Eina de control d'accés a la xarxa

David Lopez Perea

Resum– L'eina de control d'accés a la xarxa que es proposa en aquest article, sorgeix de la mancança d'una plataforma de gestió pels tallafocs d'una organització. L'eina en qüestió pretén facilitar la gestió treballant d'una manera organitzada, com és agrupant les regles per grups i podent assignar aquests grups als servidors. D'aquesta manera, es té una plataforma on poder veure quines regles estan donades per cada màquina de l'organització. Aquesta eina utilitza una plataforma d'automatització per comunicar-se amb els servidors i poder així, introduir les regles adients a cada màquina. Fent-la així, una eina innovadora al món de la gestió remota dels tallafocs.

Paraules clau– Tallafocs, Iptables, Grups de Seguretat, Automatització, Saltstack, Django, Interfície, Base de dades

Abstract– The network access control tool proposed in this article, appears from the lack of a management platform for an organization's firewall. The tool in matter aims to facilitate management working in a organized way, like for instance putting together rules for groups and assign these groups to the servers. This way, there is a platform where you can see which rules are given for each organization machine. This tool uses an automation platform to communicate with the servers and to be able to enter the suitable rules to each machine, thus making it an innovative tool in remote management of firewall.

Keywords– Firewall, Iptables, Security Groups, Automation, Saltstack, Django, Interface, Database

1 Introducció

AQUEST projecte neix de la necessitat de gestionar les regles Iptables de l'empresa Blueliv. Aquesta empresa, ubicada al sector de la ciberseguretat, disposa d'un software al núvol dissenyat per a protegir a empreses de possibles fraus de targetes de crèdit, robatori de credencials i altres amenaces cibernètiques. L'empresa disposa d'un gran volum de servidors en els quals hi ha instal·lat majoritàriament el Sistema Operatiu Debian. Cada servidor té el seu propi script amb el llistat de regles respectives, i és aquí on es troba el **problema**. En aquests scripts es disposa d'un gran nombre de regles, i per tant, hi ha cert descontrol, sense saber quines regles hi ha actives per cada servidor, cosa que implica haver de mirar-ho manualment. La **solució** proposada en aquest projecte, és la de realitzar una interfície d'usuari via web des d'on poder gestionar els diferents Iptables dels servidors. A més, aquesta interfície utilitza una eina d'automatització per transferir les regles a les diferents màquines, realitzant així una solució innovadora i adaptada a les necessitats específiques de l'empresa.

El firewall en el que està centrat aquest projecte ve integrat en el kernel de Linux i s'ocupa de controlar tot el tràfic que entra i surt de les màquines [3]. Com a molts firewalls, Iptables funciona a través de regles, les regles del firewall s'agrupen en cadenes i aquestes cadenes es divideixen en taules. Aquest projecte se centra en la taula filter, que és la taula per defecte de Iptables. La taula en qüestió conté tres cadenes integrades: la cadena que s'ocupa dels paquets destinats a la màquina local (INPUT), la cadena que s'ocupa dels paquets generats per la màquina (OUTPUT) i la cadena que s'ocupa dels paquets que passen per la màquina (FORWARD).

Així doncs, en aquest article s'explicarà la solució proposada i s'entrarà en detall en les diferents tecnologies utilitzades per realitzar la interfície d'usuari. També es veurà com està estructurada la Base de Dades que s'ha utilitzat per realitzar el projecte i es parlarà d'una de les eines que més identifiquen el projecte com és Saltstack, i el perquè de la seva utilització.

En aquest article, en primer lloc es definiran els objectius i requisitis que té el projecte, seguidament, es veuran treballs relacionats amb aquest en l'apartat de l'estat de l'art, a continuació, a l'apartat de proposta s'explicaran les dife-

• E-mail de contacte: davidpsv17@gmail.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Ian Blanes (DEIC)
• Curs 2015/16

rents eines utilitzades per realitzar el projecte. A l'apartat de metodologia i desenvolupament es veurà com ha estat desenvolupada l'aplicació i finalment s'explicaran els resultats obtinguts i es farà una valoració del projecte a l'apartat de conclusió.

2 Objectius i Requisits

El projecte conté uns objectius clars i que han estat definits a l'inici, per poder assolir els resultats esperats. A més a més dels objectius també s'han definit els requisits necessaris per al correcte funcionament de l'aplicació.

2.1 Objectius

Seguidament es llisten els objectius principals del projecte, els quals s'han realitzat de forma seqüencial:

- Dissenyar i implementar un sistema de Base de Dades on s'emmagatzemaran les dades de l'aplicació. Aquest és un dels objectius principals i crítics del projecte, és imprescindible tindre una base de dades que sigui consistent i que s'adapti a les necessitats proposades, per tal de tenir una aplicació adient.
- Implantar i acoblar amb la interfície un sistema que ens permeti agafar les regles i introduir-les als servidors. Aquest objectiu també és crític i té la necessitat de ser assolit dins de la duració del projecte.
- Desenvolupar una interfície web des d'on es podran gestionar les regles i els servidors de l'empresa. Aquest probablement sigui l'objectiu menys important dels proposats, malgrat això, és necessari el seu desenvolupament. L'empresa manca d'una plataforma on poder tindre un control de les regles dels servidors i poder executar remotament aquestes regles. Els dos objectius anteriors ja són suficients per donar d'alta regles i executar-les als servidors, però tindre una interfície ajuda a que l'eina sigui més amigable i dinàmica.

2.2 Requisits

En l'inici del projecte es van concertar diferents reunions amb el client (Blueliv) per definir els requisits relacionats amb l'aplicació, llista que s'anomena a continuació:

- La interfície web ha de tindre un accés restringit per usuaris donats d'alta al sistema.
- S'ha de poder assignar rols als usuaris, perquè depenent el rol tinguin certs privilegis dintre de l'aplicació.
- Disposar a la interfície la possibilitat d'aplicar els canvis realitzats, entrant les regles que pertoca per a cada servidor.
- La interfície ha de ser amigable.
- Utilitzar Saltstack com a eina d'automatització, ja que l'empresa ja té aquest sistema integrat a les màquines.
- Des de Saltstack s'ha de poder extreure les regles de la Base de Dades.
- La base de dades ha de ser compatible amb el framework que s'utilitzi per realitzar la interfície i amb l'eina d'automatització.

