che non possono essere risolti automaticamente dalla piattaforma e devono essere rivisti dall'utente. Dopo questa fase, se il dato requisito di sicurezza è espresso in un linguaggio di alto livello, il modulo High-to-Medium (H2M) [8] esegue il raffinamento per ottenere il corrispondente set di requisiti di medio livello, che contengono tutte le informazioni utili per la creazione delle policy delle funzioni di rete che verranno

allocate automaticamente sul grafo e la configurazione di basso livello delle VNF posizionate sulla rete. Un ruolo chiave è poi ricoperto dal modulo NF Selection (SE), il quale, sulla base degli NSR di alto e medio livello inseriti, decide quali NSF sono necessari per soddisfarli. Gli NSF vengono scelti da un catalogo precostituito, ovvero lo stesso elenco a cui il progettista del servizio ha accesso attraverso la Service GUI. Questo passaggio richiede un processo di ottimizzazione attraverso il quale viene selezionato l'insieme ottimale di NSF, nonostante l'operazione non sfrutti alcuna conoscenza della topologia dell'Allocation Graph. Questo risultato si ottiene nel seguente modo:

- Innanzitutto, SE ottiene l'elenco delle istanze della funzione richiesta da un modulo esterno al framework.
- Quindi, tra le funzioni presenti nel catalog NF, cerca le funzioni che supportano le funzioni richieste, seleziona le funzioni migliori, tenendo conto delle risorse fisiche disponibili, e le assegna ai server fisici.

La scelta delle caratteristiche è condizionata. Innanzitutto, la funzione deve essere in grado di supportare le funzioni, ma devono essere disponibili anche determinate risorse fisiche affinché la funzione sia ospitata sul server. Inoltre, la scelta è soggetta a ottimizzazione. È possibile ridurre il costo delle funzioni e ridurre la quantità di RAM. Il modulo Allocation, Distribution and Placement (ADP) è uno degli elementi principali dell'architettura, il cui scopo è quello di calcolare un Service Graph con i NSF aggiunti e di decidere il posizionamento dei VNF sulla rete substrato ricevendo come input gli NSR di medio livello, l'elenco degli NSF selezionati e il Service Graph originale o direttamente l'Allocation Graph. Il modulo ADP utilizza z3Opt come solver MaxSMT, e Verigraph come strumento per la verifica delle NSR per fornire tre caratteristiche principali:

- Sulla base dell'elenco NSF selezionato dal modulo di selezione NF, il modulo ADP regola le distribuzioni nell'Allocation Graph ricevuto in ingresso o ottenuto dal Service Graph elaborato per soddisfare i requisiti dati iningresso;
- Contemporaneamente alla fase di allocazione, il secondo compito è quello di propagare la policy nella NSF allocata;
- Nella seconda fase, dopo la creazione del Service Graph NSF-enhanced, le VNF che implementano le funzioni di rete del SG originale e le NSF annesse vengono collocate nell'infrastruttura fisica secondo il principio della minimizzazione del consumo di risorse.

Un ulteriore output dell'elemento ADP è l'elenco delle policy di medio livello con cui deve essere configurata ogni istanza di funzione di rete; poi, la corrispondente configurazione di basso livello, che dipende dall'implementazione specifica della funzione implementata, viene generata dal modulo Medium to Low (M2L), che esegue una traduzione delle espressioni indipendenti dal fornitore nelle regole da impostare sulla funzione appropriata.

Capitolo 3

Web Application Firewall

Uno dei punti chiave di questo capito lo è un'analisi preliminare della funzione di rete in esame: il web application firewall. In questo capitolo esamineremo e analizzeremo cos'è un web application firewall e come funziona esattamente.
La sezione seguente discute anche i vari tipi di WAF oggi sul mercato, concentrandosi sui principali punti in comune che hanno.

3.1 Cos'è un Web Application Firewall

Che cos'è un Web Application Firewall? Un web application firewall (WAF) è una forma speciale di application firewall che filtra, monitora e blocca il traffico HTTP proveniente dai servizi web. È possibile prevenire gli attacchi che sfruttano le vulnerabilità note delle applicazioni Web come XSS, File inclusion e gli errori di configurazione del sistema. Pertanto,lo scopo dei WAF è proteggere le applicazioni Web e aggiungere un ulteriore livello di protezione a quelli forniti da un firewall o da un sistemadi prevenzione delle intrusioni(IPS).

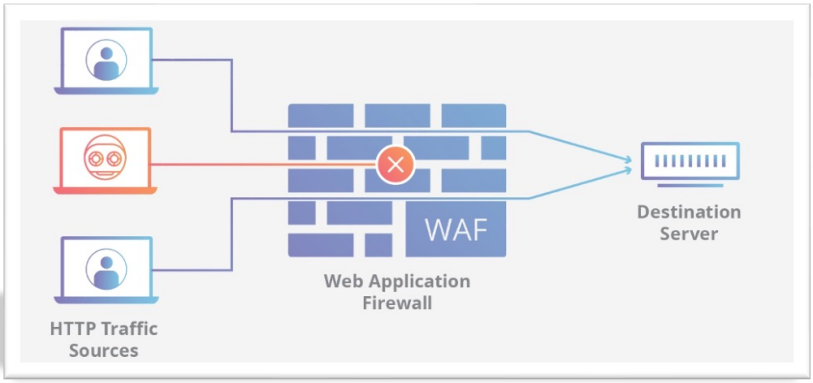

Figura 3.1. Stuttura di un WAF

Nel corso della storia i WAF si sono evoluti dando vita ad una serie di generazioni:

- WAF 1.0: la prima generazione di sicurezza delle applicazioni Web ha introdotto due innovazioni chiave in IDS/IPS, ovvero l'utilizzo degli attributi HTTP. Si tratta di soluzioni non particolarmente intelligenti, basate su un metodo "signature-based", che hanno come obiettivo la protezione degli attacchi ai server;

- WAF 2.0: Nel corso degli anni,il web si è evoluto ed è diventato sempre più complesso. Per garantire la sicurezza, anche i dispositivi di protezione dovevano andare di pari passo con l'innovazione. I nuovi WAF di questa generazione sono in grado di intercettare e analizzare i pacchetti che passano attraverso la rete, e attraverso questa operazione sono in grado di identificare potenziali pericoli.

- WAF 3.0: Questo è il WAF di ultima generazione ed è un sistema di sicurezza completo. Infatti, può intercettare, analizzare e monitorare il traffico e, a differenza di un WAF di seconda generazione, può proteggere attivamente la rete, non solo rispondendo ad un attacco, ma cercando di intercettare la vulnerabilità.

3.2 Perchè usare un Web Application Firewall?

In genere si consiglia di disporre di un'istanza WAF in esecuzione se si verifica una situazione che riflette uno o più dei seguenti problemi di sicurezza relativi all'applicazione Web:

- nessuna possibilità di ispezionare il codice sorgente;
- vulnerabilità che non possono essere riparate o mitigate nel sistema;
- vulnerabilità che richiedono troppi sforzi (costi) per essere risolte;
- troppo vecchia per applicare una patch;
- troppo critica per applicare una patch;
- troppo complessa per essere protetta in modo efficiente;
- autenticazione/autorizzazione non disponibile sul sistema;

Quando parliamo di sicurezza delle applicazioni web, di solito parliamo anche di patch virtuali. Virtual Patch è un termine usato per descrivere una delle funzionalità chiave del WAF. Ciò consente di applicare virtualmente patch alle applicazioni Web vulnerabili e bloccare gli attacchi prima che raggiungano il gateway dell'Application Proxy.

3.3 Come funziona un Web Application Firewall

In questa sezione andiamo a descrivere come funziona nel dettaglio un waf e quali sono i suoi principi di funzionamento. Per i WAF esistono due modelli di lavoro principali:

- Modello positivo
- Modello negativo

Il modello negativo prevede che qualsiasi comunicazione sarà sempre consentita, a meno che non ci siano regole che bloccano esplicitamente una particolare comunicazione. Prendiamo come esempio la Comunicazione C. Questa è sempre consentita a meno che una regola non lo vieti esplicitamente. Questo modello fornisce un elenco di regole "negative" che vietano determinate comunicazioni. Questa lista, chiamata "Blacklist" viene costruita con due possibili approcci [12]: Signature Based e Rule Based.

- Signature Based: questo approccio richiede il confronto del traffico in entrata o in uscita con una stringa o un'espressione regolare. Se c'è una corrispondenza, c'è un'alta probabilità che la minaccia sia stata rilevata.

- Basato su regole: basato sull'uso di regole create utilizzando espressioni booleane. Utilizziamo queste espressioni per determinare se il comportamento di un utente rappresenta una potenziale minaccia. Ad esempio, se una transazione contiene un parametro URL "esempioPagina.php" con un indirizzo IP di origine sconosciuto, questa potrebbe essere una minaccia.

Il modello positivo, noto anche come approccio whitelist, è l'esatto opposto del modello negativo sopra descritto. In effetti, questo modello consente di far passare solo i messaggi ritenuti validi nell'applicazione e di rifiutare altri messaggi che non hanno superato la fase di convalida preliminare. Il modello positivo è considerato più sicuro perché lascia passare solo ciò che è determinato essere sicuro. D'altro canto, lo svantaggio è che l'utilizzo di questo approccio richiede una conoscenza approfondita dell'applicazione da proteggere e può essere difficile da mantenere se l'applicazione viene aggiornata di frequente.

L'approccio utilizzato dal modello positivo è chiamato "Anomaly Based". Questo approccio consiste nell'utilizzare le statistiche per profilare l'attività del sistema nel tempo. Questa profilazione aiuta a tenere traccia del comportamento considerato normale. Qualsiasi modifica a questo comportamento al di sopra di una certa soglia è considerata un potenziale attacco.

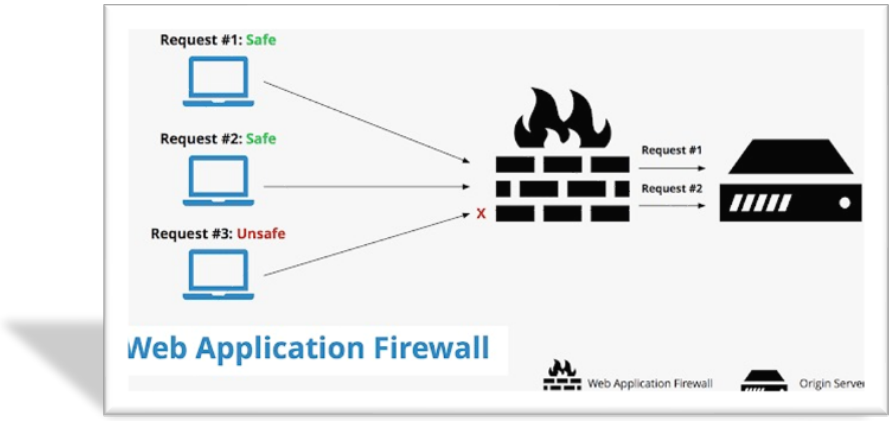

Figura 3.2. Principio di funzionamento WAF

3.4 OWASP Top Ten

Il progetto OWASP Top Ten [13] dell'OWASP mira a migliorare la comprensione di importanti aspetti della sicurezza delle applicazioni web. Tutti gli aspetti di sviluppo e analisi sono legati alla protezione delle applicazioni web, quindi è destinato a tutti coloro che lavorano in rete, sviluppatori e amministratori di rete. Questo progetto identifica i 10 principali fattori di rischio per la sicurezza delle applicazioni web. In cima ci sono Injection e Cross-Site Scripting.

3.4.1 Injection

L'injection[14] è considerato il principale tipo di attacco nelle applicazioni web.

I motivi degli attacchi di Injection variano e spesso includono:

- attacchi di tipo denial-of-service (DoS);
- perdita di dati;
- modifica dei dati;

Qualsiasi forma di dati, esterna o interna, dovrebbe essere considerata pericolosa e un potenziale vettore di attacco.

I risultati degli attacchi spesso includono:

- Alcune informazioni potrebbero essere comunicate ad utenti che non vi hanno accesso. Pertanto, la confidenzialità può essere violata.
- Si potrebbe accedere alle informazioni private di uno specifico utente senza averne le credenziali. In questo caso, la sicurezza dell'autenticazione è compromessa.
- Alcune informazioni nel database possono essere modificate da utenti che non dovrebbero. Pertanto, i criteri di autorizzazione possono essere aggirati.
- Oltre alla modifica potrebbero essere anche compromessi alcuni dati, si potrebbe quindi perdere la proprietà di integrità dei dati subendo modifiche, cancellazioni o inserimento di dati errati.

```
<?php
$id = $_POST['id'];
mysql_query("SELECT * FROM Utenti WHERE id = $id");
...
?>
```

Figura 3.3. SQL Injection

La figura 3.3 mostra un esempio di come la riservatezza può essere violata utilizzando script di codice. Vediamo nel diagramma come la variabile viene inviata direttamente dall'utente in una richiesta HTTP. Se si considera nella richiesta id=abcd, la query SQL risultante è:

```
"SELECT * FROM Utenti WHERE id=abcd"
```

Una query scritta in questo modo restituirà in output tutte le informazioni disponibili nel database relative all'utente con l'ID dato. Se invece il parametro id passato nella query viene modificato con un valore del tipo

```
"id=abcd OR 1=1"
```

allora la query eseguita all'interno del sistema sarà:

```
"SELECT * FROM Utenti WHERE id=abcd OR 1=1"
```

Una query eseguita in questo modo restituisce tutte le informazioni nel database per tutti gli utenti. Questo perché la seconda condizione "OR 1=1" è sempre verificata. Il requisito principale per la sicurezza dell'applicazione web è la convalioda dell'input per bloccare tutte le combinazioni di caratteri che potrebbero sfruttare questa vulnerabilità.

