Treball de Fi de Màster

Màster Universitari en Enginyeria Industrial

Anàlisi, experimentació i millora d'un sistema de ciber-seguretat de la llar "Crowdsourced"

MEMÒRIA

Autor: Edgar González Quevedo

Director/s: Pr. Salvador Manich Bou
Pr. Eric Keller

Convocatòria: Juliol 2017

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

Aquest projecte consisteix en l'anàlisi, experimentació i millora de CommunityGuard, un sistema de ciber-seguretat col·laboratiu aplicable a xarxes industrials, empresarials i domèstiques. CommunityGuard consisteix, en essència, en un dispositiu que resideix entre el "router" de la xarxa i el cable mòdem d'enllaç extern, i que es connecta a un sistema en núvol, al qual hi ha connectats més dispositius com aquest. El sistema automàticament escolta i monitoritza els paquets d'informació que accedeixen o surten de la xarxa amb l'objectiu de trobar activitat sospitosa i proporcionar una resposta adequada per tal de rebutjar satisfactòriament els atacs.

Per a fer aquest projecte el primer pas ha estat la definició d'especificacions, com el nombre de nodes guardians o el tràfic maliciós a bloquejar. Després, s'ha analitzat i recreat CommunityGuard en un servidor, per tal d'entendre els aspectes més importants del seu funcionament. Això s'ha fet mitjançant màquines virtuals i respectant les especificacions originals. En aquesta etapa, s'observa que hi ha certs aspectes del codi de CommunityGuard que no estan acabats o que presentaven certs errors, i es corregeixen per acabar d'implementar-lo.

D'altra banda, s'han realitzat proves per comprovar si el software utilitzat per a detectar tràfic maliciós, anomenat Snort, presenta un efecte negatiu en el rendiment de la xarxa. Els resultats de les proves demostren que l'efecte que causa és mínim, i que per tant no hi ha necessitat d'optimització en aquest sentit.

La següent etapa del projecte és una ampliació de CommunityGuard. L'ampliació consisteix en un sistema orientat a la seguretat dels dispositius IoT d'una xarxa. El sistema conserva la figura del node guardià i afegeix la figura d'una persona. Els dispositius IoT de la xarxa han de demanar autorització al node guardià per a realitzar qualsevol acció nova, i la persona s'encarrega de donar resposta manualment a les noves sol·licituds.

Aquest projecte, realitzat a la Universitat de Colorado, Boulder, gràcies a la beca Balsells i sota la direcció del professor Eric Keller, ha assolit satisfactòriament els objectius proposats. S'ha aconseguit que CommunityGuard funcioni correctament, i s'ha implementat una millora del mateix que incrementa la seguretat dels dispositius IoT. A l'annex F s'inclou una carta de valoració escrita pel professor Eric Keller.

Índex

1. Glossari

Adreça IP (de l'anglès *Internet Protocol*): és un identificador assignat a cada computador i altres dispositius connectats a una xarxa TCP/IP que és usat per localitzar i identificar el node en comunicacions amb altres nodes de la xarxa.

Adreça MAC (de l'anglès *Media Access Control*): en xarxes computacionals, és un identificador de 48 bits (6 blocs de 2 caràcters hexadecimals) que correspon de forma única a una targeta o dispositiu de xarxa. Es coneix també com a direcció física, i és única per cada dispositiu.

Ansible: és una plataforma de software lliure per a configurar i administrar computadores. Combina instal·lació multinode, execució de tasques ad hoc i administració de configuracions.

Atlantic.Net: és un proveïdor de serveis Web-hosting amb presència del centre de dades als Estats Units d'Amèrica.

AWS (de l'anglès *Amazon Web Services*): és una plataforma de serveis en núvol que ofereix potència de càlcul, emmagatzematge de bases de dades i entrega de contingut.

CentOS (de l'anglès *Community Enterprise Operating System*): és una distribució de Linux que pretén proporcionar una plataforma computacional gratuïta, orientada a organitzacions i compatible amb la seva font original, Red Hat Enterprise Linux (RHEL).

CIDR (de l'anglès *Classless Inter-Domain Routing*): és un mètode per assignar adreces i rutes IP.

Cisco Anyconnect: és una aplicació per a establir conneció amb una xarxa privada. El client executa accions remotament amb els recursos d'una altra xarxa disponible d'una manera segura, com si l'usuari estigués directament connectat a aquella xarxa "privada".

D-ITG (de l'anglès *Distributed Internet Traffic Generator*): és una plataforma capaç de produir tràfic IPv4 i IPv6 replicant precisament la càrrega de treball d'aplicacions en Internet actuals, a la vegada que permet realitzar mesures en xarxa com la velocitat, el retard dels paquets o la pèrdua de paquets. D-ITG pot generar tràfic seguint models estocàstics de la mida dels paquets o del temps entre paquets.

DDoS (de l'anglès *Distributed Denial of Service*): és un tipus d'atac DoS en el que molts sistemes compromesos, que en moltes ocasions són infectats per un virus Troià, són usats per atacar un únic sistema, causant una denegació de servei. Les víctimes d'aquest atac

són alhora els sistemes control·lats pel hacker en qüestió i el sistema objectiu de l'atac.

Debian: és una comunitat conformada per desenvolupadors i usuaris, que manté un sistema operatiu GNU basat en software lliure.