3 Estat de l'art

Actualment hi ha una gran quantitat d'aplicacions que treballen amb el firewall Iptables, però la gran majoria són interfícies gràfiques per poder gestionar les regles de la màquina on està instal·lada l'aplicació. Això facilita la modificació de les regles localment a cada màquina, però no soluciona el problema proposat.
També existeixen aplicacions que gestionen les regles de les màquines remotament, una de les més utilitzades és "Firewall Builder" [1]. Aquesta aplicació permet donar d'alta diferents màquines i assignar regles d'entrada i sortida en el Iptables. Per comunicar-se i transferir les regles a les màquines, crea un script per cada una, que conté les comandes adients per afegir les regles al Iptables. Aquest script és passat a cada màquina via SSH per finalment ser executat. En aquest mecanisme de passar les regles als servidors és on l'aplicació proposada millora notablement una eina com FirewallBuilder.
L'aplicació que es proposa utilitza una plataforma d'automatització anomenada Satstack molt més potent que crear un script per executar-lo remotament. Saltsack aporta diversos mòduls dels quals beneficiar-se i la possibilitat d'interactuar amb múltiples màquines directament. Per aconseguir la interacció de múltiples màquines a la vegada Saltstack fa servir ZeroMQ, que és una llibreria de missatgeria que proporciona sockets per transportar missatges a través del protocol TCP [8]. A SSH quan un servidor necessita enviar missatges als clients, obre directament una connexió explícita per cada un, a diferència d'això, a ZeroMQ molts clients poden subscriure's al servidor central i així poder veure els missatges que li pertoquen.
En l'arquitectura de comunicació del projecte tots els clients estan subscrits al servidor central, quan el servidor central disposa de noves regles a entrar, fa un broadcast a tots els clients. Cada client rep el missatge, i si veu que el missatge està assignat a ell entrarà les regles que li pertoca i respondrà al servidor central amb el resultat. Saltstack utilitza dues cues de missatges, el port 4505 del servidor central on els clients se subscriuen per rebre els missatges i el port 4506 és la cua on s'envien els resultats per part dels clients. D'aquesta forma s'aconsegueix una connexió lleugera i persistent, que elimina la sobrecàrrega del servidor [6].

A part de les eines vistes, hi ha plataformes de cloud com Amazon EC2 [12] i Microsoft Azure [11] que tenen un sistema semblant al proposat. L'únic problema és que has de tindre totes les màquines donades d'alta en el seu sistema, i en el cas de l'empresa on s'està realitzant el projecte no es així. També basat en el cloud tenim una plataforma OpenSource com és OpenStack [4], que té un sistema de grups de regles i també utilitza una eina d'automatització com és Heat per introduir les regles a les màquines. El problema és que per utilitzar Openstack es necessita que totes les màquines estiguin a la mateixa LAN o tindre un networking privat.

4 PROPOSTA

La proposta d'aplicació del projecte és la de realitzar una interfície web accessible des de tota l'empresa, des d'on es puguin gestionar els firewalls dels servidors. En l'aplicació que es proposa les regles depenen d'un Security Group, que és un grup de regles d'entrada i sortida, que seran assignats als servidors. Quan s'assigni un Security Group a un servidor, el servidor passarà a tindre totes les regles especificades d'aquest grup.

4.1 Security Group

Un Security Group ajuda a organitzar les regles introduïdes als servidors, i a controlar el tràfic de les instàncies donades d'alta al sistema. A les màquines donades d'alta a l'aplicació, hi ha la possibilitat d'assignar-li un o varis Security Groups. Cada Security Group té un llistat de regles d'entrada i sortida que poden ser modificades en qualsevol moment.
Les regles del Security Group tindran els següents camps:

- *Tipus*: Tipus de regla creada, per exemple "SSH".
- *Descripció*: Una petita descripció de la regla creada. Aquest camp no és obligatori.
- *Protocol*: El protocol a permetre , com podria ser TCP, UDP o ICMP.
- *Port mínim*: El port mínim que tindrà la regla.
- *Port màxim*: Port màxim de la regla, per si es vol entrar un rang de ports.
- *Origen*: Destinació de la regla, aquest camp pot tindre tres valors possibles: IP, IP Group o el mateix Security Group on s'està donant d'alta la regla. En el cas d'aquest valor, es crea una regla per totes les IP's que tenen assignat aquest Security Group.
- *Outbond/Inbound*: Opció per marcar si la regla serà d'entrada o sortida.

Tal com s'ha comentat a la introducció, a cada servidor es treballarà sobre la taula filter de Iptables, on les cadenes INPUT i OUTPUT tindran per defecte la política de denegar, i les regles que es vagin afegint seran les encarregades de permetre o acceptar el tràfic. Cal recalcar que només s'actua sobre les cadenes INPUT i OUTPUT de Iptables, la cadena de FORWARD està configurada perquè accepti el tràfic per defecte.

4.1.1 Per què una política que tingui per defecte denegar el tràfic?

Tindre un sistema on les polítiques siguin per defecte acceptar tot el tràfic, implica que s'hauran d'entrar regles explícites per cada un dels serveis que es vulguin limitar. Això equival a tindre una "black list", on s'ha d'anticiparse per bloquejar el tràfic no desitjat. D'aquesta forma, fàcilment poden aparèixer nous errors.
L'alternativa (i la que s'utilitza en el projecte) és per defecte eliminar tot el tràfic. Això garanteix severament la seguretat del servidor, ja que es té constància del tràfic que entra i surt d'aquest. Així doncs, tot el tràfic que no sigui explícitament indicat a les regles, serà eliminat.

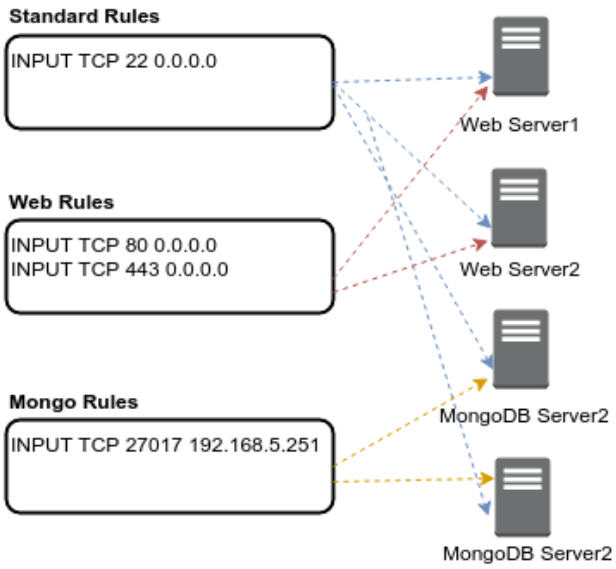


Fig. 1: Arquitectura dels Security Groups.