3.4.2 Cross-Site Scripting

Il Cross Site Scripting [15] è una seconda tipologia di attacco che sfrutta del codice "cattivo" per rubare informazioni ad un utente.

Come per gli attacchi injection, tutto il traffico dovrebbe essere visto come dannoso a priori e ogni fonte di dati dovrebbe essere considerata un possibile vettore di attacco. In questo caso,i possibili risultati di un attacco di cross-site scripting sono:

- il reindirizzamento delle persone verso pagine fasulle che emulano siti reali per rubare informazioni;
- il furto di credenziali agli utenti di sezioni private (ad esempio, nelle banche).
- il Defacement: consiste in un'operazione di pirateria con lo scopo di modificare il contenuto di un sito;

Come esempio di attacco XSS, si consideri il seguente scenario: il codice PHP in Fig.3.4, è il codice per la pagina example.php. Questo codice dovrebbe salutare l'utente con un messaggio di benvenuto, ricavando il nome utente dal parametro espresso nella richiesta al sito stesso, e visualizzando il nome utente sullo schermo. Ad esempio,una richiesta GET valida a un sito sarebbe simile a questa:

```
"http://sito.com/esempio.php?nome=Michele"
```

Tuttavia, in uno scenario simile, vediamo che l'input dell'utente non supera alcuna pre-convalida e viene eseguito direttamente. Pertanto, se l'utente immettesse il valore

```
'nome=michele<script>window.location.href
='http://sito2.com/home.php</script>'
```

come parametro anziché immettere solo il nome, l'input sarebbe ancora valido e il codice immesso verrebbe eseguito. Il codice iniettato reindirizza l'utente a un sito diverso dall'originale, possibilmente un sito falso per rubare informazioni. Pertanto, per proteggere adeguatamente un sito Web, si consiglia di convalidare tutti gli input per evitare attacchi XSS.

```
<html>
    <body>
        <p>Ciao <?php echo $_GET['name'] ?></php>
    </body>
</html>
```

Figura 3.4. Cross-Site Scripting

3.5 Web Application Firewall in commercio

Uno dei puntoi di partenza fondamentali per lo svolgimento del lavoro di tesi è stato l'analisi sulle tipologie di Web Application Firewall che sono attualmente in commercio. Questa analisi è importante in quanto uno degli obiettivi è quello di creare delle configurazioni che, successivamente, tramite un processo di raffinamento possano essere compatibili con tecnologie già esistenti. In questa sezione vengono, quindi, presentati e descritti brevemente i Web Application Firewall più comuni presenti sul mercato Open Source.

Questa prima fase di studio è stata importanta per potere definire la struttura all'interno di VEREFOO, come vedremo nei capitoli successivi.

3.5.1 ModSecurity

ModSecurity [16] è un firewall per applicazioni Web (WAF). Con oltre il 70% degli attacchi attualmente effettuati a livello di applicazione Web, le organizzazioni hanno bisogno di tutto l'aiuto possibile per proteggere i propri sistemi. Il WAF è implementato per creare un livello più elevato di sicurezza esterna per rilevare e/o prevenire gli attacchi prima che raggiungano l'applicazione web.

ModSecurity fornisce protezione dagli attacchi per le applicazioni Web e consente di monitorare e analizzare il traffico http in tempo reale. I server Web non possono registrare il corpo della richiesta. Gli aggressori lo sanno, motivo per cui una grande parte degli attacchi oggi viene eseguita utilizzando le richieste POST. ModSecurity fornisce la registrazione completa delle transazioni HTTP, consentendo la registrazione di richieste e risposte. Poiché alcune richieste e/o risposte possono contenere dati sensibili in determinati campi, ModSecurity può essere configurato

per mascherare questi campi prima che vengano scritti nel registro di controllo. Oltre alle funzionalità di registrazione, ModSecurity può monitorare il traffico HTTP in tempo reale per rilevare gli attacchi. In questo caso, ModSecurity funge da strumento di rilevamento delle intrusioni web e può rispondere a eventi sospetti che si verificano sui sistemi web.

ModSecurity può anche agire immediatamente per prevenire attacchi all'applicazione web. Esistono tre approcci ampiamente utilizzati.

- Modello di sicurezza negativo. Il modello di sicurezza negativo monitora le richieste di anomalie, comportamenti anomali e attacchi noti contro le applicazioni web.
- Modello di sicurezza positivo. Quando si distribuisce un modello di sicurezza positivo, vengono accettate solo le richieste note per essere valide e tutte le altre vengono rifiutate. Questo modello richiede la conoscenza dell'applicazione web da proteggere. Pertanto, ilmodello di sicurezza proattiva funziona al meglio con le applicazioni utilizzate di frequente ma raramente aggiornate, riducendo al minimo la manutenzione del modello.
- Debolezze e vulnerabilità note. Il linguaggio delle regole rende ModSecurity uno strumento di patching esterno ideale. Il patching esterno (a volte chiamato patching virtuale) ha lo scopo di ridurre le possibilità. Il tempo necessario per correggere le vulnerabilità delle applicazioni è spesso di settimane per molte organizzazioni. ModSecurity consente di applicare una patch a un'applicazione esternamente senza toccare (o nemmeno accedere) al codice sorgente dell'applicazione, mantenendo così il tuo sistema sicuro fino a quando l'applicazione non viene adeguatamente patchata.

ModSecurity è un firewall per applicazioni Web incorporabile. Ciò significa che si può implementare come parte dell'infrastruttura di un server Web esistente se è Apache, IIS7 o Nginx. Questo metodo di distribuzione ha una serie di vantaggi:

- Nessuna modifica alle reti esistenti. L'aggiunta di ModSecurity a un server Web esistente richiede pochissimo tempo. Inoltre,è completamente passivo per impostazione predefinita, quindi si può implementare in più fasi e si possono utilizzare solo le funzionalità di cui si hai bisogno. Si può facilmente rimuovere o disabilitare secondo necessità.
- Nessun singolo punto di errore. A differenza delle distribuzioni di rete, non vengono aggiunti nuovi puntidi errore al sistema.
- Bilanciamento e ridimensionamento del carico impliciti. Poiché funziona integrato nei server Web, ModSecurity sfrutta automaticamente le funzionalità aggiuntiva di bilanciamento del carico e scalabilità.
- Sovraccarico minimo. Poiché viene eseguito all'interno di un processo del server Web, non vi è alcun sovraccarico di rete e un sovraccarico minimo per l'analisi dei dati e la comunicazione.

- Nessun problema con i contenuti crittografati o compressi. Molti sistemi IDS hanno difficoltà ad analizzare il traffico SSL. ModSecurity è predisposto per funzionare quando il traffico viene decifrato e decompresso, quindi questo non è un problema.

ModSecurity utilizza le direttive per creare regole. Una direttiva è un comando scritto sotto forma di Sec(direttiva) che consente di inserire alcune configurazioni nel web application firewall. Ad esempio, la direttiva SecRule. Questa è una direttiva che consente agli amministratori di rete di creare nuove regole nel Web Application Firewall.

Il formato della regola è il seguente:

SecRule VARIABLES OPERATOR [ACTIONS]

Ogni regola deve fornire una o più variabili insieme all'operatore che dovrebbe essere utilizzato per ispezionarle. Se non vengono fornite azioni, verrà utilizzato l'elenco predefinito. Esiste sempre un elenco predefinito, anche se non è stato impostato in modo esplicito con la direttiva SecDefaultAction. Se sono presenti azioni specificate in una regola, verranno unite all'elenco predefinito per formare le azioni finali che verranno utilizzate. Le azioni nella regola sovrascrivono quelle nell'elenco predefinito.

3.5.2 Vulture

Vulture [17] consente di filtrare il traffico web in entrata e in uscita e bloccare minacce come injection, cross-site scripting... e altri attacchi di OWASP Top10. Si basa su modsecurity, moddefender (fork di Naxsi) e modsvm (Apprendimento automatico basato su Support Vector Machines) per filtrare il traffico HTTP.

- mod_security è il motore di filtraggio storico di Vulture.
- mod_svm è un modulo Apache scritto da aDvens che implementa l'apprendimento automatico non supervisionato. Funziona calcolando una rappresentazione matematica del "traffico buono" (codice Python nella GUI) e quindi, in tempo reale, confrontando le richieste in entrata e in uscita con questa rappresentazione. Il codice realtime è gestito da mod_svm, un modulo Apache sviluppato in linguaggio C. Se viene rilevata una "richiesta anormale", mod_svm aumenta il punteggio della richiesta, come fanno mod_security e mod_defender.
- mod_defender è un modulo firewall per applicazioni Web leggero, velocissimo e scalabile per Apache. mod_defender è un fork di Naxsi, un WAF progettato per NGINX. Funziona con un approccio whitelist: per impostazione predefinita, tutto è bloccato ed è consentito solo il "traffico esplicitamente accettato". mod_defender utilizza un file di configurazione delle regole principali, che contiene espressioni regolari su noti "pattern di attacco". Quando una regola

principale corrisponde, il punteggio della richiesta viene incrementato. Quando viene raggiunta la soglia di blocco, mod_defender blocca la richiesta. Oltre alle regole principali, mod_defender utilizza le regole di base per inserire esplicitamente nella whitelist le richieste buone. Le regole fortunatamente possono essere generate in modo automatico: mod_defender utilizza una "modalità di apprendimento": durante l'apprendimento, le richieste non vengono bloccate: vengono archiviate in un database (elasticsearch o mongodb) e quindi elaborate da uno script che costruirà la base regole.

3.5.3 IronBee

IronBee [18] è stato originariamente sviluppato e implementato dal team responsabile di ModSecurity. Per questo motivo, ci sono alcune somiglianze nei concetti e nella terminologia. Tuttavia, IronBee non è un fork di ModSecurity, ma un design completamente nuovo basato sull'esperienza con ModSecurity. Concettualmente, ModSecurity e IronBee non sono così diversi. Entrambi forniscono funzionalità di verifica delle transazioni HTTP. Lo fa fornendo dati HTTP analizzati che possono essere verificati utilizzando vari mezzi. È quindi possibile eseguire azioni come la registrazione o l'interruzione della transazione. Tuttavia, il design di ModSecurity e IronBee differisce notevolmente. ModSecurity è stato progettato per essere eseguito come un modulo del server Web Apache e quindi richiede l'API del server Web Apache per essere eseguito. Sebbene da allora il progetto abbia estratto questa API del server Web Apache e l'abbia utilizzata per il porting su altri server Web, questi port sono ancora sperimentali e non completi di funzionalità. ModSecurity è limitato all'essere configurato dal linguaggio di configurazione del server web nativo. IronBee è stato progettato come una piattaforma portatile che può essere facilmente estesa e integrata. IronBee è fondamentalmente solo una libreria condivisa che espone un'API e ti consente di caricare moduli esterni per estenderne le funzionalità. La piattaforma separa la ricerca e la composizione dei dati dal suo nucleo. Ciò significa che IronBee non decide come ricevere i dati HTTP o come definire le regole. Questo viene fatto al di fuori di IronBee. IronBee è configurato utilizzando il proprio file di configurazione, non la configurazione integrata del server. Questo ha diversi vantaggi:

- la configurazione non è limitata dal linguaggio di configurazione nativo;
- La configurazione non èc orrelata (potenzialmente in conflitto) con il server web.
- IronBee è configurato allo stesso modo (può condividere la configurazione) su diversi tipi di server

I linguaggi delle regole di IronBee sono definiti da moduli caricabili. Per questo motivo, IronBee è in grado di essere incorporato in tutto ciò che può trasmettergli dati e le regole possono essere scritte in varie forme, anche a livello di programmazione. Attualmente IronBee supporta l'integrazione in Apache TrafficServer, Apache Webserver, Nginx, Taobao Tengine (nginx fork), nonché uno strumento da riga di comando che accetta vari formati come input come HTTP grezzo, file pcap, ecc. Le

regole IronBee possono attualmente essere scritte in un linguaggio di regole semplicistico, che è simile a ModSecurity, configurato tramite il linguaggio Lua (sebbene eseguito nel motore C) o scritte in puro Lua.