DNS (de l'anglès *Domain Name System*): és un sistema de nomenclatura jeràrquic descentralitzat per a dispositius connectats a xarxes IP com Internet o una xarxa privada. La seva funció més important és traduir noms intel·ligibles per a les persones en identificadors binaris associats amb els equips connectats a la xarxa, amb el propòsit de poder localitzar i direccionar aquests equips mundialment.

DoS (de l'anglès *Denial of Service*): és un atac a un sistema de computadors o xarxa que causa que un servei o recurs sigui inaccessible a usuaris legítims. Normalment provoca la pèrdua de la connectivitat amb la xarxa pel consum d'ample de banda de la xarxa de la víctima o sobrecàrrega dels recursos computacionals del sistema atacat.

DynDNS (de l'anglès *Dynamic Network Services*): és una companyia d'Internet dels Estats Units d'Amèrica dedicada a solucions de DNS en direccions IP dinàmiques.

Emerging Threats: és una col·lecció de projectes de seguretat, la majoria d'ells relacionats amb la detecció d'intrusions i amb l'anàlisi de tràfic en xarxa.

Ethernet: és un estàndard de xarxes d'àrea local per a computadors amb accés al medi per detecció de l'ona portadora i amb detecció de col·lisions (CSMA/CD). Ethernet defineix les característiques de cablejat i senyalització a nivell físic i els formats de les trames de dades del nivell d'enllaç de dades del model d'interconnexió de sistemes oberts (OSI).

Firewall: és un sistema de seguretat en xarxa que monitoritza i controla el tràfic en xarxa de sortida i d'entrada basant-se en normes de seguretat predeterminades.

Git: és un software de control de versions dissenyat per Linus Torvalds, pensant en l'eficiència i la confiança del manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font.

Hash (funció informàtica): és qualsevol funció que pot ser utilitzada per "mapejar" dades d'una mida arbitrària a una mida fixada. Els valors retornats per una funció hash s'anomenen valors hash, codis hash, digerits, o simplement hashes. Una de les seves utilitats es troba en criptografia. Una funció hash criptogràfica permet verificar fàcilment que una dada d'entrada condueix a un valor hash donat, però si la dada d'entrada és desconeguda, es molt difícil de reconstruir a partir del valor hash emmagatzemat.

Hping3: és un analitzador/assemblador de paquets TCP/IP orientat a terminal. Suporta els

protocols TCP, UDP, ICMP, i RAW-IP, té un mode de traçat de ruta, l'habilitat d'enviar arxius entre un canal cobert, i moltes altres característiques.

IDS (de l'anglès *Intrusion Detection System*): és un programa de detecció d'accessos no autoritzats a un computador o a una xarxa.

Imatge ISO: és un arxiu informàtic on s'emmagatzema una copia o imatge exacta d'un sistema d'arxius.

IoT (de l'anglès *Internet of Things*): es refereix, en termes d'informàtica, a una xarxa d'objectes, dispositius, vehicles, edificis, i altres elements amb electrònica, software, sensors, actuadors i connectivitat a internet incorporada, que permet a dits objectes recollir i intercanviar dades.

Jitter: és la variabilitat temporal durant l'enviament de senyals digitals, una lleugera desviació de l'exactitud del senyal de rellotge.

Kernel (nucli): en informàtica, un nucli o kernel és un software que constitueix la part fonamental del sistema operatiu, i es defineix com la part que s'executa en mode privilegiat (també conegut com mode nucli). És el principal responsable de facilitar als diferents programes accés segur al hardware de la computadora.

Màquina virtual: en informàtica, és un software que simula un ordinador i pot executar programes com si fos un ordinador real.

Mobaxterm: és un terminal per a Windows amb servidor X11, client SSH, i diverses eines en xarxa per a computació remota (VNC, RDP, telnet, rlogin).

Modem (de l'acrònim anglès de ***mo****dulator* ***dem****odulator*): és un dispositiu que converteix els senyals digitals en analògics (modulació) i a l'inrevés (desmodulació), i permet d'aquesta manera la comunicació entre computadors a través de la xarxa telefònica o del cablemodem.

NAT (de l'anglès *Network Address Translation*): és un mecanisme utilitzat per routers IP per a intercanviar paquets entre dues xarxes que assignen mútuament adreces incompatibles. Consisteix a convertir, a temps real, les direccions utilitzades en els paquets transportats.

Netcat: és una eina en xarxes d'ordinadors per a llegir i escriure a connexions en xarxa mitjançant TCP o UDP.

Norma: en l'àmbit de seguretat en xarxa, una norma descriu característiques indicatives de malícia en el flux de dades. Típicament, una norma consisteix en una llista de valors que han d'estar presents en els camps de capçalera d'un paquet (com per exemple la font o el

destinatari de la informació), juntament amb una seqüència de bytes que ha d'estar també present, o bé una expressió que ha de lligar amb el contingut del tràfic.

Pont de xarxa (*bridge* en anglès): és el dispositiu d'interconnexió de xarxes de computadores que opera en la capa 2 (nivell d'enllaç de dades) del model OSI.

PuTTy: és un client SSH, Telnet, rlogin, i TCP raw amb llicència lliure.