4.2 Beneficis de treballar amb Security Groups

La finalitat d'un firewall convencional i un que treballi amb Security Groups és la mateixa, controlar el tràfic que flueix a la xarxa i restringir les comunicacions. La principal diferència és que treballar amb un firewall amb Security Groups és més fàcil a l'hora de gestionar totes les regles i permet tindre una visió més organitzada de tot el sistema. A continuació, es mostra un cas pràctic imaginant l'estructura d'exemple de la Figura 1.

Si es treballés amb un sistema sense Security Groups, s'hauria d'afegir les regles estàndards per cada servidor i les de MongoDB als servidors de Bases de dades Mongo i les de Web als servidors web. Quan hi hagués nous servidors s'haurien de tornar a entrar les regles per cada servidor nou, depenent de la seva funcionalitat.
Treballant amb els Security Groups, només s'hauria de configurar cada grup de regles del qual es vulgui disposar (en el cas de l'exemple hi ha les estàndard, Mongo i Web), i relacionar cada màquina amb els Security Groups que es necessités. En l'exemple exposat, es relacionarien tots els servidors amb el grup de regles estàndards. D'altra banda, els de web es relacionarien amb el Security Group de Web i els de bases de dades Mongo amb el Security Group de Mongo. Relacionant-ho d'aquesta manera cada servidor tindria les regles que li pertoquen, i en el cas que hi hagués servidors nous només s'hauria de relacionar amb aquells Security Groups que li pertoquessin, fent així una arquitectura forçament escalable.

4.3 L'aplicació

L'aplicació fonamentalment està dividida en tres parts molt significatives. Hi ha la interfície des d'on l'usuari interactuarà amb l'aplicatiu, també hi ha una base de dades on es guardaran totes les dades contingudes i ,finalment, l'eina d'automatització per extreure les regles de la base de dades i introduir-les a la seva destinació. En aquesta secció es detallarà les eines que s'han utilitzat per realitzar cada un dels mòduls de l'aplicació.

4.3.1 Interfície

La interfície web d'usuari ha estat programada amb Django, que és un Framework web basat en Python, que fomenta el desenvolupament ràpid i el disseny net i pragmàtic [14]. Python és un llenguatge senzill i que permet realitzar una aplicació amb un codi fàcil d'entendre. Un dels avantatges de Django és, proporcionar de serie una interfície d'administració. Des d'aquesta interfície d'administració es poden gestionar els usuaris i grups de l'aplicació, i també dóna l'opció de gestionar les dades que hi hagi creades a la Base de Dades.
En la interfície d'usuari es podran gestionar aquests diferents objectes:

- *Màquines*: Servidors finals on van aplicades les regles introduïdes a l'aplicació. L'aplicació permet donar d'alta i modificar la configuració d'una màquina donada d'alta en qualsevol moment. Quan es dongui d'alta o es modifiqui una màquina, hi ha la possibilitat d'assignar-li un o més d'un Security Group.
- *Security Groups*: Tal com s'ha vist en el punt anterior, son grups de regles que són assignats a les màquines.
- *Ip Groups*: Són grups de IP's. Aquest objecte és útil, perquè en diverses ocasions hi ha moltes IP's que pertanyen a un mateix grup o a les quals es vol assignar una mateixa regla. Per no haver de crear una regla per cada IP, s'afegeixen les IP's als grups i després, al crear la regla, s'introdueix com a camp destinació el grup creat.

Dintre de la interfície també es podran aplicar els canvis realitzats de les següents maneres:

- A la pàgina de l'índex es poden aplicar els canvis màquina per màquina.
- Quan es modifiqui un Security Group hi ha l'opció d'aplicar els canvis realitzats. Marcant aquesta opció s'aplicaran els canvis per totes les màquines que tinguin aquest Security Group assignat.

La interfície també disposa d'un accés restringit per usuaris donats d'alta al sistema. Els usuaris es donen d'alta des de la interfície d'administració de Django i hi ha la possibilitat d'assignar un rol a cada usuari. Depenent del rol que tingui, l'usuari disposarà de certs privilegis a l'aplicació.

4.3.2 Base de dades

Per guardar totes les dades de l'aplicació s'utilitza una base de dades relacional com Mysql. Pel projecte proposat és convenient utilitzar una bases de dades d'aquestes característiques, ja que es necessita integritat de les dades i es tenen unes entitats que estan relacionades entre elles. Un altre dels punts positius pel qual s'ha decidit per Mysql, és que és compatible tant per l'eina que s'utilitza per la interfície web, com per l'eina que s'utilitza per extreure les regles i introduir-les als servidors.

4.3.3 Eina per entrar les regles a les màquines

Per entrar les regles a les màquines es necessita una eina que s'adapti a les necessitats de l'empresa i que sigui el més escalable possible. S'ha triat Saltsack, ja que l'empresa ja treballa amb aquesta eina i ja està desplegada. D'aquesta manera, no hi haurà cap cost d'instal·lació extra dins de l'organització. Saltstack és una eina d'automatització que permet executar comandes remotament, això facilita la gestió de la infraestructura a l'organització. En el punt d'estat de l'art s'ha vist els beneficis que pot aportar respecte a sistemes que utilitzen altres mètodes i en el següent punt es podrà veure com està configurat en el sistema.

5 METODOLOGIA I DESENVOLUPAMENT

La metodologia utilitzada per realitzar el projecte ha estat la iterativa. El model iteratiu és un model derivat del model en cascada, on hi ha una sèrie de fases consecutives ben definides, per poder avançar a una fase s'ha d'haver superat completament la fase anterior. Com a suplement del model en cascada, itera en diversos cicles de vida sobre aquest. Les iteracions s'han repetit fins a acabar el projecte. Un dels avantatges que proporciona aquesta tecnologia és que no cal tindre uns requisits totalment definits a l'inici del projecte, sinó que es poden anar refinant els requisits amb les iteracions. Aquest projecte ha tingut uns requisits ben definits, però fàcilment adaptables a canvis.