A differenza di ModSecurity, che è configurato nella lingua di configurazione del server Web, IronBee è principalmente configurato nella propria lingua di configurazione con una configurazione minima sul server. IronBee ha a disposizione un linguaggio di configurazione completo, ma poiché la configurazione viene fornita tramite un'API, tutta la configurazione può essere eseguita a livello di codice in un modulo se lo si desidera. Esistono due differenze principali tra l'impostazione di ModSecurity e IronBee. IronBee definisce siti/posizioni separatamente dai server Web e il linguaggio delle regole di IronBee è diverso da ModSecurity.

IronBee implementa un solido framework per il monitoraggio e la difesa della sicurezza delle applicazioni. Fornisce una serie di funzionalità distribuite su diversi livelli di astrazione, consentendo agli utenti di scegliere l'approccio migliore per il lavoro che svolgono.

I comandi chiavi su cui lavora IronBee sono i seguenti:

- Site: i siti sono uno degli elementi principali della personalizzazione di IronBee. L'idea è di far corrispondere gli elementi al sito web reale. La maggior parte dei siti Web ha una mappatura dei nomi di dominio one-to-one, ma il meccanismo di mappatura è molto flessibile. Si può anche creare un sito per nome di dominio, più nomi di dominio per sito o condividere nomi di dominio su più siti web.
- SiteId: identificatore univoco del sito
- Rule: carica una regola. Quando si carica una regola nella maggior parte dei contesti, si può anche eseguire la regola in quel contesto. Tuttavia, il contesto di configurazione principale è speciale. Il caricamento di una regola nel contesto di configurazione principale non attiva la regola. Viene semplicemente caricato in memoria in modo che possa essere utilizzato in altri contesti. Per abilitare una regola, si deve utilizzare esplicitamente RuleEnable in un contesto diverso.
- RuleEnable: abilita una regola per l'esecuzione nel contesto di configurazione corrente.
- RuleDisable: disabilita l'esecuzione di una regola nel contesto di configurazione corrente

3.5.4 Squid

Squid [19] fornisce un ricco framework di controllo degli accessi, autorizzazione e registrazione per lo sviluppo di proxy Web e applicazioni per la distribuzione di contenuti. Squid viene fornito con molte opzioni di ottimizzazione del traffico, la maggior parte delle quali sono abilitate per impostazione predefinita per una facile configurazione e prestazioni migliori.

Squidi fornisce le seguenti configurazioni:

- Opzioni di autenticazione
- Controllo degli accessi
- Opzioni di rete
- Opzioni per il file di log

3.5.5 NAXSI

NAXSI [20] sta per Nginx Anti XSS e SQL Injection. Tecnicamente, questo è un modulo nginx di terze parti disponibile come pacchetto per molte piattaforme simili a UNIX. Per impostazione predefinita, questo modulo legge un piccolo sottoinsieme di regole semplici (e leggibilidall'uomo) contenenti il 99% dei modelli noti associati alle vulnerabilità del sito web. Ad esempio, ¡, — o drop nondevefar partedell'URI.

Essendo molto semplici, questi modelli possono corrispondere a query legittime, è dovere dell'amministratore di Naxsi aggiungere regole specifiche che inseriscano nella whitelist i comportamenti legittimi. Gli amministratori possono aggiungere le whitelist manualmente analizzando i log degli errori nginx o (consigliato) avviare il progetto con un'intensa fase di autoapprendimento che genera automaticamente le regole della whitelist relative al comportamento del sito web.

In altre parole, Naxsi si comporta come un firewall DROP per impostazione predefinita. L'unico compito è aggiungere le regole di ACCETTAZIONE necessarie affinché il sito Web di destinazione funzioni correttamente.

A differenza della maggior parte dei firewall per applicazioni Web, Naxsi non si basa su pattern basati su firme comegli antivirus, quindi non può essere aggirato-conpattern di attacco "sconosciuti".

Alcuni concetti chiave:

- WHITELIST: hanno lo scopo di istruire naxsi a ignorare modelli specifici in contesti specifici per evitare falsi positivi.
- RULES: hanno lo scopo di cercare corrispondenze all'interno di una richiesta per rilevare gli attacchi.
- CHECKRULES: indica a naxsi di eseguire un'azione (LOG, BLOCK, DROP, ALLOW) in base a un punteggio specifico associato alla richiesta. Questo punteggio è stato generalmente stabilito da una o più regole.
- DeniedUrl: è una direttiva che indica dove naxsi reindirizzerà le richieste bloccate.
- LearningMode: indica a naxsi di abilitare la modalità di apprendimento
- SecRulesEnabled: è una parola chiave obbligatoria per abilitare naxsi in una location
- SecRulesDisabled: può essere utilizzato per disabilitare in modo esplicito naxsi in una location

Qui di seguito le fugigure 3.5 e 3.6 mostrano sinteticamente come vengono costruite le regole e come funziona la CheckRule all'interno di NAXSI.

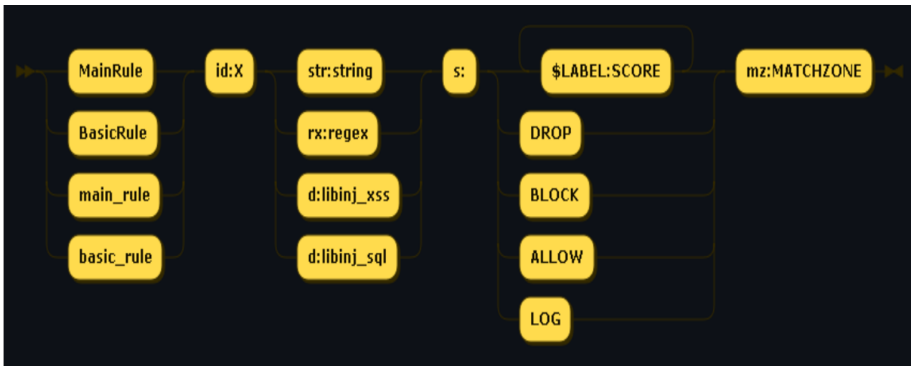

Figura 3.5. Rule NAXSI

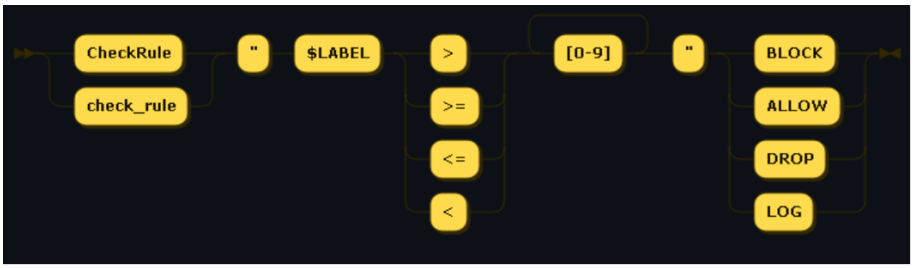

Figura 3.6. CheckRule NAXSI

Capitolo 4

Approccio della tesi

Questo capitolo descrive l'approccio adottato per il lavoro futuro. La prima sezione espone e spiega gli obiettivi della tesi. L'obiettivo finale è suddiviso in una serie di obiettivi minori che devono essere raggiunti per raggiungere l'obiettivo generale. La seconda sezione descrive i dettagli del modulo ADP, poiché è stato il modulo VEREFOO ad essere utilizzato nell'implementazione di questo documento. Infine, l'ultima sezione spiega come questo approccio si basa sulla dichiarazione del problema MaxSMT, spiega i principi alla base di esso, spiega la differenza tra limiti rigidi e limiti flessibili e cosa significano per il modulo ADP.

4.1 Obiettivo

Come discusso nel capitolo precedente e discusso in [7], la virtualizzazione della rete ha cambiato completamente gli approcci utilizzati per gestire e configurare la sicurezza della rete. Come già accennato, le tecnologie più conosciute in questo settore sono NFV e SDN. Insieme, queste due tecnologie forniscono un elevato grado di flessibilità e agilità poiché il nuovo software semplifica il lavoro dell'amministratore. In questo contesto, le configurazioni di sicurezza devono evolvere tanto rapidamente quanto la rete stessa, quindi la configurazione manuale diventa sempre più difficile. Sulla base di questa premessa, e alla luce dell'analisi fatta nei capitoli precedenti sui web application firewall e la loro configurazione, l'obiettivo di questo articolo è quello di modellare e implementare una soluzione in grado di configurare e distribuire i web application firewall su una rete automaticamente. Con o senza un web application firewall preassegnato, questa configurazione, a partire dal primo service graph, deve soddisfare una serie di requisiti fissati dall'amministratore di rete.
Per raggiungere questo obiettivo, il modulo ADP del framework VEREFOO, descritto in dettaglio nel capitolo precedente, è stato esteso. L'estensione del modulo è stata eseguita a piccoli passi.

- Il primo obiettivo era modellare e implementare un nuovo tipo di requisito di rete che consentisse agli amministratori di rete di configurare WAF utilizzando un linguaggio di alto o medio livello (HSPL o MSPL). Anche i risultati corrispondenti sono stati modellati e implementati insieme. Per raggiungere

questo primo obiettivo, siamo partiti dall'analisi di mercato WAF presentata nel capitolo precedente.

- Il secondo obiettivo era sviluppare e implementare una nuova funzionalità in VEREFOO che ci permettesse di creare un web application firewall. Una funzionalità che può essere utilizzata per assegnare e/o configurare automaticamente un WAF in base ai requisiti definiti dall'amministratore. Anche questo secondo obiettivo è stato suddivisoin tre operazioni minori per raggiungere l'obiettivo finale. L'unica operazione implementata passo dopo passo:

  1. Configurazione manuale
  2. Configurazione automatica
  3. Posizionamento automatico

L'obiettivo principale, in sisntesi, è stato quello di rendere il modulo ADP in grado di funzionare tramite una configurazione automatica seguendo alcuni criteri di ottimalità attraverso la risoluzione di un problema Max SMT, ricevendo in input l'insieme dei requisiti e il service o allocation graph. Il problema MaxSMT è formulato in modo da fornire indicazioni se la soluzione raggiunta sia formalmente corretta.

4.2 Modulo ADP

Nel capitolo precedente, abbiamo discusso il modello generale e il flusso di lavoro di VEREFOO,un modello che ci consente di tenere conto di diversi tipi di requisiti di rete a seconda del tipo di sicurezza che l'amministratore desidera implementare. Allocation Deployment Placement (ADP) è il modulo VEREFOO che questo documento è stato utilizzato per implementare. ADP è responsabile di due compiti importanti affinché VEREFOO funzioni correttamente:

- Orchestrazione e autoconfigurazione. Responsabile dell'inserimento di NSF nella rete ricevuti comeinput o generati dal Service Graph.
- Attività di posizionamento automatico VNF: posiziona i VNF rappresentati nel grafico del servizio di output dell'attività AOC (Service Graph iniziale più gli NSF aggiunti) in un grafo fisico che rappresenta l'infrastruttura fisica.

Il modulo ADP prevede in ingresso:

- Un insieme di requisiti specificati tramite un supporto linguaggio di medio livello;
- Un Service Graph a cui possono o meno essere assegnate funzioni di sicurezza e/o specificate dove lefunzioni di rete devono o non devono essere assegnate.

Avendo fornito questi input i possibili scenari possono essere:

- Se non viene trovata una soluzione che soddisfi tutti i requisiti di sicurezza dell'input,viene fornito un report di output che descrive i problemi che rendono impossibile il calcolo del grafo. Un possibile motivo dell'impossibilità è il posizionamento della funzione virtuale sul grafo fornito come input con numero di AP insufficienti. Un esempio è mostrato nella Figura 4.2.
- In caso contrario, viene generato automaticamente, secondo dei requisiti di ottimalità, un Service Graph composto da tutte le funzioni di rete richieste e quelle preallocate. Nello specifico di questa tesi per ogni WAF viene fornita una configurazione automatica, creata a partire dai requisiti espressi dall'amministratore, mediante un linguaggio di medio livello. Un esempio è mostrato in Figura 4.1;

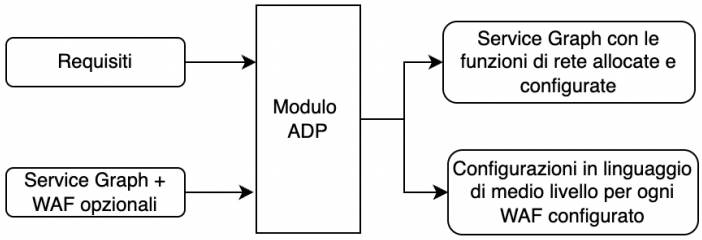

Figura 4.1. Output positivo

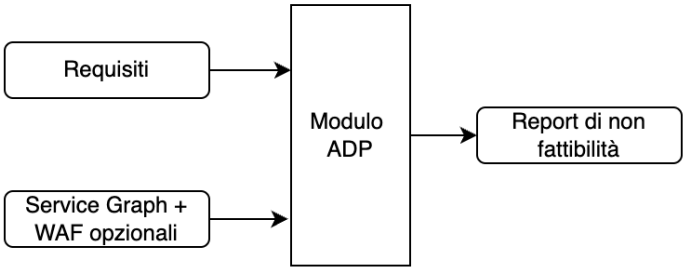

Figura 4.2. Output negativo

4.3 Formulazione del metodo

Seguendo l'approccio di ricerca di [6], questa tesi tenta di raggiungere il suo obiettivo formulando il problema dell'autoconfigurazione come un problema di Maximum Satisfiability Modulo Theories (MaxSMT), che fornisce allo stesso tempo un approccio formale e un'ottimizzazione economici. Il Maximum Satisfiability Modulo Theories problem (MaxSMT) è un'estensione del problema SMT nel contesto dell'ottimizzazione, dove dato un insieme di clausole di predicati contenenti variabili, l'obiettivo è trovare i valori ottimali di queste variabili affinchè raggiungano la massima soddisfacibilità delle clausole. Come il problema SMT, nel caso peggiore MaxSMT è NP-completo in termini di complessità computazionale. La differenza principale è che a ogni clausola viene assegnato un peso, unitario nella versione standard; di conseguenza, non è sufficiente trovare una soluzione che soddisfi le clausole del predicato, ma tra tutte le possibili soluzioni quella scelta deve massimizzare il numero di clausole soddisfatte.
Le varianti di MaxSMT più comuni sono:

- Weighted MaxSMT consente di assegnare pesi diversi a ciascuna clausola, privilegiando così la fattibilità delle clausole più importanti quando si cerca la soluzione migliore.
- la MaxSMT parziale, alcuni vincoli devono essere soddisfatti e quindi non possono essere ignorati, mentre altri elementi non devono essere soddisfatti per arrivare a una soluzione.
- la MaxSMT parziale ponderata, che è una combinazione delle due istanze precedenti.