Python: és un llenguatge de programació interpretat la filosofia del qual es centra en una sintaxis que afavoreixi un codi llegible. Es tracta d'un llenguatge de programació multiparadigma, ja que suporta programació orientada a objectes, programació imperativa i, en menor mesura, programació funcional.

Random nonce: en criptografia, una nonce es un número arbitrari pensat per fer servir un cop. És habitualment un número aleatori o pseudoaleatori emès en un protocol d'autenticació per assegurar que les comunicacions antigues no poden ser reutilitzades en atacs de repetició.

Raspbian: és una distribució del sistema operatiu GNU/Linux i per tant, lliure, basat en Debian Wheezy (Debian 7.0) per a la placa computadora Raspberry Pi.

Router o encaminador: és un dispositiu que proporciona connectivitat a nivell de xarxa o nivell tres al model d'interconnexió de sistemes oberts (OSI). La seva funció principal consisteix en enviar o encaminar paquets de dades d'una xarxa a una altra, és a dir, interconnectar subxarxes, entenent per subxarxa un conjunt de màquines IP que es poden comunicar sense la intervenció d'un encaminador.

Script (informàtica): és un programa usualment simple, que generalment s'emmagatzema en un arxiu de text pla. Els scripts són quasi sempre interpretats.

SDN (de l'anglès *Software-Defined Networking*): són un conjunt de tècniques relacionades amb l'àrea de les xarxes computacionals, l'objectiu de les quals és facilitar la implementació i implantació de serveis de xarxa d'una manera determinista, dinàmica i escalable, evitant a l'administrador de la xarxa gestionar dits serveis a baix nivell.

SHA256 (de l'anglès *Secure Hash Algorithm*): és un tipus de funció hash criptogràfica que genera un hash de mida fixada de 256 bits (32 bytes) gairebé únic.

Snort: és un sistema de prevenció d'intrusions a la xarxa obert i gratuït creat per Martin Roesch en 1998, amb l'habilitat de realitzar anàlisi de tràfic en temps real i enregistrament de paquets en xarxes IP. Snort realitza anàlisi de protocol, cerca de contingut i de lligams, i pot ser utilitzat per detectar una gran varietat d'atacs i sondejos, com buffer overflow,

exploracions de port sigil·loses, atacs CGI, sondejos SMB i intents de presa d'empremta OS, entre d'altres.

TCP (de l'anglès *Transmission Control Protocol*): és un dels protocols fonamentals a Internet. Molts programes dins una xarxa de dades formada per xarxes de computadores, poden utilitzar aquest protocol per a crear connexions entre sí a través de les quals pot enviar-se un flux de dades. El protocol garanteix que les dades seran entregades al seu destí sense errors i en el mateix ordre en que es varen transmetre.

TCP SYN Flood: és un tipus d'atac DDoS en el que l'atacant envia peticions de connexió TCP a una velocitat més elevada que la velocitat màxima de processament de la màquina, causant saturació de la xarxa.

Throughput: en el context de comunicacions en xarxa, com Ethernet o packet radio, throughput (o network throughput) és la taxa de missatges enviats amb èxit sobre un canal comunicatiu. Usualment es mesura en bits per segon (bit/s o bps).

Unix: és una família de sistemes operatius multitasca i multiusuari que deriven del sistema operatiu AT&T Unix.

VNC (de l'anglès *Virtual Network Computing*): és un sistema de compartició d'escriptori gràfic que utilitza el protocol RFB (de l'anglès *Remote Frame Buffer Protocol*) per a controlar remotament un altre ordinador.

VPN (de l'anglès *Virtual Private Network*): és una tecnologia de xarxa de computadores que permet una extensió segura de la xarxa d'àrea local (LAN) sobre una xarxa pública o no controlada com Internet. Permet que la computadora en la xarxa enviï i rebi dades sobre xarxes compartides o públiques com si fos una xarxa privada, amb tot el que això suposa.

Web-hosting: és un servei en xarxa que permet a individuals i a organitzacions fer la seva pàgina web accessible mitjançant la xarxa mundial. Inclou aquelles companyies que proporcionen espai en un servidor adquirit o en lloguer per a ús de clients, a més de proporcionar connectivitat a Internet, típicament en un centre de dades.

Win32DiskImager: és un programa de Windows per a desar i restaurar imatges de discs extraïbles.

2. Prefaci

2.1. Motivació

La finalitat principal del projecte és la realització del treball de fi de master del MUEI (Màster Universitari en Enginyeria Industrial) de l'Escola Tècnica Superior d'Enginyeria Industrial de Barcelona. El treball està definit en la memòria dels títols de màster de l'Escola com un exercici original a realitzar de manera individual, que cal presentar i defensar davant d'un tribunal.

Amb aquest projecte també es pretén explorar una nova àrea que està en procés de desenvolupament actualment: l'Internet de les coses (IoT, de l'anglès Internet of Things) i la seva seguretat. L'aparició i proliferació dels dispositius IoT en xarxes industrials, empresarials i domèstiques porta amb sí un risc sense precedents. La magnitud potencial d'aquest risc es va concretar a l'Octubre de 2016, quan un conjunt de càmeres connectades a Internet de forma no segura van llançar un atac distribuït de denegació de servei (DDoS, de l'anglès Distributed Denial of Service) a DynDNS, un proveïdor de servei DNS (de l'anglès Domain Name Servers) per a molts proveïdors de serveis online, com ara Twitter o Reddit. Tot i que aquest incident va causar disrupció a gran escala, es fa notar que l'atac va involucrar només uns pocs centenars de milers de punts finals i una taxa de tràfic d'uns 1,2 TB/s. Amb les prediccions de més d'un bilió de dispositius IoT al llarg d'entre els propers 5 i 10 anys, el risc d'atacs de molta més magnitud és imminent.