Al final de cada iteració el client ha proposat millores o la possibilitat d'afegir una nova funcionalitat al projecte. Cal especificar que les iteracions han sigut realitzades incrementalment, no s'ha esperat a tindre tot el projecte finalitzat per tornar a iterar sobre aquest. Si s'hagués fet així, es podria haver perdut el control del projecte tal com especifica Alistair Cockburn en aquest article [2].

Per realitzar el projecte s'ha dividit el treball en diferents fases o etapes, per aquestes etapes s'han establert uns temps de dedicació. En alguna etapa de la planificació s'ha hagut d'adaptar el temps per complicacions, fet que ha estat possible gràcies a la metodologia utilitzada.
Seguidament, es veuran detalladament les etapes realitzades per aconseguir l'aplicació en el temps establert.

5.1 Captura de Requisits

Fase on es capturen totes les dades necessàries per poder realitzar el projecte. Tal com s'ha especificat anteriorment es van realitzar diverses reunions amb el client per detallar els requisits que hauria de tindre l'aplicació.

5.2 Disseny ER de la Base de Dades

El model Entitat-Relació és un model de dades basat en una percepció del món real [13]. En el model hi ha un conjunt d'objectes bàsics anomenats entitats i les relacions entre aquestes. En el projecte realitzat s'han definit les següents entitats amb els seus respectius atributs: Source, Machine, SecurityGroup, IP, IPGroup, Rule i typeRule.

En l'entitat Machine es guardaran totes les dades que referencien als servidors, com per exemple el nom de la màquina, IP interna, IP externa, etc. Entre Machine i Security Group s'ha establert una relació $n - n$, ja que una màquina pot tindre més d'un Security Group associat i un Security Group pot pertànyer a més d'una màquina. Amb aquesta relació entre Machine i SecurityGroup s'ha hagut de crear una nova entitat intermèdia entre les dues taules.

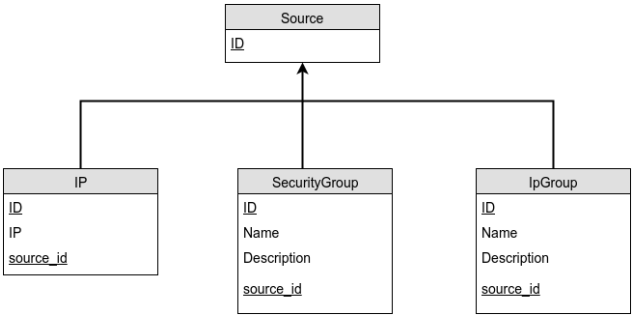


Fig. 2: Disseny de Source per la Base de dades.

L'entitat Rule guarda tots els camps de les regles. Com s'ha especificat anteriorment una regla s'ha d'afegir dins d'un Security Group, això implica que no es pot crear una regla sense abans haver creat un Security Group. La relació entre Regla i SecurityGroup és una $1-n$, ja que una regla només pot estar donada d'alta en un Security Group, però un Security Group pot tindre més d'una regla.
Per guardar els grups de IP's que es poden assignar a les regles hi ha l'entitat IpGroup, aquesta entitat està relacionada amb IP amb una $n-n$, ja que es pot tindre més d'una IP associada a un IpGroup i un IpGroup tindre associada més d'una IP.
Tal com s'ha especificat, el camp "Source" de la regla pot tindre tres possibles valors: una IP, un IpGroup o el mateix Security Group on s'està donant d'alta la regla. Per solucionar aquest problema, primer es va pensar de posar tres atributs a la regla: un pel camp sourceIP, un altre sourceIpGroup i finalment el sourceSecurityGroup. La idea era que quan es donés d'alta una regla només un d'aquests tres camps fos vàlid. Per exemple, si s'entra una regla amb la IP 8.8.8.8, els camps sourceIpGroup i sourceSecurityGroup serian $NULL$ a la base de dades.
Finalment, s'ha optat per una solució més òptima com és tindre un camp Source i que IP, IpGroup i SecurityGroup estiguin relacionats amb Source en una relació $1-n$. D'aquesta manera, les entitats IP, IpGroup i Security Group tenen la ID de Source com atribut dins de la taula. Així, en l'entitat Rule només tenim un camp Source, que és la ID de source de l'entitat IP, IpGroup o SecurityGroup. El disseny de la solució del problema del camp source es mostra a la figura 2 i el disseny complet del diagrama es pot veure a l'apèndix.
En aquesta fase s'ha dedicat més temps del previst inicialment, obligant a modificar la duració d'altres fases. Dissenyar una base de dades que sigui consistent i que s'adapti a les necessitats del projecte ha estat més complicat de l'esperat. Però haver hagut d'allargar la durada d'aquesta fase no ha estat cap problema per poder acabar el projecte a temps. Gràcies a la metodologia exposada anteriorment s'ha pogut anar adaptant i canviant la base de dades fins que s'ha obtingut una base de dades consistent i que s'adapta als requisits del projecte.

5.3 Configuració interfície d'administració

Tindre un espai on els administradors puguin gestionar el contingut de la web, normalment és un requeriment essencial. Però realitzar aquestes interfícies d'administració és una tasca repetitiva i feixuga. Django soluciona aquest problema facilitant una interfície on només poden entrar usuaris registrats per poder administrar tots els continguts de la web [5].
Des d'aquesta interfície es poden veure totes les entitats que hi ha introduïdes a la base de dades. En el cas de l'aplicació, es pot veure les entitats com Machine, Security Group, IpGroup, etc.
Gràcies a la interfície d'administració de Django s'ha pogut començar a realitzar les primeres proves de l'aplicació, com donar d'alta noves màquines i Security Groups, i relacionarlos entre ells. També, s'ha pogut provar físicament la base de dades dissenyada anteriorment.