Nel contesto di questa tesi viene adottata la MaxSMT parziale ponderata, per cui sono state definite due diverse categorie di clausole:

1. Clausole hard. Sono i vincoli non opzionali, il loro soddisfacimento è obbligatorio per ottenere una soluzione. Per questo motivo questo tipo di clausola è stata utilizzata per implementare il comportamento dei componenti del modulo ADP, al fine di garantire il corretto funzionamento dell'intero modulo. Ad esempio, l'affermazione "l'host finale può proteggere solo il proprio traffico" deve essere implementata attraverso l'uso di una clausola hard perché è qualcosa che deve essere sempre garantito.
2. Clausole soft. Sono i vincoli opzionali, quindi il loro soddisfacimento non è obbligatorio per ottenere una soluzione soddisfacente. Poiché è stato adottato un problema MaxSMT ponderato, ogni clausola soft ha un peso specifico che rappresenta la priorità assegnatagli; pertanto, il risolutore seleziona, tra tutte le soluzioni soddisfacibili, quella la cui somma di tutti i vincoli soft soddisfatti e considerando i loro pesi, è la più alta.

Grazie a questo tipo di vincoli è stato possibile gestire tutte quelle dichiarazioni che rappresentano preferenze da soddisfare per raggiungere una soluzione ottimale,

ad esempio la dichiarazione: "non utilizzare sistemi finali per imporre comunicazioni sicure", implementata come soft-constraint, ciò significa che il solutore cerca di trovare una soluzione senza utilizzare sistemi finali per imporre comunicazioni sicure, ma se non ne viene trovata una, cercherà di trovane una più soddisfacente utilizzando tali sistemi. Una definizione formale del problema MaxSMT parziale ponderato, dato un insieme H di vincoli hard e un insieme S di vincoli soft, può essere : $max \sum_{i=1}^{S} wi \times si$ subject to hj, $\forall j \in [1, H]$

Capitolo 5

Modellizzazione

Partendo dall'analisi preliminare fatta sui WAF, in questo capitolo si procede a formulare e a modellizzare la configurazione automatica dei web application firewall all'interno del framework.

5.1 Requisiti

I requisiti di sicurezza della rete rappresentano il secondo input di VEREFOO e possono essere formulati dall'utente del framework sfruttando due diverse rappresentazioni, in base alla sua esperienza in termini di sicurezza:

- se il progettista del servizio non conosce tutti i dettagli di sicurezza delle configurazioni delle Network Security Functions da implementare, quando deve definire i requisiti di input, può sfruttare una rappresentazione di alto livello, che è flessibile da gestire e semplice da capire.

- se invece il progettista del servizio sa come sono effettivamente configurate le funzioni di sicurezza ed è sicuro delle proprie capacità, può decidere di farlo immediatamente e sfruttare una rappresentazione di livello medio, inclusi dettagli come gli indirizzi IP e le porte in modo che lo strumento possa utilizzarle immediatamente per definire il problema MaxSMT, senza che sia necessaria un'ulteriore operazione di traduzione.

Più in dettaglio, i requisiti di sicurezza della rete che questa tesi ha analizzato sono i requisiti di connettività tra una coppia di endpoint; in particolare, loro possono essere classificati in due diverse tipologie:

- proprietà di raggiungibilità, se un punto finale deve essere in grado di raggiungerne un altro.

- proprietà di isolamento, se un punto finale non deve raggiungere un altro attraverso un qualsiasi percorso possibile.

5.2 Caratteristiche

Come descritto nei capitoli precedenti, i WAF sono caratterizzati da dei parametri specifici configurabili per creare delle regole. Vediamo adesso queli sono le caratteristiche che sono state scelte e rese disponibili ad un amministratore di rete. Le caratteristiche possono essere suddivise in due categorie:

- Azioni
- Campi

Le azioni possono essere:

- Deny: per bloccare la comunicazione del traffico che corrisponde alla configurazione impostata nel web application firewall, in caso di WAF preallocato, o alla configurazione automatica generata in caso di configurazione automatica creata a partire dai requisiti espressi dall'amministratore di rete.
- Allow: per consentire la comunicazione del traffico che corrisponde alla configurazione impostata nel web application firewall, in caso di WAF preallocato, o alla configurazione automatica generata in casa di configurazione automatica creata a partire dai requisiti espressi dall'amministratore di rete.
- Log/Check: comandi non ancora disponibili allo stato dell'arte dell'attuale framework, ma che possono tornare utili in lavori futuri per poter controllare il traffico senza che esso venga necessariamento bloccato o lasciato passare a seconda della configurazione di default.

I campi considerati invece sono i seguenti:

- IpSrc: è l'indirizzo IP di origine della comunicazione per cui il requisito è specificato;
- IpDst: è l'indirizzo IP di destinazione della comunicazione per cui il requisito è specificato;
- PortSrc: è la porta di origine a livello trasporto della comunicazione per cui il requisito è specificato;
- PortDst: è la porta di destinazione a livello trasporto della comunicazione per cui il requisito è specificato;
- TransportProto: è il protocollo a livello di trasporto della comunicazione per cui il requisito è specificato.
- Domain: è il nome del dominio per cui il requisito è specificato;
- Method: è il comando del protocollo HTTP della comunicazione per cui il requisito è specificato;

- Uri: nello specifico ci riferiamo all'URL della comunicazione per cui il requisito è specificato;
- Body: è il messaggio contenuto nella comunicazione per cui il requisito è specificato.

Per la rappresentazione degli indirizzi IP, ovvero IPSrc e IPDst, è stata usata la notazione punto-decimale:

$$ip1.ip2.ip3.ip4$$

dove ipi, $\forall$ i $\in$ 1,2,3,4, può essere un numero intero nell'intervallo da 0 a 255, estremi inclusi, o in alternativa un elemento jolly, identificato con $*$. Questo simbolo consente una dichiarazione univoca sia per un indirizzo di rete che per il corrispondente netmask, invece di due elementi separati: per esempio, la rappresentazione 20.0.0.$*$ può essere utilizzato per esprimere gli end point presenti nella rete 20.0.0.0/24, mentre il 30.0.$*$.$*$ caratterizza la rete 30.0.0.0/16.

La rappresentazione delle porte a livello di trasporto di origine e destinazione, portSrc e portDst, possono essere formulate usando un singolo numero o un intervallo di numeri, considerando un intervallo da 0 a 65535. Per chiarire questo concetto, se è richiesto che la fonte 10.0.0.1 non deve essere in grado di raggiungere la destinazione 20.0.0.1 attraverso i numeri di porta compresi nell'intervallo [1000, 2000], il flusso di traffico tra i due punti finali deve essere bloccato se la comunicazione è caratterizzata da un numero di porta sorgente incluso in questo intervallo, in modo che il requisito di isolamento sia soddisfatto.

La rappresentazione del transportProto rappresenta il protocollo di livello 4 utilizzato al di sopra del livello IP; la formulazione può avere TCP e UDP come possibili valori oppure, anche per questo componente il carattere jolly $*$. In questo caso, $*$ richiede che la soddisfacibilità deve essere garantita considerando la possibilità che la fonte possa inviare pacchetti TCP e UDP.

Per la rappresentazione del method si possono avere GET, POST, PUT, PATCH e DELETE come possibili valori, oppure anche per questa campo si può avere il carattere jolly $*$ che vuol dire un valore qualsiasi della lista appena menzionata.

Per la rappresentazione di URI, Body e Domain si prevede in input una stringa o il carattere jolly $*$. In questo caso, il carattere $*$ rappresenta qualsiasi stringa.

5.3 Requisti rappresentabili

Una volta descritte le caratteristiche principali in questa sezione vediamo, attraverso i campi e le azioni definite precedentemente, alcuni esempi dei requisiti e degli scenari ricopribili a livello teorico. Nei capitoli successivi vedremo come questi scenari verranno ricoperti e risolti in VEREFOO, attraverso la modellizzazione dei parametri definiti.

Scenario 1: partendo da una topologia semplice di Allocation Graph come quella mostrata in figura 5.1, l'amministratore di rete potrebbe voler bloccare la comunicazione con i due server se la comunicazione presenta il dominio di Facebook. In questo scenario un amministratore di rete potrebbe seguire, idelamente, due strade: una ottimale e una no.

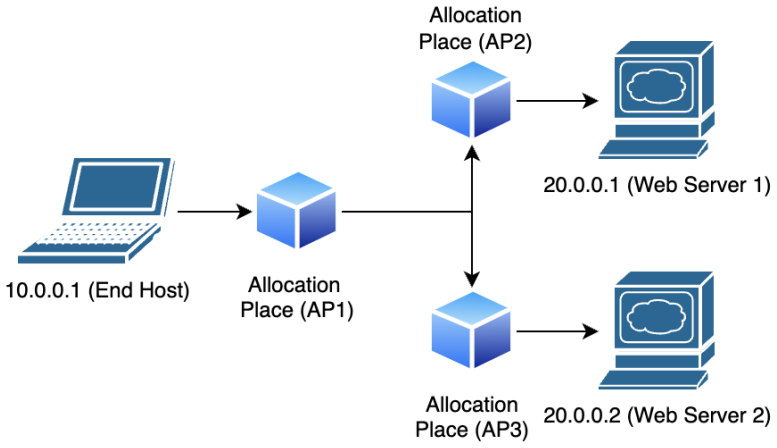

Figura 5.1. Scenario 1

- Soluzione non ottimale: l'amministratore di rete, vedendo due server distinti e volendo bloccare entrame le comunicazioni configura due requisiti diversi; il primo per bloccare la comunicazione con il dominio di Facebook proveninete dall'IP sorgente del server 1 e il secondo dall'IP sorgente del server 2. In questo caso la soluzione è corretta ma non ottimale.

- Soluzione ottimale: come mostrato nella figura 5.2, l'amministratore di rete, conoscendo la topologia della rete, configura un solo requisiti che rappresenta entrambi i requisiti: bloccare la comunicazione da tutti gli IP sorgenti per il dominio di Facebook. In questo caso, invece, la soluzione è sia corretta che ottimale ed è quella che vorremmo che il nostro framework generi.

Scenario 2: partendo dalla topologia come quella mostrata in figura 5.1, l'amministratore di rete richiede di bloccare il traffico di Facebook da tutti le sorgenti; inoltre richiede di bloccare il traffico segnato dal metodo HTTP GET proveniente

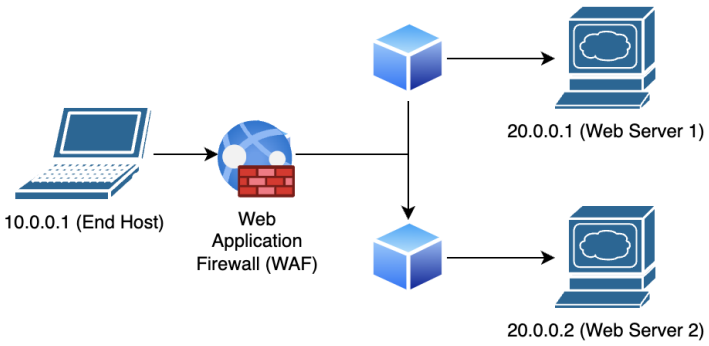

Figura 5.2. Soluzione WAF

dal Web Server 1 e di bloccare il traffico segnato dal metodo HTTP POST proveniente dal Web Server 2. In questo scenario le possibili soluzioni potrebbero essere le seguenti:

- Soluzione non ottimale: come mostrato nella figura 5.3, l'amministratore di rete potrebbe cercare di ottimizzare in maniera errata la soluzione. L'amministratore per ridurre il numero di regole posizione 3 Web Application Firewall sui 3 Allocation Places e configura 3 regole: su AP1 blocca il traffico segnato dal dominio Facebook, su AP2 blocca il traffico segnato dal metodo HTTP GET e su AP3 quello segnato dal metodo HTTP POST. Questa è una soluzione non ottimale, in quanto si andrebbero a configurare ben 3 WAF e condizione di ottimalità è ridurre il numero di WAF configurati sulla rete.
- Soluzione ottimale: come mostrato nella figura 5.2, l'amministatore di rete minimizza il numero di WAF da allocare. Per minimizzare il numero di elementi sulla rete, l'amministratore configura 2 WAF: uno su AP2 e uno su AP3. I due WAF configurati entrambi con due regole per soddifsare i requisti.