Per poder realitzar aquest projecte, ha estat de gran ajuda l'estada a la Universitat de Colorado, a Boulder, amb l'ajuda d'una beca Balsells. Aquesta estada ha permès, d'una banda, reforçar coneixements adquirits sobre programació i electrònica durant el grau i el màster. D'altra banda, s'han adquirit nous coneixements relatius a la programació d'alt nivell i a xarxes. Això suposa una motivació important ja que són les branques de l'enginyeria més interessants personalment dins el marc d'assignatures cursades. Cal destacar que la col·laboració amb dos dels desenvolupadors del projecte base CommunityGuard: el professor Eric Keller i Chase Stewart, ha estat clau per poder documentar, experimentar, i millorar CommunityGuard.

A nivell personal, l'oportunitat de viure a la ciutat de Boulder, als Estats Units, és única i permet conviure amb una cultura diferent i aprendre maneres de pensar diferents. Boulder és una ciutat muntanyosa situada a una vall de les Muntanyes Rocoses, amb un gran ambient universitari. És una ciutat extensa però poc edificada, donant lloc a molts espais verds i facilitant la pràctica d'esports a la natura com l'escalada, ciclisme o altres. També resulta un avantatge aquesta estada ja que facilita la visita a diferents ciutats dins el país.

3. Introducció

CommunityGuard consisteix en un sistema de seguretat de la xarxa col·laboratiu. El terme seguretat de la xarxa fa referència a qualsevol activitat dissenyada per a protegir la integritat de la xarxa i de les dades. Inclou tecnologies de hardware i de software. La seguretat en xarxa efectiva gestiona l'accés a la xarxa i té com a objectiu evitar que diverses amenaces entrin o es propaguin dins la xarxa.

Hi ha diferents tipus de sistemes de seguretat en xarxa. Els més coneguts són els anomenats *antimalware software* o antivirus, i s'utilitzen a qualsevol tipus d'ordinador per a detectar intents d'entrada de virus informàtics (*Trojans, worms, randomware, spyware...*), escanejar arxius per trobar anomalies, eliminar aquests virus i revertir possibles danys. Altres sistemes de seguretat de la xarxa relacionats d'alguna manera amb CommunityGuard són sistemes de control d'accés, tallafocs (*firewalls*), sistemes de prevenció d'intrusions i sistemes d'anàlisi de comportament. Els sistemes de control d'accés gestionen el nivell d'accés a la xarxa de diversos usuaris i dispositius. El segon tipus situa una "barrera" entre la xarxa interna i les xarxes exteriors, com Internet, i utilitzen un conjunt de normes definides per a permetre o bloquejar tràfic. Els sistemes de prevenció d'intrusions escanegen el tràfic de la xarxa per a bloquejar atacs activament. Finalment, els sistemes d'anàlisi de comportament tenen com a objectiu detectar comportament anormal de la xarxa (coneixent prèviament a què s'assembla aquest comportament anormal), i discernir activitats que es desvien del model. Altres sistemes de seguretat de la xarxa sense relació amb CommunityGuard són sistemes de seguretat d'aplicacions, de prevenció de dades, de seguretat del correu electrònic, de dispositius mòbils o de xarxes sense fils.

CommunityGuard és un terme entremig entre un sistema de prevenció d'intrusions i de firewall, que aplica conceptes de sistemes de control d'accés per a la pròpia seguretat. A més, permet assegurar un conjunt de xarxes realitzant anàlisi de comportament, i és complementari a qualsevol antivirus. Està format per uns dispositius intel·ligents anomenats nodes guardians, que aprenen i eviten que el tràfic maliciós entri o surti de la xarxa de l'usuari. Cada node guardià està situat en una xarxa personal diferent (amb ordinadors i dispositus IoT), en un punt tal que tot el tràfic d'aquesta xarxa passi a través d'ell, i poden fer funcions de xarxa per tal de comunicar als altres les amenaces emergents, bloquejant aquestes amenaces per a tots els altres usuaris tan aviat com es generen. D'altra banda, els nodes guardians s'actualitzen periòdicament amb els últims models d'amenaces per proporcionar una seguretat efectiva davant les amenaces emergents.

Tot això ho aconsegueixen amb l'ajuda d'un gestor central: el Community Outpost. Aquest gestor s'executa en un servidor en el núvol que interactua amb cadascun dels nodes

guardians. Community Outpost monitoritza aquest tràfic en cerca de qualsevol problema de seguretat i desplega les defenses que es necessitin. A la Figura 1 es mostra l'esquema de CommunityGuard.