5.4 Instal·lació i configuració de Saltstack

Un cop la base de dades ha estat dissenyada i implementada, i també s'ha tingut l'administració de Django, s'ha d'instal·lar la plataforma d'automatització de Saltstack, que ens permetrà introduir les regles en els servidors. Saltstack és el connector entre la interfície d'usuari i les màquines, constituint així, una de les parts importants d'aquest projecte.
Saltstack treballa amb dues peces diferents:

- *Master*: És l'eix central, ja que executa les comandes i aplica les configuracions a sobre els Minions. En l'aplicació proposada, el Master està instal·lat a la mateixa màquina on està instal·lada la interfície d'usuari de Django.

- *Minion*: Connecten amb el Master, reben i executen les seves ordres. Els Minions a l'aplicació són els servidors on es vol introduir les regles.

Hi ha dues maneres de comunicar-se amb els Minions des del Master. La primera, consisteix en instal·lar en les màquines el mòdul de Minion. Quan s'instal·la aquest mòdul es configura el Minion especificant la IP del Master i una ID que servirà per identificar el Minion quan es vulgui executar alguna comanda sobre aquest. Aquesta forma utilitza la llibreria ZeroMQ.
La segona forma de comunicar-se el Master amb els Minions és a través d'un mòdul que implementa SSH. Per aquest mòdul no cal tindre cap element de salt instal·lat en els Minions, ja que el Master es comunica amb els Minions a través d'un túnel SSH per executar les seves comandes. Per tal de què el Master es pugui comunicar amb els Minions, han d'estar afegides les IP's dels Minions en un fitxer de configuració, en aquest fitxer també hi ha d'haver l'enllaç on es troba la clau privada per poder accedir al Minion. Els Minions també han de contenir la clau pública del Master perquè aquest es pugui connectar a ells.
Actualment, l'aplicació està configurada per treballar amb el mòdul que implementa SSH, ja que els servidors de l'empresa treballen tots amb aquesta metodologia. La intenció és migrar-ho tot en un futur no molt llunyà a la metodologia master-minion tenint l'aplicació salt-minion instal·lada als servidors. D'aquesta manera, es millora la comunicació entre els masters i els minions. Cal especificar però, que l'aplicació està preparada i provada perquè pugui funcionar amb les dues metodologies.
Saltstack proveeix una interfície anomenada Pillar on es poden emmagatzemar variables estàtiques per després ser

utilitzades. Per la base de dades s'utilitza un mòdul que s'anomena external-pillar, que extreu les dades d'un recurs extern. En aquest cas, el recurs utilitzat ha estat Mysql, d'aquesta manera es podran fer consultes a la base de dades i extreure les regles. Saltstack també proporciona una interfície que s'anomena Grain, de la qual es poden extreure dades de les característiques dels Minions (ex: Sistema Operatiu, IP, etc.). En aquesta fase s'ha configurat el mòdul external-pillar de Mysql, en el següent punt s'explica més detalladament el seu funcionament.

5.5 Extreure regles amb Saltstack i introduir-les als servidors

Abans d'explicar com s'utilitza Saltstack per extreure les regles, cal entendre com funcionarà el sistema de regles de Iptables dels servidors. Realitzant les primeres proves amb els Iptables es va identificar un problema de persistència, quan es reiniciava els servidors es perdien totes les regles ingressades en el Firewall. Per resoldre aquest problema s'ha utilitzat el paquet iptables-persistent. Aquesta eina treballa amb les opcions de Iptables: iptables-save i iptables-restore. La primera opció guarda totes les regles de Iptables actuals en un fitxer i amb iptables-restore es pot restaurar la configuració del Iptables passant per paràmetre un fitxer de regles. Així doncs, iptables-persistent quan s'atura la màquina realitza un iptables-save en un fitxer del servidor, i al iniciar aquest servidor fa un iptables-restore del fitxer guardat prèviament.
Per gestionar els IpGroups de la interfície s'utilitza l'eina Ipset, que permet especificar un grup i afegir adreces en aquest grup [9]. Amb els grups creats al Ipset es poden definir regles a Iptables, especificant en el camp *source* o *destination* de la regla un dels grups especificat a Ipset, d'aquesta manera, s'estalvia haver de crear una regla per cada IP del grup.
Un dels problemes trobats amb els Ipset i iptables-persistent, és que a l'hora de reiniciar els servidors s'iniciava primer el servei iptables i no carregava Ipset. D'aquesta manera, quan Iptables volia carregar les regles que contenien un Ipset donava un error i no s'aixecava el servei, ja que amb Ipset passa el mateix que amb Iptables, al reiniciar el servidor es perd tota la configuració. Per resoldre aquest problema s'ha hagut de modificar el script d'inici de iptables-persistent, afegint una nova funció que s'executi abans que la de carregar les regles de Iptables. Aquesta funció anomenada $load\_ipset()$ carrega tots els Ipsets guardats en un fitxer i d'aquesta manera, en executar la funció de càrrega de les regles de Iptables, es disposen dels grups de IP's carregats a l'eina Ipset. Per carregar tots els Ipset s'utilitza l'opció de restaurar que també proporciona ipset: ipset restore.
En aquest codi es veu un petit exemple de com funciona el script d'inici de iptables-persistent:

```
load_ipset(){
  ipset destroy
  ipset restore < /etc/iptables/ipset
}
load_rules(){
  iptables-restore < /etc/iptables/rules.v4
}
save_rules(){
  iptables-save > /etc/iptables/rules.v4
}
flush_rules(){
  /sbin/iptables -F
}
```

Una vegada s'ha vist la configuració de Iptables del servidor, es passa a veure com són extretes les regles i introduïdes als servidors. Per realitzar-ho s'utilitza el "salt system" que és un component de Saltstack fet per la gestió de la configuració. Els fitxers states tenen com extensió ".sls", i en aquests fitxers s'indiquen totes les configuracions que es volen realitzar.
En aquest exemple es pot observar el codi que hi hauria en un fitxer d'estat de Saltstack per instal·lar el paquet Iptables (tot i que en un fitxer d'estat hi poden haver més d'una instal·lació o configuració):

```
iptables:
  pkg.installed
```

Saltstack permet utilitzar el State System de forma dinàmica. Això ho realitza a través de templates, per defecte els fitxers d'estat utilitzen Jinja. Gràcies al template també es pot interactuar amb els Pillar i els Grains. Aquí es pot veure un exemple d'un fitxer SLS dinàmic utilitzant aquest template (en aquest exemple s'extreuen dades dels Grains):

```
{% if grains["os_family"] == "Debian" %}
  apache2:
{% elif grains["os_family"] == "CentOS" %}
  nginx:
{% endif %}
    pkg:
      - installed
    service:
      - running
```