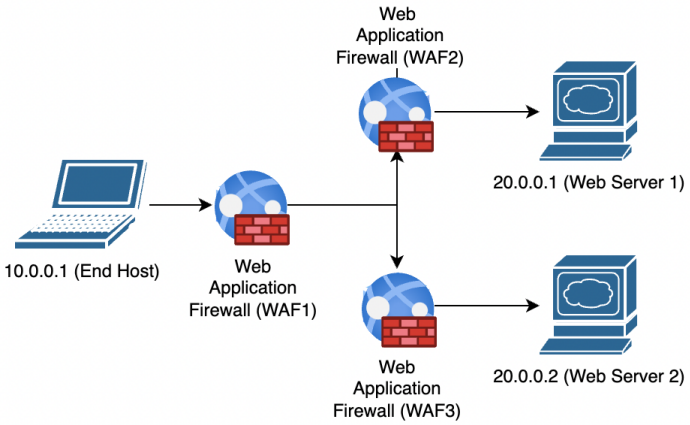

Figura 5.3. Soluzione non ottimale

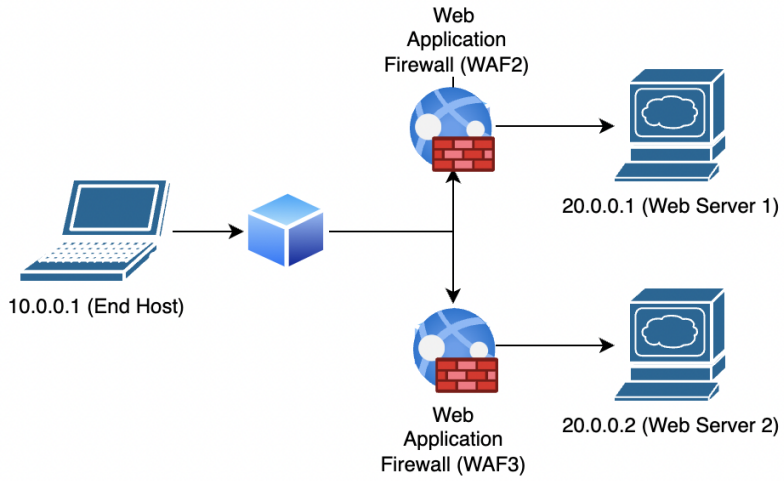

Figura 5.4. Soluzione ottimale

5.4 Limitazioni

In quest'ultima sezione del capitolo andiamo a descrivere la limitazione principale introdotta al momento per semplificare l'implementazione iniziale della configurazione automatica per i Web Application Firewall.

5.4.1 Carattere Jolly

La limitazione introdotta è stata riguardo alla gestione del carattere jolly all'interno delle stringhe. Per una corretta gestione del carattere jolly all'interno di requisiti come ad esemio il seguente:

```
<Property graph="0" name="ReachabilityProperty" src="10.0.0.1"
   dst="20.0.0.1">
    <HTTPDefinition url="*.it"/>
</Property>
```

dove il parametro relativo all'url richiederebbe che a tutte le comunicazioni con IP sorgente 10.0.0.1 e IP destinazione 20.0.0.1 sia aperta la raggiungibilità se il campo URL contenga un valore che termini in ".it".

Questa limitazione risulta abbastanza penalizzante in termini di prestazioni, come vedremo nei capitoli successivi, in quanto obbliga a duplicare le regole qualora si volesse, come in questo caso, richidere la raggiungibilità per più url con stessa sorgente e destinazione.

Capitolo 6

Soluzione

In questo capitolo entriamo nel dettaglio della soluzione. Nelle varie sezioni di questo capitolo viene spiegata la rappresentazione dei vari parametri configurabili in un Web Application Firewall.

6.1 Rappresemtazione XML

In questa prima sezione vediamo come sono stati modellati i vari elementi utilizzati, modificati e/o sfruttati in questa tesi.

6.1.1 Requisiti

Partiamo dal descrivere la struttura di un requisito che un amministratore di rete può esprimere, come mostrato nella figura 6.1, per configurare il Web Appication Firewall:

```
<xsd:complexType name="Property">
    <xsd:choice>
        <xsd:element name="HTTPDefinition" type="HTTPDefinition" minOccurs="0"/>
        <xsd:element name="POP3Definition" type="POP3Definition" minOccurs="0"/>
    </xsd:choice>
    <xsd:attribute name="name" type="P-Name" use="required"/>
    <xsd:attribute name="graph" type="xsd:long" use="required"/>
    <xsd:attribute name="src" type="xsd:string" use="required"/>
    <xsd:attribute name="dst" type="xsd:string" use="required"/>
    <xsd:attribute name="lv4proto" type="L4ProtocolTypes" default="ANY"/>
    <xsd:attribute name="src_port" type="xsd:string"/>
    <xsd:attribute name="dst_port" type="xsd:string"/>
    <xsd:attribute name="isSat" type="xsd:boolean"/>
    <xsd:attribute name="body" type="xsd:string"/>
</xsd:complexType>
```

Figura 6.1. XML Property

Dove le caratteristiche fondamentali ai fini di questa tesi sono:

- Name: è il nome del requisito;
- Graph: è il riferimento al grafico per cui è valido il requisito;

- Src e Dst: sono gli indirizzi IP sorgente e destinazione della comunicazione per cui deve essere valido il requisito;
- lv4proto: è il protocollo di livello 4 per cui deve essere valido il requisito
- SrcPort e DstPort: sono le porte per cui deve essere valifo il requisito;
- IsSat: conterrà il risultato del problema MaxSMT: true se soddisfatto, false altrimenti;
- HTTPDefinition: struttura che contiene i parametri che l'amministratore può configurare a livello applicativo;
  - Url
  - Domain
  - Body
  - Method

  La struttura viene mostrata nella figura 6.2

```
<xsd:complexType name="HTTPDefinition">
        <xsd:attribute name="url" type="xsd:string" />
        <xsd:attribute name="domain" type="xsd:string" />
        <xsd:attribute name="method" type="xsd:string" />
        <xsd:attribute name="body" type="xsd:string" />
        <xsd:attribute name="options" type="xsd:string" />
</xsd:complexType>
```

Figura 6.2. XML HTTPDefinition

6.1.2 Web Application Firewall

Passiamo ora alla definizione dell'elemento centrale della tesi, il web application firewall.

```
<xsd:element name="web_application_firewall">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="waf_elements" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="defaultAction" type="ActionTypes"/>
    </xsd:complexType>
</xsd:element>
```

Figura 6.3. XML Web Application Firewall

Come mostrato nella figura 6.3, la modellizzazione del WAF prevede due sotto-strutture wafElements e defaultActuon, dove:

- wafElements: mostrato nella figura 6.4, è la strttura interna che contiene i campi precedentemente descritti Action e Fields (Azioni e Campi)
- defaultAction: definisce se il waf deve lavorare in blacklisting o whitelisting; Se si desidera che il waf lavori in whitelisting allora la defaultAction sarà configurata come DENY, altrimenti ALLOW per il blackListing.

```
<xsd:element name="waf_elements">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Fields" minOccurs="0"></xsd:element>
            <xsd:element name="Actions" type="WAFActionType" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Figura 6.4. XML WAF Elements

```
<xsd:element name="Fields">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="IpSrc" type="xsd:string"  ></xsd:element>
            <xsd:element name="IpDst" type="xsd:string" ></xsd:element>
            <xsd:element name="Protocol" type="L4ProtocolTypes"></xsd:element>
            <xsd:element name="PortSrc" type="xsd:string"  ></xsd:element>
            <xsd:element name="PortDst" type="xsd:string"  ></xsd:element>
            <xsd:element name="Urls" type="url"></xsd:element>
            <xsd:element name="Method" type="method"  ></xsd:element>
            <xsd:element name="Domains" type="domain" ></xsd:element>
            <xsd:element name="Body" type="xsd:string"  ></xsd:element>
            <xsd:element name="Status" type="status" ></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Figura 6.5. XML Fields

6.2 Esempi di configurazione

In questa sezione verrano forniti alcuni esempi di configurazioni. Gli esempi verranno divisi in scenari distinti, ogni scenario rappresenta uno degli step seguiti per arrivare alla soluzione finale. Questi scenari sono:

1. Configurazione manuale
2. Configurazione automatica
3. Posizionamento automatico

6.2.1 Configurazione manuale

La configurazione manuale è stato il primo step implementato duranto lo svolgimento di questa tesi. La configurazione manuale prevede in input:

1. Un Service Grapgh, al cui interno sono presenti i WAF necessari preallocati e configurati;
2. Una serie di requisiti da rispettare;

In output si avrà, invece:

1. Un report di soddisfacibilità (SAT o UNSAT)
2. Il service graph fornito in input validato

Di seguito un esempio:

```
<node functional_type="WEB_APPLICATION_FIREWALL" name="30.0.0.1">
  <neighbour name="10.0.0.1"/>
  <neighbour name="20.0.0.1"/>
  <configuration description="A simple description" name="conf1">
      <web_application_firewall defaultAction="DENY">
          <waf_elements>
              <Fields>
                  <IpSrc>10.0.0.1</IpSrc>
                  <IpDst>20.0.0.1</IpDst>
                  <Protocol>TCP</Protocol>
                  <PortSrc>*</PortSrc>
                  <PortDst>*</PortDst>
                  <Urls>
                      <url>google.it</url>

                  </Urls>
                  <Method>ANY</Method>
                  <Domains>
                      <domain>*</domain>
                  </Domains>
                  <Body>*</Body>
                  <Status>*</Status>
              </Fields>

              <Actions>
                  <Action Type="Allow"></Action>
              </Actions>
          </waf_elements>
      </web_application_firewall>
  </configuration>
```

Figura 6.6. Service Graph con WAF preallocato

```
<PropertyDefinition>
      <Property graph="0" name="ReachabilityProperty" src="10.0.0.1" dst="20.0.0.1">
          <HTTPDefinition url="google.it"/>
      </Property>
</PropertyDefinition>
```

Figura 6.7. Requisiti input

```
15:18:41.712 [main] INFO result - SAT
15:18:41.715 [main] INFO result - ----------------------OUTPUT----------------------
15:18:41.721 [main] INFO result - <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NFV xsi:noNamespaceSchemaLocation="./xsd/nfvSchema.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <graphs>
        <graph id="0">
            <node name="10.0.0.1" functional_type="WEBCLIENT">
                <neighbour name="30.0.0.1"/>
                <configuration name="confA" description="A simple description">
                    <webclient nameWebServer="20.0.0.1"/>
                </configuration>
            </node>
            <node name="30.0.0.1" functional_type="WEB_APPLICATION_FIREWALL">
                <neighbour name="10.0.0.1"/>
                <neighbour name="20.0.0.1"/>
                <configuration name="conf1" description="A simple description">
                    <web_application_firewall defaultAction="DENY">
                        <waf_elements>
                            <Fields>
                                <IpSrc>10.0.0.1</IpSrc>
                                <IpDst>20.0.0.1</IpDst>
                                <Protocol>TCP</Protocol>
                                <PortSrc>*</PortSrc>
                                <PortDst>*</PortDst>
                                <Urls>
                                    <url>google.it</url>
                                </Urls>
                                <Method>ANY</Method>
                                <Domains>
                                    <domain>*</domain>
                                </Domains>
                                <Body>*</Body>
                                <Status>*</Status>
                            </Fields>
                            <Actions>
                                <Action Type="Allow"/>
                            </Actions>
                        </waf_elements>
                    </web_application_firewall>
                </configuration>
            </node>
            <node name="20.0.0.1" functional_type="WEBSERVER">
                <neighbour name="30.0.0.1"/>
                <configuration name="confB" description="A simple description">
                    <webserver>
                        <name>b</name>
                    </webserver>
                </configuration>
            </node>
        </graph>
    </graphs>
    <Constraints>
        <NodeConstraints/>
        <LinkConstraints/>
    </Constraints>
    <PropertyDefinition>
        <Property name="ReachabilityProperty" graph="0" src="10.0.0.1" dst="20.0.0.1" isSat="true">
            <HTTPDefinition url="google.it"/>
        </Property>
    </PropertyDefinition>
    <ParsingString></ParsingString>
</NFV>
```

Figura 6.8. Output

6.2.2 Configurazione automatica

La configurazione automatica è stato lo step successivo implementato duranto lo svolgimento di questa tesi. La configurazione automatica prevede in input:

1. Un Service Grapgh, al cui interno sono presenti i WAF necessari preallocati ma non necessariamente configurati;
2. Una serie di requisiti da rispettare;

In output si avrà, invece:

1. Un report di soddisfacibilità (SAT o UNSAT)
2. Il service graph fornito in input validato e configurato

Di seguito un esempio:

```
<node functional_type="WEB_APPLICATION_FIREWALL" name="30.0.0.1">
  <neighbour name="10.0.0.1"/>
  <neighbour name="20.0.0.1"/>
</node>
```

Figura 6.9. Service Graph con WAF preallocato ma non configurato

```
<PropertyDefinition>
      <Property graph="0" name="ReachabilityProperty" src="10.0.0.1" dst="20.0.0.1">
          <HTTPDefinition url="google.it"/>
      </Property>
      <Property graph="0" name="IsolationProperty" src="10.0.0.1" dst="20.0.0.1">
          <HTTPDefinition url="facebook.it"/>
      </Property>
</PropertyDefinition>
```

Figura 6.10. Requisiti input

```
15:27:54.266 [main] INFO result - ----------------------OUTPUT----------------------
15:27:54.269 [main] INFO result - <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NFV xsi:noNamespaceSchemaLocation="./xsd/nfvSchema.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <graphs>
        <graph id="0">
            <node name="10.0.0.1" functional_type="WEBCLIENT">
                <neighbour name="30.0.0.1"/>
                <configuration name="confA" description="A simple description">
                    <webclient nameWebServer="20.0.0.1"/>
                </configuration>
            </node>
            <node name="30.0.0.1" functional_type="WEB_APPLICATION_FIREWALL">
                <neighbour name="10.0.0.1"/>
                <neighbour name="20.0.0.1"/>
                <configuration name="AutoConf">
                    <web_application_firewall defaultAction="ALLOW">
                        <waf_elements>
                            <Fields>
                                <IpSrc>10.0.0.1</IpSrc>
                                <IpDst>20.0.0.1</IpDst>
                                <Protocol>ANY</Protocol>
                                <PortSrc>*</PortSrc>
                                <PortDst>*</PortDst>
                                <Urls>
                                    <url>"facebook.it"</url>
                                </Urls>
                                <Method>ANY</Method>
                                <Domains>
                                    <domain>"*"</domain>
                                </Domains>
                                <Body>"*"</Body>
                                <Status>"*"</Status>
                            </Fields>
                            <Actions>
                                <Action Type="Block"/>
                            </Actions>
                        </waf_elements>
                    </web_application_firewall>
                </configuration>
            </node>
            <node name="20.0.0.1" functional_type="WEBSERVER">
                <neighbour name="30.0.0.1"/>
                <configuration name="confB" description="A simple description">
                    <webserver>
                        <name>b</name>
                    </webserver>
                </configuration>
            </node>
        </graph>
    </graphs>
    <Constraints>
        <NodeConstraints/>
        <LinkConstraints/>
    </Constraints>
    <PropertyDefinition>
        <Property name="ReachabilityProperty" graph="0" src="10.0.0.1" dst="20.0.0.1" isSat="true">
            <HTTPDefinition url="google.it"/>
        </Property>
        <Property name="IsolationProperty" graph="0" src="10.0.0.1" dst="20.0.0.1" isSat="true">
            <HTTPDefinition url="facebook.it"/>
        </Property>
    </PropertyDefinition>
    <ParsingString></ParsingString>
</NFV>
```

Figura 6.11. Output

6.2.3 Posizionamento automatico

Come ultimo step è stato implementato il supporto per il posizionamento automatico di un web application firewall all'interno di una rete in base alla necessità espressa da un amministratore di rete., tramite requisiti. Questo è stato lo step conclusivo in quanto tutta la sua realizzazione è stato il punto di raccordo di tutti gli sviluppi realizzati. Per fornire un esempio di questo step, infatti, è necessario parlare di posizione automatico più configurazione automatica di un web application firewall. In questo caso in input avremo:

1. Un Service Grapgh, al cui interno non sono presenti WAF nè preallocati nè configurati;
2. Una serie di requisiti da rispettare;

In output si avrà, invece:

1. Un report di soddisfacibilità (SAT o UNSAT)
2. Il service graph fornito in input validato, con i WAF ritenuti necessari dall'algoritmo allocati e configurati

Di seguito un esempio:

```
<graphs>
  <graph id="0">
   <node functional_type="WEBCLIENT" name="10.0.0.1">
      <neighbour name="30.0.0.1"/>
      <configuration description="A simple description" name="confA">
        <webclient nameWebServer="20.0.0.1"/>
      </configuration>
    </node>
    <node name="30.0.0.1">
      <neighbour name="10.0.0.1"/>
      <neighbour name="20.0.0.1"/>
    </node>
    <node functional_type="WEBSERVER" name="20.0.0.1">
      <neighbour name="30.0.0.1"/>
      <configuration description="A simple description" name="confB">
        <webserver>
          <name>b</name>
        </webserver>
      </configuration>
    </node>
  </graph>
</graphs>
```

Figura 6.12. Service Graph input

```
<PropertyDefinition>
      <Property graph="0" name="ReachabilityProperty" src="10.0.0.1" dst="20.0.0.1">
          <HTTPDefinition url="google.it"/>
      </Property>
      <Property graph="0" name="IsolationProperty" src="10.0.0.1" dst="20.0.0.1">
          <HTTPDefinition url="facebook.it"/>
      </Property>
</PropertyDefinition>
```

Figura 6.13. Requisiti input

```
15:27:54.266 [main] INFO result - ----------------------OUTPUT----------------------
15:27:54.269 [main] INFO result - <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NFV xsi:noNamespaceSchemaLocation="./xsd/nfvSchema.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <graphs>
        <graph id="0">
            <node name="10.0.0.1" functional_type="WEBCLIENT">
                <neighbour name="30.0.0.1"/>
                <configuration name="confA" description="A simple description">
                    <webclient nameWebServer="20.0.0.1"/>
                </configuration>
            </node>
            <node name="30.0.0.1" functional_type="WEB_APPLICATION_FIREWALL">
                <neighbour name="10.0.0.1"/>
                <neighbour name="20.0.0.1"/>
                <configuration name="AutoConf">
                    <web_application_firewall defaultAction="ALLOW">
                        <waf_elements>
                            <Fields>
                                <IpSrc>10.0.0.1</IpSrc>
                                <IpDst>20.0.0.1</IpDst>
                                <Protocol>ANY</Protocol>
                                <PortSrc>*</PortSrc>
                                <PortDst>*</PortDst>
                                <Urls>
                                    <url>"facebook.it"</url>
                                </Urls>
                                <Method>ANY</Method>
                                <Domains>
                                    <domain>"*"</domain>
                                </Domains>
                                <Body>"*"</Body>
                                <Status>"*"</Status>
                            </Fields>
                            <Actions>
                                <Action Type="Block"/>
                            </Actions>
                        </waf_elements>
                    </web_application_firewall>
                </configuration>
            </node>
            <node name="20.0.0.1" functional_type="WEBSERVER">
                <neighbour name="30.0.0.1"/>
                <configuration name="confB" description="A simple description">
                    <webserver>
                        <name>b</name>
                    </webserver>
                </configuration>
            </node>
        </graph>
    </graphs>
    <Constraints>
        <NodeConstraints/>
        <LinkConstraints/>
    </Constraints>
    <PropertyDefinition>
        <Property name="ReachabilityProperty" graph="0" src="10.0.0.1" dst="20.0.0.1" isSat="true">
            <HTTPDefinition url="google.it"/>
        </Property>
        <Property name="IsolationProperty" graph="0" src="10.0.0.1" dst="20.0.0.1" isSat="true">
            <HTTPDefinition url="facebook.it"/>
        </Property>
    </PropertyDefinition>
    <ParsingString></ParsingString>
</NFV>
```

Figura 6.14. Output

6.3 MaxSMT Problem

Z3 è un risolutore per SMT di Microsoft Research che è mirato a risolvere i problemi che sorgono nei contesti di verifica e analisi del software. Z3 è diventato open source con una licenza MIT. Supporta Windows, OSX, Linux e FreeBSD. È anche possibile chiamare Z3 proceduralmente utilizzando una varietà di API disponibili in C, C++, Java, .Net, OCaml e Python.
Il flusso di lavoro del framework è presentato nella figura 6.7.

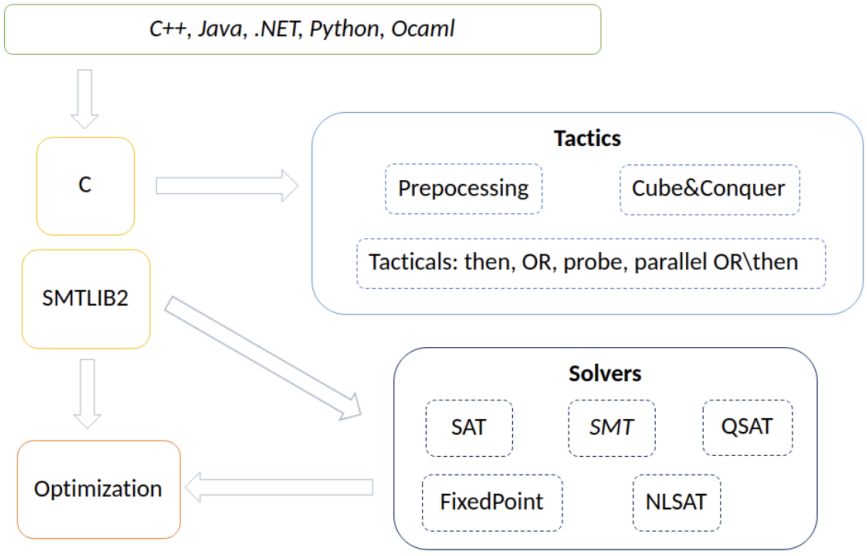

Figura 6.15. Workflow Z3

- Vengono fornite una serie di formule in input al framework utilizzando uno dei linguaggio di programmazione supportato da Z3.
- Una mappatura uno-a-uno dal dominio dei dati del programma in fase di test al dominio dati di Z3 che viene eseguito, tutte queste formule sono tradotto in un file SMTLIB2.
- Z3 cerca di applicare tattiche come la pre-elaborazione per limitare il tempo totale di calcolo o l'euristica per ottenere una soluzione non ottimale.
- Infine, seleziona un risolutore specifico (ad es. SMT) per calcolare se per una soluzione o meno è necessaria una fase di ottimizzazione, per calcolare la soluzione ottimale; per la fase di ottimizzazione, in particolare, z3 sfrutta un motore ottimizzatore, denominato z3Opt.

Di seguito, nella figura 6.8, viene presentato un esempio di MaxSMT parziale pesato modellato in lingua Z3 e la soluzione corrispondente

La prima metà dell'immagine mostra le espressioni fornite come input a Z3. Afferma che la variabile booleana "a" deve essere uguale alla variabile booleana

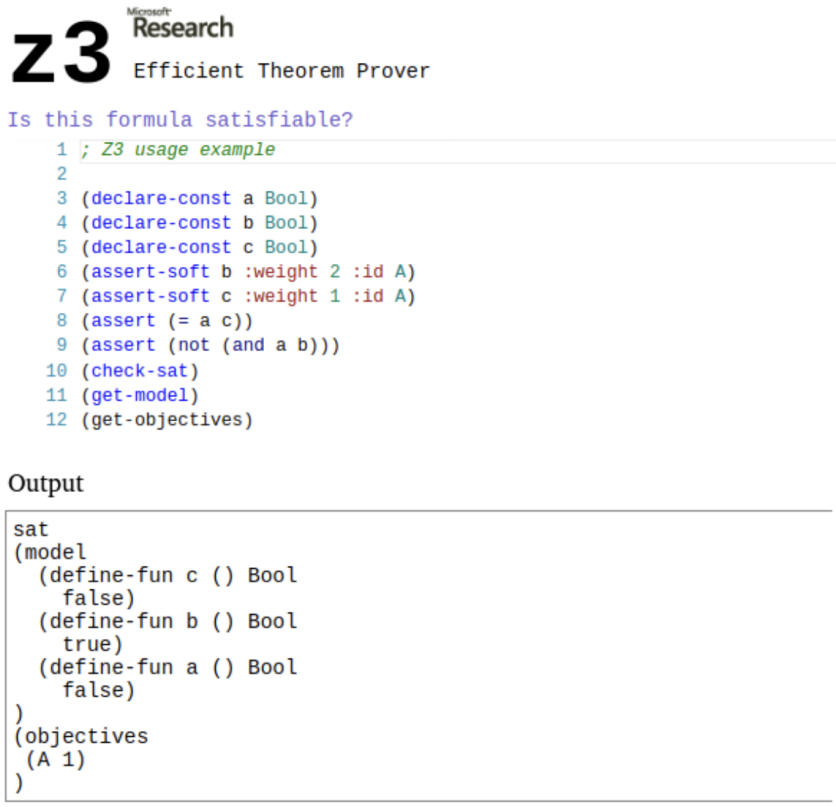

Figura 6.16. Esempio Z3

"c" e che "a" e la variabile booleana "b" non possono essere entrambe vere contemporaneamente. Dati questi vincoli "hard", ci sono due soluzioni. Primo, dove "a" e "c" sono uguali a "vero" e "b" è uguale a "falso". Il secondo è dove "a" e "c" sono entrambi "falsi" e "b" sono "veri". Se possibile, ci sono anche due vincoli "soft" che affermano che le variabili "b" e "c" devono essere "vere" ma hanno pesi diversi, quindi Z3 emette il secondo vincolo, mostrato nella seconda metà della figura..