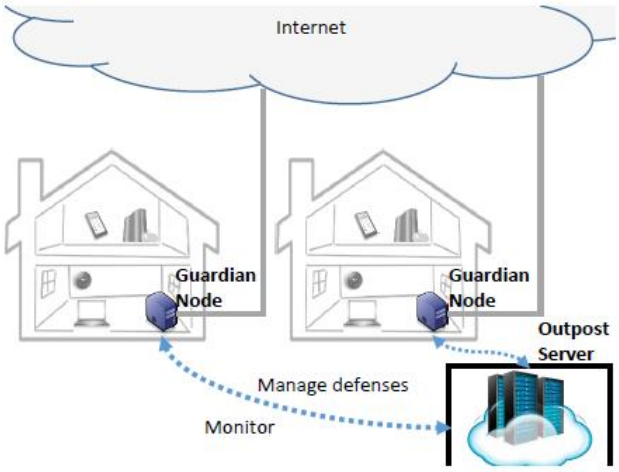


Figura 1: Arquitectura del CommunityGuard [1]

3.1. Objectius del projecte

El projecte consisteix en l'anàlisi, experimentació i millora del projecte CommunityGuard, desenvolupat a la Universitat de Colorado, Boulder, per Chase E. Stewart, Anne Maria Vasu i Eric Keller [1].

El primer objectiu del projecte és l'anàlisi i la documentació de CommunityGuard, per tal d'entendre com s'ha desenvolupat el projecte, amb quins coneixements i amb quines eines. Per tal de fer una anàlisi detallada, es vol fer una recreació del projecte basada en màquines virtuals, situant en un mateix servidor els nodes guardians i el Community Outpost de manera simulada.

Un cop assolit el primer objectiu, s'ha d'analitzar la necessitat d'optimitzar el software inclòs als nodes guardians per a un menor efecte en la capacitat de la xarxa. Amb tal fi, s'han de realitzar proves amb el software dels nodes guardians quan aquests s'implementen en un dispositiu físic. En funció dels resultats, es buscarà (o no) algun mètode d'optimitzar el software.

Finalment, es pretén ampliar el projecte CommunityGuard implementant un sistema de seguretat que actuï directament sobre els dispositius IoT i que resideixi en aquests i en els nodes guardians, sense la necessitat del gestor central Community Outpost.

3.2. Abast del projecte

En aquest apartat es descriuen les limitacions que s'han establert abans de començar el projecte i que afecten al desenvolupament i al resultat del mateix.

En primer lloc, s'ha decidit fer servir components i hardware disponible a la Universitat de Colorado, concretament amb accés per part del professor Eric Keller i dels seus companys de laboratori. Entre aquests dispositius destaquen una Raspberry Pi 2, un ordinador tipus Switch ADI i un servidor, juntament amb monitors, teclats, cables HDMI, cables Ethernet, i altres eines necessàries per al control d'aquests.

D'altra banda, el projecte destaca per la varietat de llenguatges de programació involucrats en el desenvolupament del mateix. En destaquen els propis llenguatges del sistema operatiu Linux, el llenguatge Python i el llenguatge C, però també hi ha alguns aspectes relacionats amb el llenguatge SQL i els llenguatges dels sistemes operatius Debian, OpenSUSE i CentOS7.

Tenint en compte això, no s'ha establert cap limitació relacionada amb aquests llenguatges ja que, durant el grau i el màster, molts d'ells s'han après amb certa profunditat i la seva aplicació suposa un bon complement al contingut curricular.

4. CommunityGuard

Tal i com es descriu a la Figura 1, CommunityGuard es composa de dos entitats diferents: els nodes guardians i el gestor central Community Outpost. Abans d'explicar les especificacions d'aquest sistema, convé aclarir la funció que té cadascuna d'aquestes entitats. El node guardià és el dispositiu que controla el tràfic d'entrada i sortida d'una xarxa en cerca d'activitat sospitosa. És l'encarregat d'alertar al gestor central de tota aquella activitat que trobi sospitosa, i de bloquejar o permetre aquest tràfic segons li ordeni el gestor central. D'altra banda, el gestor central o Community Outpost, és un software executat en un servidor en el núvol. Community Outpost escolta les alertes generades pels nodes guardians sobre activitat sospitosa, i en base a aquestes alertes, decideix si cal bloquejar aquest tràfic o no. En cas afirmatiu, ordena als nodes guardians que bloquegin aquest tipus de tràfic.

Tenint en compte això, als següents apartats es defineix amb detall quines són les característiques que presenta, d'una banda, el prototip del projecte CommunityGuard original. Això inclou, per exemple, el nombre de Nodes Guardians en les proves experimentals, l'efecte dels nodes en la taxa de tràfic del router, i els models d'atacs a prevenir.

D'altra banda, s'expliquen les especificacions que ha de complir el CommunityGuard ampliat. Aquestes fan referència a la millora de la seguretat dels dispositius IoT.

4.1. Especificacions de sistema

4.1.1. Ubicació node guardià

El node guardià és la peça clau de l'arquitectura de CommunityGuard, i ha d'estar situat en una posició on pugui veure els fluxos de tràfic complets d'entrada i sortida de la xarxa de l'usuari, i ha de bloquejar tot tràfic sospitós. A més, l'addició d'aquest dispositiu ha de ser el menys intrusiva possible, de manera que no s'hagi de fer cap modificació en la configuració del Modem o Router (encaminador). Basant-se en això, es decideix que dispositiu estigui entre el Modem i el Router. Per a xarxes que tinguin el Modem i el Router en el mateix dispositiu, es preveu que en una implementació a escales més grans, el Node Guardià estaria situat en la mateixa capsa que aquests, presentant una arquitectura semblant a la de la Figura 1. El fet de col·locar els nodes guardians al punt d'entrada de les xarxes domèstiques, permet crear una xarxa de protecció sòlida que es capaç de comunicar amenaces tan bon punt com són generades.