Saltstack també dóna la possibilitat de treballar amb "environments", en aquests, s'emmagatzemen els fitxers d'estat. Els "environments" són executats contra les màquines. Per exemple, es pot tindre un "environment" base, que en executar-lo contra els servidors els posi en un estat bàsic. Per executar aquest "environment" sobre una màquina, es posaria la següent sentència a la línia de comandes:

```
salt "Machine7" state.sls saltenv=base
```

Un cop s'ha vist com treballa Saltstack, es pot veure com s'ha configurat per poder extreure les regles en l'aplicació proposada. S'ha creat un nou "environment" anomenat *nacl*, sobre aquest "environment" pengen dues carpetes.
La carpeta $setup$ s'utilitza per "preparar" la màquina, per poder utilitzar-la en l'aplicació. En aquesta carpeta es troba un fitxer de Saltstack que primer et comprova que estigui el paquet iptables-persistent instal·lat i seguidament, crea un fitxer que contindrà les regles per defecte del Iptables. En aquestes regles s'especifica que INPUT i OUTPUT per defecte estan a DROP, també s'especifiquen regles per donar accés al Master (aquestes regles permetran tot el tràfic del protocol TCP provinent i que vagi cap a la IP del Master). Aquest fitxer es guarda a la ruta "/etc/iptables/rules.v4" que és el fitxer que s'ha vist abans que es crida al fer un iptables-restore dins del script d'inici de Iptables. Una vegada creat el fitxer es reinicia el servei de iptables-persistent perquè automàticament entri les regles estàndard a Iptables.
Per executar el setup del environment *nacl* des de la línia de comandes es fa amb aquesta sentència:

```
salt "*" state.sls setup saltenv=nacl
```

La carpeta *update* s'utilitza per carregar les regles introduïdes a la interfície en els servidors destinataris. D'aquesta carpeta pengen tres fitxers:

- *ip.sls*: Aquest fitxer s'encarrega d'entrar totes les regles que tenen com a "source" una IP. Per això utilitza el external pillar de Saltstack, fent una consulta contra la base de dades per extreure les regles relacionades amb la màquina contra la qual s'està executant l'estat, i extraient només les regles que tenen com a "source" una IP. La consulta retorna una llista amb totes les regles, per extreure cada regla s'itera sobre la llista i s'afegeix cada regla al fitxer "etc/iptables/rules.v4". Abans d'extreure i escriure les regles en el fitxer, s'elimina (si existeix) el fitxer "/etc/iptables/rules.v4" de la màquina on s'està realitzant l'execució de l'estat.

- *ipgroup.sls*: Una vegada es tenen totes les regles amb IP's entrades al fitxer, es torna a utilitzar el external pillar de mysql per fer una consulta on el camp source és un IpGroup. Abans d'escriure les regles en el fitxer s'han de crear els ipset. Per fer-ho, es crea un fitxer nou ("/etc/iptables/ipset") i s'afegeixen tots els Ipgroups amb les IP's corresponents. Anteriorment s'ha especificat el contingut del fitxer d'inici de iptables-persistent, on es veu que al carregar els ipset fa un restore del fitxer que s'està creant.

- *securitygroup.sls*: Fitxer que té una funcionalitat semblant a l'anterior. Els Security Groups també són tractats com Ipsets. Es fa una consulta sobre la base de dades i s'afegeixen els Security Groups al fitxer dels ipset. També s'afegeixen les regles al fitxer "etc/iptables/rules.v4" on el camp source és un Security Group. Finalment, es reinicia el servei iptables-persistent. Un cop és reiniciat, es tornen a carregar els Ipsets (carregant els que hi ha al fitxer "/etc/iptables/ipset") i es tornen a carregar les regles (carregant les regles que hi ha al fitxer "/etc/iptables/rules.v4").

Per extreure les regles des dels templates s'utilitza el mòdul external-pillar indicat anteriorment. A l'arxiu de configuració de Saltstack s'indiquen les consultes que es faran a la Base de dades. Seguidament, des dels templates es criden aquestes consultes. Per exemple aquesta és la consulta per extreure totes les regles que tenen com a source una IP, a "m.name" anirà el nom del Minion en el qual s'està executant l'estat (aquesta és una consulta reduïda per temes d'espai, a la real hi ha més camps en el select):

```
query_ip: 'select i.ip
from machine m, nacl n,
rule r, ip i
where n.machine_id=m.id
and r.SG_id=n.SG_id
and m.name=%s
and i.source_id=r.source_id'
```

Per cridar aquesta consulta des del template s'utilitza la següent sentència:

```
{%set query = pillar["query_ip"]%}
```

Aquesta sentència introdueix tots els resultats de la consulta en format llista. Una vegada feta la consulta s'ha d'iterar per la llista per extreure les regles i introduir-les al fitxer.