Di seguito, mostrato nella figura 6.9, vengono forniti alcuni esempi per mostrare come funziona la libreria java Z3 e di come è stata sfruttata per impostare il problema MaxSMT:

```
Expr src = ctx.mkConst(waf + "_manual_src_"+i+"_"+y, nctx.addressType);
Expr dst = ctx.mkConst(waf + "_manual_dst_"+i+"_"+y, nctx.addressType);
Expr proto = ctx.mkConst(waf + "_manual_proto_"+i+"_"+y, ctx.mkIntSort());
Expr srcp = ctx.mkConst(waf + "_manual_srcp_"+i+"_"+y, nctx.portType);
Expr dstp = ctx.mkConst(waf + "_manual_dstp_"+i+"_"+y, nctx.portType);
Expr dom = ctx.mkConst(waf + "_manual_dom_"+i+"_"+y, ctx.getStringSort());
Expr bod = ctx.mkConst(waf + "_manual_bod_"+i+"_"+y, ctx.getStringSort());
Expr met = ctx.mkConst(waf + "_manual_met_"+i+"_"+y, ctx.getStringSort());
Expr url = ctx.mkConst(waf + "_manual_url_"+i+"_"+y, ctx.getStringSort());
```

Figura 6.17. Constanti Z3

Nella figura si vede come viene dichiarata una costante Z3 in java, vediamo nella figura diversi tipi di costanti: addressType per gli indirizzi IP, una strttura complessa dove ogni byte dell'Ip e un elemento distinto, IntSort per gli interi, PortType per le porte, rappresenta la valorizzazione di un intervallo da 0 a 65535 e infine lo StringSort per le stringhe generiche.

La seconda immagine, figura 6.10, mostra l'impostazione dei vincoli hard relativi al Web Application Firewall:

```
if(predicate.getMehods()!=null && predicate.getMehods().size()>0) {
        for(String s : predicate.getMehods()) {
                if(s!=null) {
                        if(tmp!=null) tmp = ctx.mkOr(tmp,ctx.mkEq( met,ctx.mkString(s)));
                        else tmp = ctx.mkEq( met,ctx.mkString(s));
                }
        }
}
if(tmp==null) retExpr = ctx.mkAnd(retExpr,ctx.mkEq(met, ctx.mkString("*")));
//else retExpr = ctx.mkAnd(retExpr,tmp);
else retExpr = ctx.mkAnd(retExpr,ctx.mkOr(tmp, ctx.mkEq(met, ctx.mkString("*"))));
```

Figura 6.18. Vincolo Hard

Obiettivi del problema MaxSMT

I due principali obiettivi del problema MaxSMT, in relazione ai waf, sono i seguenti:

- Ridurre al minimo il numero di Web application firewall assegnati al Service Graph per ridurre al minimo il consumo di risorse utilizzato dai VNF implementati per istanza virtuale (RAM, CPU, ecc.);

- Ridurre al minimo il numero di regole in ogni WAF, è più facile per gli amministratori del servizio leggere e gestire la configurazione risultante e modificarla con meno rischi.

Sebbene i due obiettivi siano teoricamente indipendenti, il modello z3 del problema MaxSMT è modellato con variabili condivise, quindi entrambe le soluzioni ottimali vengono raggiunte simultaneamente. Tuttavia, ridurre al minimo ilnumero di quote waf è una priorità, poiché la distribuzione di funzionalità aggiuntive è molto più costosa rispetto all'aggiunta di nuove regole alle istanze waf esistenti.

Capitolo 7

Implementazione e Validazione

7.1 Implementazione

In questa sezione il punto centrale è la spiegazione di come è stata implementata effettivamente la costruzione e l'allocazione automatica dei WAF sull'Allocation Graph fornito in input.

Il processo di autoconfigurazione prevede in input:

- Un service o allocation graph, che rappresenta la struttura della rete da amministrare
- Da uno a N requisiti
- opzionalmente dei waf configurati forzatamente in posizioni predefinite

Ricevuti questi input si hanno due scenari:

- Configurazione manuale: l'amministratore di rete definisce la configurazione in un web application firewall pre allocato sul grafo di rete
- Configurazione automatica: l'amministratore di rete non preconfigura nulla, vengono solo definiti i requisiti da soddisfare

In base a questi due scenari si avranno due algoritmi simili ma con alcune differenze.

7.1.1 Configurazione manuale

Partiamo dalla configurazione manuale. In questo scenario l'algoritmo prevede due fasi di generazione vincoli:

- si generano prima i vincoli da rispettare secondo quanto configurato sui waf esistenti

- si generano poi i vincoli calcolati rispetto ai requisiti da soddisfare

A partire da questi due set di vincoli, la logica stabilirà poi la loro coerenza e la fattibilità rispetto a quanto indicato.

L'algoritmo prevede che per ogni WAF si segua il seguente processo. Dato in input un waf configurato come il seguente:

```
<web_application_firewall defaultAction="DENY">
    <waf_elements>
        <Fields>
            <IpSrc>10.0.0.1</IpSrc>
            <IpDst>20.0.0.1</IpDst>
            <Protocol>TCP</Protocol>
            <PortSrc>*</PortSrc>
            <PortDst>*</PortDst>
            <Urls>
                <url>www.google.it</url>
                <url>www.facebook.com</url>
            </Urls>
            <Method>*</Method>
            <Domains>
                <domain>wikipedia</domain>
            </Domains>
            <Body>*</Body>
            <Status>*</Status>
        </Fields>
        <Actions>
            <Action Type="Allow"></Action>
        </Actions>
    </waf_elements>
</web_application_firewall>
```

Si avrà che il comportamento di default sarà di DENY, quindi si avrà un comportamento di tipo whitelisting. In questo caso l'unica comunicazione richiesta da lasciar passare è quella che va dai 2 nodi 10.0.0.1 a 20.0.0.1 che sfrutta il protocollo TCP per il trasporto e che contenga nell'url o "www.google.it" o "www.facebook.com".

Per la costruzione dei vincoli nel modello Z3 si procede nel seguente modo: per ogni combinazione di url e domini specificato, se specificati, si crea un set di vincoli hard nel modello z3. Nello specifico dell'esempio sopra riportato avremo, quindi, 2 gruppi di vincoli:

- uno per la combinazione "www.google.it" - "wikipedia"
- l'altro per la combinazione "www.facebook.com" - "wikipedia"

questi due gruppi avranno in comune IpSrc, IpDst e Protocol.

Definiti quindi i vincoli per il WAF configurato si passa alla generazione dei vincoli per i requisiti specificati. Un esempio di requisito è il seguente:

```
<Property graph="0" name="ReachabilityProperty" src="10.0.0.1"
   dst="20.0.0.1">
    <HTTPDefinition url="www.google.it"/>
</Property>
```

Il requisito così espresso richiede che il traffico che va da 10.0.0.1 a 20.0.0.1 che viaggia con url pari a "www.google.it". Per questo requisito verrà creato un'altro set di vincoli hard.

Alla fine dell'algoritmo tutti questi set verrano passati al verificatore di Z3 che produrrà in output, come definito nei capitoli precedenti, un report di fattibilità o meno; in caso di fattibilità verra fornito anche un grafo con la configurazione del/dei waf allocati.

7.1.2 Configurazione automatica

In questa casistica non avremo nessuna configurazione preimpostata dall'ammnistratore di rete, ma avremo solo un set di requisiti richiesti da rispettare.

Come per la configurazione manuale a partire dai requisiti scritti nella forma:

```
<Property graph="0" name="ReachabilityProperty" src="10.0.0.1"
   dst="20.0.0.1">
    <HTTPDefinition url="www.google.it"/>
</Property>
```

Si crea un set di vincoli hard da passare a Z3.

La differenza in questo caso è che non abbiamo nessuna configurazione di partenza da cui partire per controllare la fattibilità o meno di tali requisiti. Per la fattibilità sarà compito di Z3 costruire, o provare, una configurazione per il waf presente nel grafo che possa soddisfare tutti i requisiti richiesti. In caso negativo viene fornito un report di non fattiblità, in caso positivo invece oltre ad essere fornito un messaggio di fattbilità viene fornito in output anche il grafo passato in input con il waf allocato e configurato secondo quanto richiesto dai vincoli

In aggiunta, in questo scenario sono previsti anche dei vincoli soft da passare al motore z3, per cercare una soluzione ottimale che rispetti i requisiti. I vincoli soft aggiunti sono per cercare di:

- impostare se possibili gli Ip con il valore di "wildcard" affinchè non si generi la regola;
- usare la "wildcard" per il protocollo;
- usare la "wildcard" per la porta sorgente;

- usare la "wildcard" per la porta destinzazione;
- usare la "wildcard" per il metodo HTTP;

Infine, se il WAF è autoposizionato, viene generata la configurazione sfruttando il modello fornito in output dal risolutore di x3.

7.2 Validazione

Quanto detto fino ad ora rimane a livello teorico e concettuale. In questa sezione si vede più l'aspetto pratico e come si comportano gli algoritmi definiti in questa tesi andando ad analizzare alcuni Use Case per vederene funzionamento, scalabilità e performance.

7.2.1 Use Case 1

Il primo caso di studio è il più semplice e immediato. Prendendo come esempio una topologia molto basilare, senza loop e senza complicazioni. Il grafo di riferimento è quello mostrato in figura 7.1:

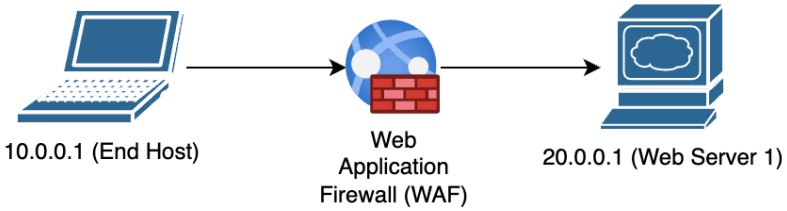

Figura 7.1. Use Case 1

A partire da questo grafo di esempio prendiamo in considerazione le due tipologie di configurazioni:

- Manuale: per la configurazione manuale il calcolo di fattiblità è pressochè immediato; nel primo test effettuato è stato considerato un solo requisito.
- Automatica: anche per la configurazione automatica il calcolo di fattiblità è pressochè immediato; anche in questo secondo test effettuato è stato considerato un solo requisito.

Per entrambe le tipologie di configurazione per topologie semplici e numero di requisiti ridotti i tempi di elaborazione sono molto simili e ridotti. In questi due scenari, molto semplici, presentati i tempi sono gli stessi e corrispondono a 230ms.

7.2.2 Use Case 2

In questo secondo scenario vediamo come evolve il tempo di elaborazione in caso di topologie più complesse. Prendiamo come riferimento la figura 7.2.

Questa figura è stata semplificata rispetto a quanto utilizzato per effettuare il test, ma è utile per rendere l'idea della complessità di un grafo più ampio, come

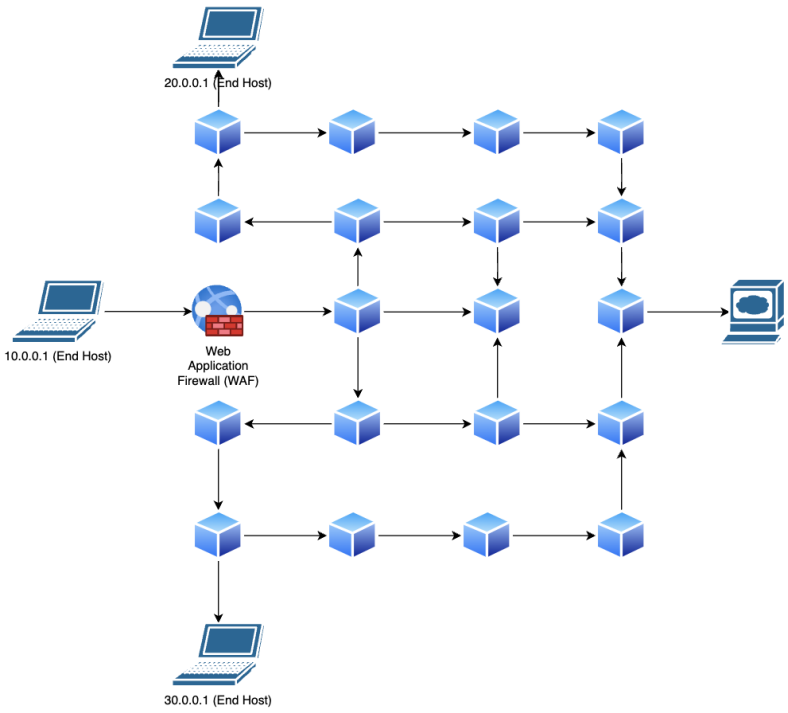

Figura 7.2. Use Case 2

possiamo vedere ci sono un numero elevato di nodi e di conseguenza di percorsi plausibili, ci possono essere anche diversi cicli. Per il primo test effettuato è stata considerata una topologia simile a quella mostrata in figura ma con 100 Nodi, 1 WAF e 10 requisiti.