La ubicació del node guardià implica un possible coll d'ampolla, podent arribar a estancar el

tràfic tant d'entrada com de sortida de la xarxa domèstica. Així doncs, l'elecció del hardware i la programació del software del node guardià ha de ser tal que el seu efecte sobre la velocitat de la xarxa sigui menyspreable. Amb tal fi, es fixa un mínim de 50 Mbps per la velocitat de transmissió de dades de les interfícies Ethernet del node guardià.

4.1.2. Nombre de nodes guardians

Per poder verificar l'eficàcia de CommunityGuard, és necessari fer proves amb, com a mínim, 2 nodes guardians en xarxes diferents. Això és degut a l'aspecte col·laboratiu que presenta el sistema, que es basa en que les amenaces detectades en una ubicació han d'ajudar a protegir les altres, usant el servidor central o Community Outpost. Es fa notar que la capacitat del sistema per detenir amenaces es proporcional a la quantitat de subxarxes que inclogui.

4.1.3. Tràfic maliciós a bloquejar

Existeixen molt tipus d'atacs en xarxa, cadascú dels quals suposa un tràfic maliciós diferent. Per a detectar aquest tràfic, els nodes guardians es basen en un sistema de detecció d'intrusions (IDS: de l'anglès *Intrusion Detection System*), que es recolza en un conjunt de normes dissenyades per detectar amenaces específiques i que no tenen perquè ser igual d'efectives en totes les xarxes. La funció del sistema no és ni desenvolupar ni millorar aquestes normes, sinó fer servir un conjunt de normes disponibles gratuïtament online als repositoris Snort IDS, i comprovar amb certa periodicitat l'existència d'actualitzacions o de noves normes per descarregar-les. El sistema, per tant, no ha de bloquejar un tipus de tràfic maliciós concret (com ara DDoS, atacs CGI, sondejos SMB, etc.), sinó treure profit de les normes disponibles a Snort. Cal destacar que el sistema tracta per separat els atacs DDoS.

4.1.4. Community Outpost

Community Outpost és la font d'intel·ligència per als nodes guardians als que serveix. Ha de ser capaç de rebre la informació sobre possibles amenaces en els seus nodes guardians associats, analitzar i processar aquesta informació per determinar amenaces vàlides, i presentar les amenaces vàlides als nodes guardians per a que bloquegin les corresponents adreces IP.

Donat que el servidor on s'executi el Community Outpost és un objectiu obvi per a qualsevol que vulgui parar el servei de CommunityGuard, es vol donar un cert nivell de seguretat mínim al servidor. En aquest projecte, el Community Outpost ha d'estar dissenyat de manera que minimitzi el risc de patir qualsevol dels següents 3 tipus d'atacs:

1. Afegir adreces IP a la llista negra per a bloquejar-les pels usuaris.
2. Eliminar adreces IP malicioses de la llista negra.

3. Llegir dades de l'usuari de la llista del servidor.

Es preveu que la seguretat del Community Outpost augmenti en el moment que s'implementi a escala de producció.

4.2. CommunityGuard Ampliat

En el cas de l'ampliació de CommunityGuard, les especificacions són més simples. El node guardià ha de ser un dispositiu que, amb la supervisió d'una persona, pugui permetre o rebutjar les accions que intentin realitzar els dispositius de la xarxa, intentant cobrir el màxim tipus d'accions i de sistemes operatius dels dispositius IoT.

El sistema a dissenyar no necessita que el node guardià estigui ubicat entre el router i el cable modem, sinó que és suficient amb què aquest tingui accés a la resta de dispositius de la xarxa, i el nombre de nodes guardians mínim amb els que fer les proves passa a ser 1, ja que aquest sistema ja no presenta l'aspecte col·laboratiu del CommunityGuard original.

5. Descripció del prototip CommunityGuard

En aquest apartat es detalla quins elements es fan servir per muntar el prototip CommunityGuard original, com funciona i com està estructurat el software del node guardià i del Community Outpost, així com els resultats obtinguts de les primeres proves realitzades.

5.1. Hardware

5.1.1. Node Guardià

El projecte original CommunityGuard utilitza un dispositiu BeagleBone Black (Figura 2) com a node guardià i una tarjeta SD per muntar el sistema operatiu necessari. El BeagleBone Black és una placa computadora de hardware lliure de baix consum produïda per Texas Instruments, que compta amb totes les funcions bàsiques d'una computadora bàsica. Com a característiques rellevants, inclou un processador ARM Cortex-A8 que pot córrer els sistemes operatius Linux, Minix, FreeBSD, OpenBSD, RISC OS o Symbian, i presenta un port Ethernet 10/100 Mbps. A més, se li ha afegit una altra interfície usant un adaptador de USB a Ethernet 10/100 Mbps necessària per al seu funcionament, ja que una interfície és per a la connexió amb el Mòdem i l'altra pel Router.