5.6 Implementació interfície d'usuari

Una vegada s'ha tingut implementada la base de dades i s'extreuen les regles per introduir-les als servidors, s'ha realitzat la interfície web final des d'on es podrà accedir loguejat i gestionar les màquines i Security Groups. En primer lloc, s'han realitzat els formularis d'entrada de dades com Machine, Security Group etc. Per fer la interfície amigable s'ha utilitzat el framework de Bootstrap i també s'ha fet servir la biblioteca de Javascript jQuery per tal de simplificar la manera d'interactuar amb els documents HTML.
Seguidament, s'ha restringit la interfície a usuaris registrats a l'aplicació. Per realitzar-ho s'ha utilitzat un mòdul d'autenticació que proporciona Django *aut/aut* (Autenticació/Autorització) [10], aquest mòdul també s'ha utilitzat per restringir l'accés als usuaris a diferents seccions de l'aplicació depenent del grup al qual estiguin assignats.
Finalment, el repte més important de programar la interfície ha estat la manera de cridar a Saltstack des de Django, per realitzar-ho s'utilitza l'API de Saltstack. A part de l'API de Saltstack hi ha hagut la necessitat d'instal·lar un mòdul de Python per poder tindre una Rest API, el mòdul "CherryPy" [7].
Rest és un estil d'arquitectura de software vinculat al protocol HTTP que ens permet crear serveis webs escalables. Una vegada instal·lat el paquet de "Cherrypy", per configurar la Rest API s'ha d'anar al fitxer de configuració de saltstack i indicar un port i el host on s'executarà l'API. En el cas d'aquest projecte l'API funciona en el mateix servidor que està corrent Saltack, amb el que en host es té indicat "localhost".
Hi ha dues possibles maneres d'utilitzar la API (l'aplicació està preparada per utilitzar les dos). En un primer cas, hi ha l'opció d'utilitzar l'arquitectura de Saltstack amb minion-master, en aquest cas cal autenticar-se contra l'API. Per realitzar-ho es fa una petició amb usuari i contrasenya a la URL http://localhost:8000/login. La resposta de la petició és un token, amb aquest token ja es podrien executar comandes de salt contra l'API.
Una segona opció, és no utilitzar minion-master i utilitzar el mòdul de Saltstack que implementa SSH, en aquest cas no cal autenticar-se contra l'API i es poden executar sentències directament. Per executar les peticions de Salt contra l'API es fa una petició HTTP contra localhost i el port indicat en el fitxer de configuració de Saltstack, en aquest exemple es pot veure una petició via l'eina curl:

```
curl -L localhost:9191/run \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"client":"ssh", "tgt":"Machine1",
"fun":"state.sls",
"arg":["update", "nacl"]}'
```

Quan s'executa una comanda amb Saltstack executant un estat contra una màquina, Saltstack retorna en forma de missatge el resultat de l'execució. Si hi ha algun error, retorna el missatge d'error on indica el que no s'ha pogut realitzar. Si per contra tot anat bé també ho indica al missatge. S'ha indicat que aquest missatge de resposta sigui retornat en format JSON, ja que treballar amb aquest format és molt més senzill. Una vegada retornat el missatge, es mostra per pantalla el resultat perquè l'usuari pugui veure si ha anat tot bé o hi ha hagut algun problema.

Des de Django per interactuar amb la API s'utilitza la llibreria "urllib2", que permet la interacció amb URLs per llegir dades utilitzant el protocol HTTP.

5.7 Testing

S'han realitzat dos tipus de tests a l'aplicació. En primer lloc, s'ha dut a terme testing unitari, on s'ha dividit l'aplicació per mòduls i testejat el correcte funcionament de cada un d'ells. Finalment, també s'ha realitzat un test d'integració de l'aplicació final.
A la interfície s'ha dividit cadascun dels objectes i s'han testejat per separat els seus formularis. Per exemple, en l'apartat d'entrar i modificar una màquina, s'han provat tots els camps d'afegir una màquina i tots els camps d'editar una màquina ja donada d'alta. També s'han testejat les relacions entre els objectes, per exemple la relació entre màquina i Security Group, que quan es dongui una màquina d'alta se li pugui assignar múltiples Security Groups. Una vegada provats tots els mòduls, també s'han realitzat proves per veure que si s'assignaven regles a una màquina, després d'aplicar els canvis, aquestes regles estiguessin afegides al Iptables d'aquesta. Les regles afegides a les màquines s'han testejat per veure que realment estiguin ben creades, realitzant proves pràctiques entre màquines. Finalment, s'ha realitzat un test d'integració amb tota la interfície ja acabada. En aquest test s'ha provat el correcte funcionament d'aquesta i s'han provat tots els elements de l'aplicació interactuant entre ells.

6 Resultats

A l'inici del projecte es van definir uns requisits i objectius que havia de tindre l'eina realitzada. Actualment, es disposa d'una versió final amb alguns petits canvis funcionals, però que compleix tots els requisits proposats per l'empresa. Aquesta aplicació ha estat supervisada pel client i actualment s'està posant en producció per a poder utilitzar-la en l'organització. Els objectius s'han anat solucionant segons el seu grau d'importància, realitzant primer els dos objectius crítics com dissenyar i crear la base de dades, i implementar el sistema d'extracció i introducció de les regles als servidors. Finalment, quan s'han tingut aquests dos objectius realitzats, s'ha programat la interfície final des d'on l'usuari interactuarà amb l'aplicació.

Actualment es disposa d'una base de dades relacional on es guarden totes les dades de l'aplicació i aquesta base de dades està perfectament acoblada a la interfície.
S'ha instal·lat i configurat l'eina d'automatització Saltstack com s'ha explicat en aquest article, aquesta eina funciona perfectament per extreure les regles de forma automatitzada i introduir-la als servidors destinataris.
També es disposa d'una interfície simple i intuïtiva des d'on poder gestionar tots els Security Groups i màquines de l'organització.

Que l'aplicació estigui acabada no significa que no es pugui modificar o millorar en un futur, ja que és una aplicació fàcilment escalable i adaptable a futurs canvis. La intenció és pujar l'aplicació a un repositori públic i ficar-li una llicència GPL, fent així una aplicació Open Source i que la comunitat pugui proposar futuribles millores o adaptar-la a les seves necessitats.

7 Conclusions

L'aplicació proposada en aquest projecte es basa en una arquitectura anomenada Security Groups, que són regles agrupades que poden ser assignades als diferents servidors, facilitant així l'organització i gestió d'aquestes. Gràcies a aquesta eina s'estalvia una tasca repetitiva com haver d'anar entrant regla per regla a cada servidor. Utilitzant aquesta arquitectura, al servidor se li assigna directament un o varis Security Groups, passant a tindre totes les regles que componen els grups assignats. Això es realitza d'aquesta forma, ja que en una organització normalment es disposa de diferents servidors que ofereixen els mateixos recursos, havent de tindre la mateixa configuració al Firewall per cada un.
Les regles són guardades en una Base de Dades Mysql i s'utilitza una aplicació d'automatització per l'extracció d'aquestes, i la posterior introducció a les màquines adients. L'eina d'automatització anomenada Saltstack permet la comunicació amb diferents servidors alhora i la possibilitat d'executar comandes remotament. Utilitzar una eina d'automatització com Saltstack, és principalment una de les grans diferències amb aplicacions semblants en el sector de la gestió dels Iptables, millorant d'aquesta manera la comunicació amb cada una de les màquines remotes.
Com s'ha especificat anteriorment, en el projecte s'ha dut a terme tots els objectius proposats des d'un principi, encara que, hi ha la possibilitat d'estendre el projecte i afegir nous mòduls. Actualment, s'està parlant amb el client de realitzar futurs canvis dins de l'aplicació. Per exemple, una de les possibles extensions que podria tindre aquest projecte en un futur, és tractar regles amb IPv6. Aquesta seria una extensió fàcilment adaptable al sistema i que no implicaria canviar l'estructura d'aquest.
Com també s'ha comentat anteriorment. s'estan adaptant els servidors de l'organització perquè treballin amb l'arquitectura de Saltstack $master - minion$. Un altre dels canvis que s'ha plantejat, és el de canviar la forma de comunicació entre el master i els minions. Actualment, és l'usuari en el servidor $master$ qui executa les regles contra els $minions$, el possible canvi que s'està valorant és que siguin els $minions$ que cada cert temps facin $polling$ al master per actualitzar-se de manera automàtica. Tot i que treballar d'aquesta manera automàtica ens impossibilitaria veure en el moment del canvi si hi ha hagut algun problema, faria el sistema encara més dinàmic.