In questo contesto il nostro framework continua a lavorare correttamente ma ha un tempo di calcolo molto più elevato, come si può immaginare, e, infatti, abbiamo che il tempo utile per calcolare la fattbilità di configurazione in uno scenario del genere sale a 12039ms.

Risfruttando la stessa topologia abbiamo poi provato a stressare ulteriormente il framework aggiungendo ulteriori requisiti. Il risultato, però, è stato che per una topologia simile con un numero superiore ai 10 requisiti fa esplodere il tempo di calcolo. Con questo secondo test abbiamo quindi incontrato il nostro primo limite.

7.3 Testing

Dopo aver visto qualche use case di utilizzo del nostro framework, in questa sezione andiamo a vedere alcuni test svolti. In particolare, sono stati svolti tre tipi di test:

- Rete di piccola dimensione con numero di requisiti crescente
- Rete di media dimensione con numero di requisiti crescente
- Numero fisso di requisiti con numero di Allocation Places crescente

Nel primo caso, mostrato in figura 7.3, vediamo un esempio di come evolve il tempo computanzionale in relazione al numero di requisiti crescente. In blu vediamo un esempio di url filtering, in verde, invece, un esempio di domain filtering. Alla luce di questo test vediamo come il tempo cresce linearmente rispetto al numero di requisiti fonriti in input dall'amministratore

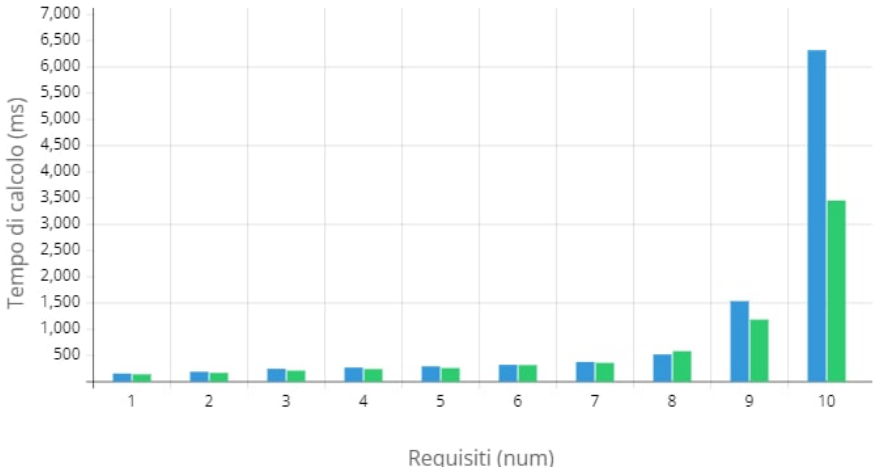

Figura 7.3. Url vs domain filtering

Nel secondo caso, si parte da una rete di dimensioni più estesa (20 AP) e si è ripetuto lo stesso test. Come si può vedere dal grafico in figura 7.4, anche in questo caso abbiamo una crescita lineare del tempo computanzionale in relazione al numero di requisito. In questo secondo esempio vediamo come, oltre al tempo crescente in base al numero di requisiti, abbiamo un tempo "fisso" crescente in base al numero di AP dovuto all'algoritmo del framework stesso per arrivare alla definizione del modello da dare in input al risolutore di z3

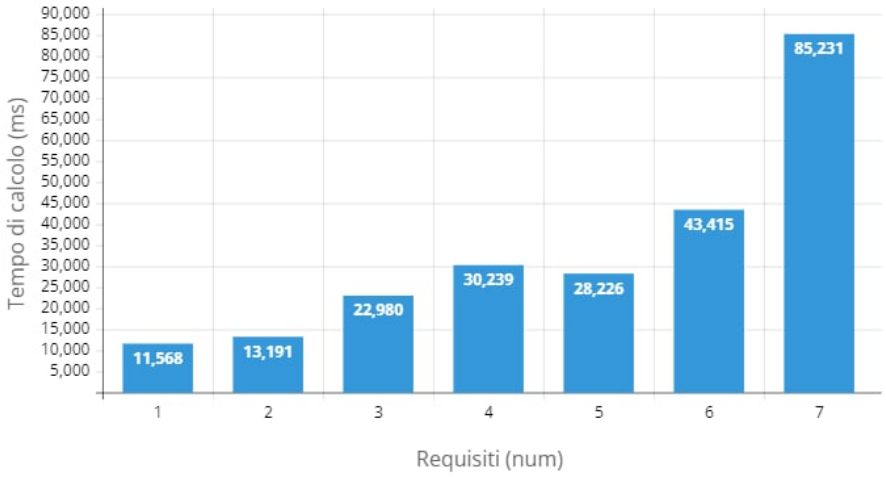

Figura 7.4. Rete di grande dimensione, numero di requisiti crescente

Nell'ultimo test svolto, abbiamo mantenuto fissi i requisiti e variato gli AP. In questo scenario, figura 7.5, vediamo come avendo un numero basso, in verde, di requisiti il tempo di calcolo cresce abbastanza linearmente rispetto agli AP. Aumentando il numero di requisiti, aumenta notevolmente il tempo di calcolo. Da quest'ultimo esempio possiamo dedurre che il punto critico è la quantità di vincoli definiti all'interno della rete dall'amministratore.

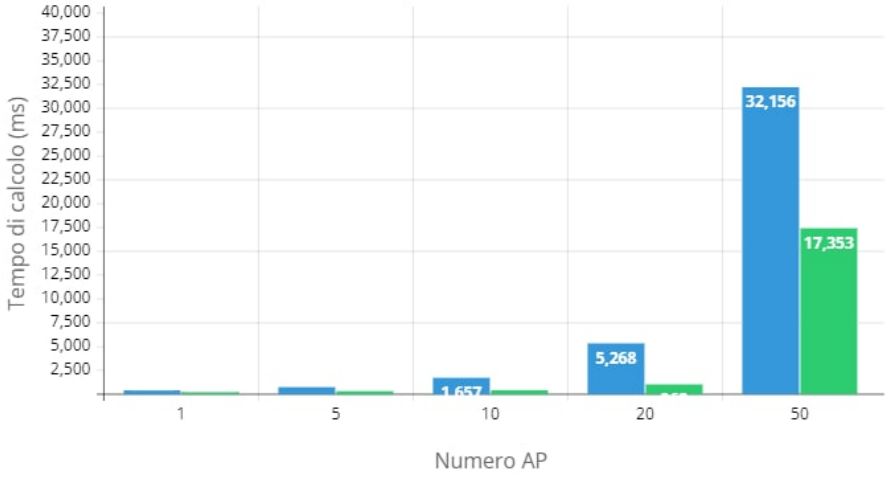

Figura 7.5. AP crescenti, requisiti fissi

Da questi risultati possiamo concludere che per i lavori futuri un passaggio sicuro e necessario sarà quello di ottimizzare quanto sviluppato per cercare di ridurre

queste tempistiche e poter lavorare su una scala ancora maggiore, per fare questo il punto di partenza deve essere la minimizzazione dei vincoli.

Capitolo 8

Conclusioni e lavori futuri

8.1 Conclusioni

Durante il lavoro di tesi, sono stati sviluppati e progettati miglioramenti del modulo ADP di VEREFOO, con lo scopo di estendere le capacità del framework. Questo lavoro è stato eseguito con l'obiettivo futuro di utilizzare il framework sviluppato in ambiente NFV, sfruttando tutti i vantaggi offerti dal disaccoppiamento tra le funzioni virtuali e quelle fisiche.

Come prima cosa è stato fatto uno studio completo su quali sono gli scenari di utilizzo più importanti dei web application firewall, per capire quali fossero i comportamenti principali e quali quelli comuni a tutti i web application firewall presi in esame. Dopo averli identificati, l'attenzione si è spostata sul design e sullo sviluppo delle fasi di allocazione e configurazione dei Web Application Firewall, che rappresentano ormai una delle più importanti Funzioni di Sicurezza di Rete, dal momento che al giorno d'oggi ci sono tantissime web application da proteggere e controllare.

In un secondo momento è stata modellata la soluzione secondo i risultati ottenuti dalla prima fase di analisi, scegliendo al forma dei requisiti esprimibili e dei parametri configurabili per un singolo Web Application Firewall.

Infine è stato modificato e migliorato il meccanismo che in automatico, preso in input un Service Graph, o un Allocation Graph, e dei requisiti di rete per proteggere le web application come desiderato dall'amministratore di rete, configura i Web Application Firewall all'interno della rete

Dopo che tutti questi lavori sono stati conclusi, sono stati eseguiti una serie di test di prestazione, questo è stato fatto sia per comprendere le differenze in termini di tempo di calcolo tra le possibili condizioni di lavoro sia per studiare la scalabilità. Su questi aspetti, il risultato raggiunto è che sia per il tempo di calcolo sia per la scalabilità della soluzione ci possono essere ottimizzazioni che possono essere introdotte per migliorari i risultati ottenuti fino ad oggi.

8.2 Lavori futuri

I lavori futuri che possono essere svolti per migliorare quanto fatto fino ad oggi sono, come presentato nei capitoli precedenti, realtivi alle limitazioni che sono state trovate e/o introdotte dalla nuova modellizzazione. Il primo lavoro necessario per migliorare le prestazioni è rimuovere una limitazione importante: supportare il carattere jolly nei campi di tipo stringa, questo ridurrebbe il carico sulla configurazione, evitando la creazione di regole duplicate, e migliorerebbe di conseguenze la tempistiche di calcolo.

Bibliografia

[1] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza and J. Yusupov, "Automated firewall configuration in virtual networks," in IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2022.3160293.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[3] D. Bringhenti, G. Marchetto, R. Sisto, S. Spinoso, F. Valenza and J. Yusupov, "Improving the Formal Verification of Reachability Policies in Virtualized Networks," in IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 713-728, March 2021, doi: 10.1109/TNSM.2020.3045781.

[4] D. Bringhenti, G. Marchetto, R. Sisto and F. Valenza, "A novel approach for security function graph configuration and deployment," 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), 2021, pp. 457-463, doi: 10.1109/NetSoft51509.2021.9492654.

[5] Fulvio Valenza, Serena Spinoso, Riccardo Sisto, Formally specifying and checking policies and anomalies in service function chaining, Journal of Network and Computer Applications, Volume 146, 2019, 102419, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2019.102419.

[6] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza and J. Yusupov, "Introducing programmability and automation in the synthesis of virtual firewall rules," 2020 6th IEEE Conference on Network Softwarization (NetSoft), 2020, pp. 473-478, doi: 10.1109/NetSoft48620.2020.9165434.

[7] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza and J. Yusupov, "Towards a fully automated and optimized network security functions orchestration," 2019 4th International Conference on Computing, Communications and Security (ICCCS), 2019, pp. 1-7, doi: 10.1109/CCCS.2019.8888130.

[8] C. Basile, F. Valenza, A. Lioy, D. R. Lopez and A. Pastor Perales, "Adding Support for Automatic Enforcement of Security Policies in NFV Networks," in IEEE/ACM Transactions on Networking, vol. 27, no. 2, pp. 707-720, April 2019, doi: 10.1109/TNET.2019.2895278.

[9] Daniele Bringhenti, Jalolliddin Yusupov, Alejandro Molina Zarca, Fulvio Valenza, Riccardo Sisto, Jorge Bernal Bernabe, Antonio Skarmeta, Automatic,

verifiable and optimized policy-based security enforcement for SDN-aware IoT networks, Computer Networks, Volume 213, 2022, 109123, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2022.109123.

[10] Daniele Bringhenti, Fulvio Valenza, Optimizing distributed firewall reconfiguration transients, Computer Networks, Volume 215, 2022, 109183, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2022.109183.

[11] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza and J. Yusupov, "Automated optimal firewall orchestration and configuration in virtualized networks," NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1-7, doi: 10.1109/NOMS47738.2020.9110402.

[12] Web Application Security Consortium, "Web Application Firewall Evaluation Criteria", Gennaio 2006, http://projects.webappsec.org/f/wasc-wafec-v1.0.pdf

[13] OWASP Top Ten Project, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[14] SQL Injection, https://www.owasp.org/index.php/SQL_Injection

[15] Cross-site Scripting, https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[16] ModSecurity, https://github.com/SpiderLabs/ModSecurity

[17] Vulture, https://www.vultureproject.org/

[18] IronBee, https://github.com/ironbee/ironbee/wiki/

[19] Squid, http://www.squid-cache.org/

[20] Naxsi, https://github.com/nbs-system/naxsi/wiki