Figura 2: BeagleBone Black

5.1.2. Community Outpost

El gestor central que interactua amb tots els nodes guardians s'executa en un servidor. Tenint en compte que per executar el Community Outpost no es necessiten molts recursos computacionals, hi ha diverses opcions per escollir el servidor. Es podria emprar un servidor

de AWS (de l'anglès Amazon Web Services), el qual proporciona comptes educacionals gratuïts durant uns mesos, o Atlantic.net, per exemple.

5.2. Software

En els següents subapartats s'explica el software inclòs en els nodes guardians i en el Community Outpost i el seu funcionament, tot de forma qualitativa. A l'Annex A es mostra el codi original del prototip de Community Guard i s'explica com està estructurat, i també es mostren algunes modificacions bàsiques realitzades que fan el codi més independent de l'entorn, que corregeixen errors menors i acaben d'implementar-lo per complet.

5.2.1. Node Guardià

En el prototip, el node guardià (materialitzat en el BeagleBone Black) està programat en un sistema operatiu Debian Linux. Té un servidor DHCP configurat per tal que el router pugui ser afegit darrere d'ell, i passa tot el tràfic d'una interfície de xarxa a l'altra. A la Figura 3 es mostra el disseny del software del node guardià

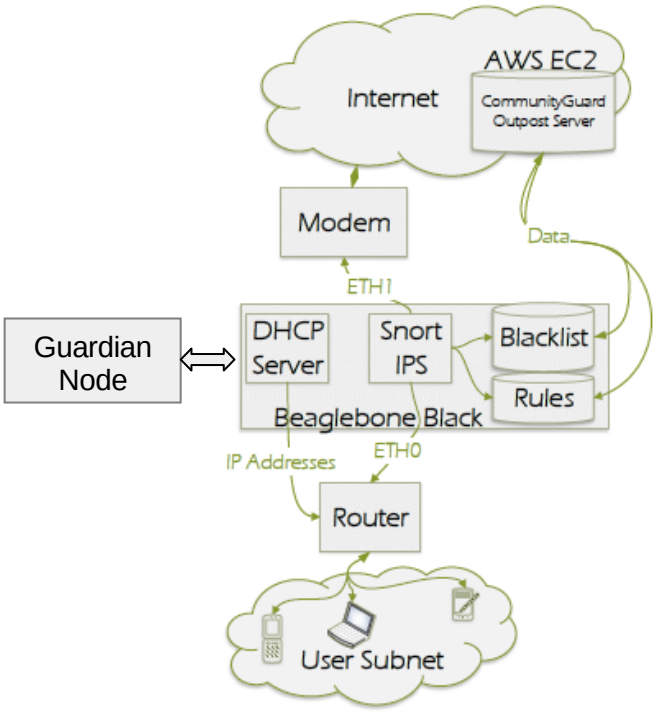


Figura 3: Prototip CommunityGuard [1]

El node, tal com s'ha especificat anteriorment, executa Snort, una eina de prevenció/detecció d'intrusions que monitora el tràfic a ambdues interfícies. Un conjunt de normes Snort són configurades per bloquejar qualsevol tràfic sospitós que satisfaci alguna

de les normes. Snort permet fer anàlisi de protocol, cerca i lligams de contingut, i pot ser utilitzat per detectar una gran varietat d'atacs i sondejos (buffer overflow, exploracions de port sigil·loses, atacs CGI, sondejos SMB, intents de captura d'empremta OS...). A més, presenta una funcionalitat de llista blanca i llista negra, que pot ser utilitzada per bloquejar i desbloquejar condicionalment adreces IP malicioses.

La comunicació d'alertes d'amenaces es un aspecte clau de CommunityGuard. Aquesta comunicació es realitza de forma periòdica mitjançant un conjunt de tres tasques executades al node guardià, que tenen les següents funcions:

- Mantenir les normes generals de Snort al dia obtenint-les dels repositoris de normes, juntament amb les adreces IP.
- Enviar el tràfic sospitós detectat al node al Community Outpost i rebre noves normes i adreces IP bloquejades del Community Outpost.
- Comprovar les balises (*beacon* en anglès) DDoS del servidor i generar noves normes DDoS si és necessari.

La primera tasca s'executa un cop al dia a un temps aleatori, i cerca la llista de direccions IP malicioses més recent al Community Outpost, i les afegeix a la llista negra de Snort. També cerca normes del projecte de seguretat Emerging Threats de Snort per defensar-se contra nous atacs.

La segona tasca s'executa periòdicament per llegir un arxiu de registre creat per Snort que conté adreces IP de fonts que han enviat tràfic maliciós. Després analitza les adreces IP font presents en el registre, i actualitza el mateix contingut en una base de dades del Community Outpost (descrita més endavant).

La tercera tasca s'executa periòdicament per cercar informació d'una llista de vigilància DDoS disponible al Community Outpost, com a part del mecanisme de prevenció DDoS de sortida que s'explica en la secció següent. Si la base de dades proporciona noves adreces IP que estan sent víctimes d'un atac DDoS, la tasca crea i afegeix normes Snort per bloquejar el corresponent tràfic de sortida DDoS. L'algoritme és descrit a la Figura 4, on DDoS_table representa la llista de vigilància del Community Outpost, Cron representa la tasca periòdica, i BBB representa el dispositiu on està el node guardià, el BeagleBone Black.