Agraïments

Vull donar les gràcies a totes les persones que han col·laborat en el projecte i que m'han ajudat amb els seus coneixements. També agrair a Blueliv l'oportunitat de realitzar el projecte amb ells i ajudar-me en la meva formació durant tot aquest temps.

Finalment, agrair a la família i parella tot el suport donat en aquest projecte i durant aquests anys a la universitat.

Referències

[1] Mick Bauer. Using Firewall Builder. `http://www.linuxjournal.com/article/6625`, 2003. Accedit: 08-06-2016.

[2] Alistair Cockburn. Using both incremental and iterative development. *Crosstalk*, 2008.

[3] Herve Eychenne. *Iptables Man Page*, Juny 2015.

[4] Tom Fifield. *OpenStack Operations Guide*, chapter 10, pages 104–106. O'Reilly Media, 2014.

[5] Saul Garcia. *La guia definitiva de Django*, chapter 6, pages 101–104. Django Software Corporation, 2015.

[6] Joseph Hall. *Mastering Saltstack*, chapter 7, pages 157–158. Packt Publishing Ltd., 2015.

[7] Thomas Hatch. *Salt-Api Documentation*. Saltstack, Desembre 2015.

[8] Pieter Hintjens. *ZeroMQ Guide*. iMatix, 2014.

[9] Jozsef Kadlecsik. *Ipset Man Page*, Juny 2015.

[10] Saul Garcia M. *La guia definitiva de Django*, chapter 14, pages 296–305. Django Software Corporation, 2015.

[11] Telmo Sampaio. *¿Qué es un grupo de seguridad de red?* Microsoft, Febrer 2016.

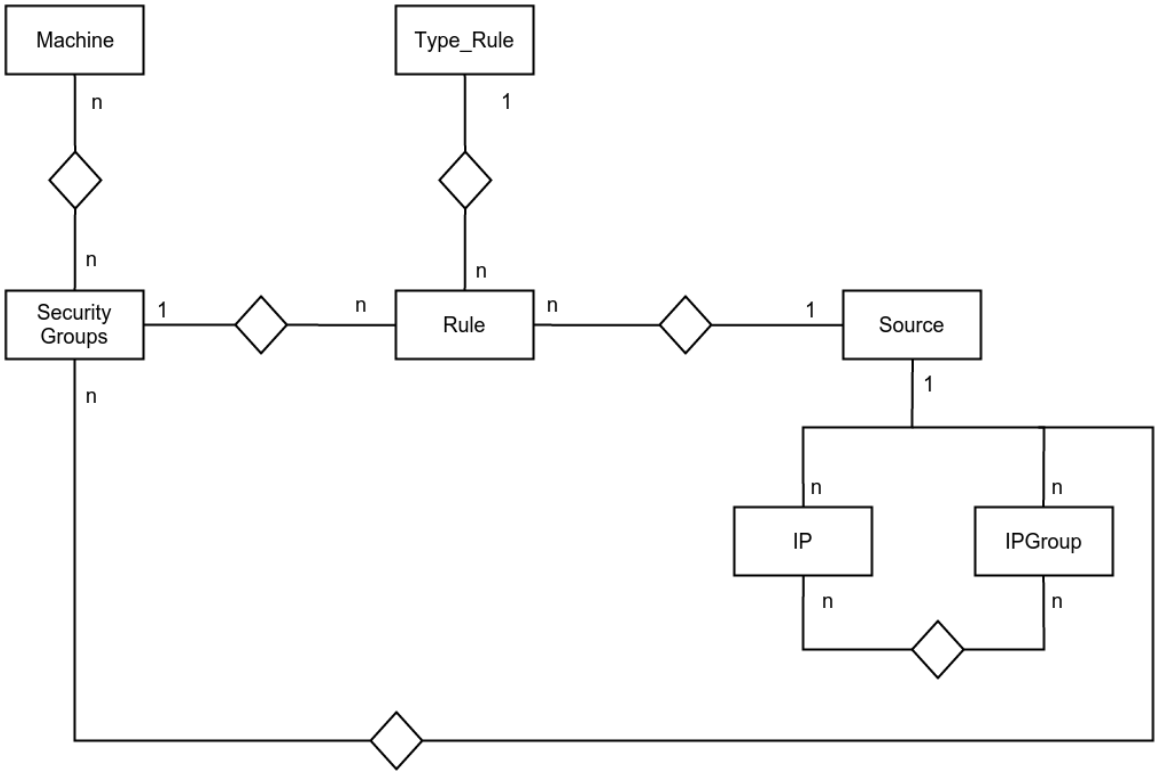
[12] Amazon Web Services. *Amazon EC2 User Guide for Linux Instances*, pages 485–487. 2012.

[13] Wikipedia. Modelo entidad-relación — Wikipedia, La Enciclopedia Libre, 2016. [Accedit: 10-06-2016].

[14] Adrian Holovaty y Jacob Kaplan-Moss. *Django Book*, chapter 1. Apress, 1 edition, 2007.

APÈNDIX

A.1 Diagrama ER de la Base de Dades



Apèndix 1: En aquesta figura es pot observar les diferents taules i relacions que composen el diagrama ER utilitzat en el projecte.