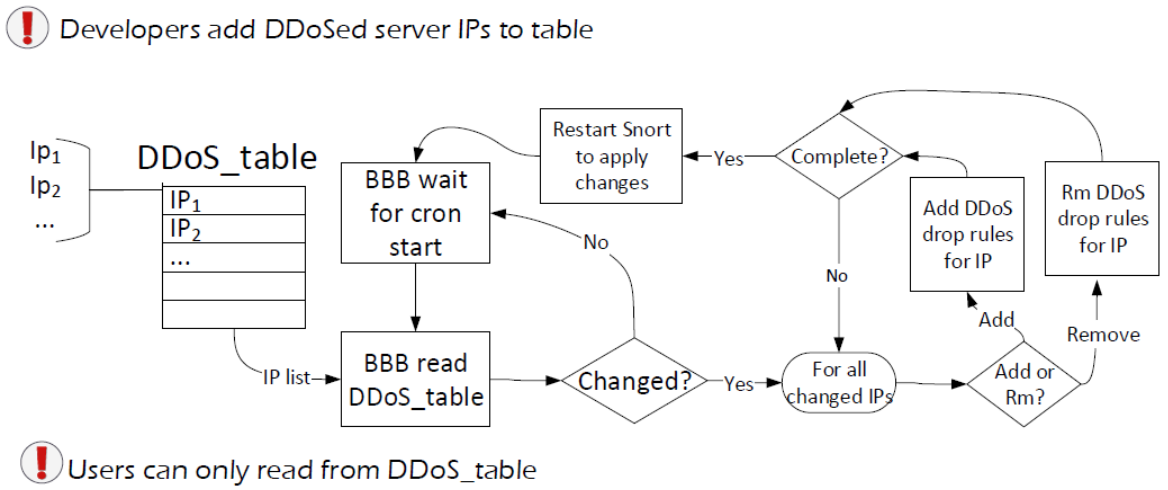


Figura 4: Algoritme de prevenció de DDoS de sortida [1]

Aquestes 3 tasques, combinades amb l'execució de Snort a tots els nodes guardians, estableixen un sistema de protecció que aprèn, comunica i té la capacitat de prevenir la majoria d'atacs.

5.2.2. Community Outpost

Community Outpost està pensat de manera que el servidor on s'executi corri el sistema operatiu CentOS. El servidor conté una base de dades que utilitza per afegir les amenaces presentades per tots els nodes guardians en conjunt, i processa aquesta informació per determinar amenaces vàlides que presenta posteriorment als nodes guardians per a la seva lectura i processament.

Per tal de comunicar-se amb tots els nodes guardians amb seguretat, s'actua com es mostra a la Figura 5. Quan un node guardià és creat, un hash SHA256 de l'adreça MAC del node i una *random nonce* emmagatzemada a la memòria del node són entrats a una taula de la base de dades anomenada *mac_addr_registry*. El valor hash serveix com a clau per a escriptures, per assegurar que aquestes puguin ser iniciades únicament des d'un node guardià vàlid. El node guardià és l'únic que té permís d'escriptura a la taula anomenada IPv4_input, i permís de lectura de la taula IPv4_output. Aquesta política evita els models d'atac 1, 2 i 3 de l'apartat 4.1.4. A més, donat el cas en que un usuari maliciós practiqués enginyeria inversa amb el node guardià per tenir contacte directe amb Community Outpost, l'accés que guanyaria seria mínim. Quan a un node guardià li correspon enviar informació sobre tràfic maliciós al Community Outpost, només li és permès escriure a la taula IPv4_input, on escriu la direcció IP sospitosa, i el hash SHA256 de la seva adreça MAC convertida a tipus de dada enter i de la *random nonce*.

Community Outpost processa la taula IPv4_input per recalcular noves amenaces a una certa

freqüència, i després buida la taula IPv4_input actual. Per cada entrada en la taula, el servidor primer comprova el hash SHA256 passat pel node guardià per tal d'assegurar que la clau estigui en el registre mac_addr_registry. Les entrades que no tinguin un hash SHA256 vàlid són descartades, deixant només les sol·licituds vàlides. Per a totes les sol·licituds vàlides, un segon hash SHA256 és calculat, aquest cop del primer resultat SHA256 i també de l'adreça IP declarada com sospitosa. El servidor comprova si aquest segon hash està a la taula IPv4_master_list, que és una llista de les combinacions SHA256 i IP que s'han entrat anteriorment. Si és una adreça IP sospitosa nova o si és una adreça IP sospitosa coneguda però reportada per un node guardià nou i vàlid, serà entrada a la taula IPv4_output. Aquesta segona funció hash assegura que una instància donada del Community Outpost només rebi un vot cap a una adreça IP determinada, com a mitjà de prevenció contra el model d'atac 1 de les especificacions. Si l'adreça IP és una nova entrada, serà col·locada a la taula amb un compte de 1; si ja existeix a la taula, el seu compte serà incrementat. Un cop el compte per una adreça arribi a un nombre suficientment gran en proporció al nombre d'amenaces detectades, la IP serà passada als nodes guardians sol·licitants, i bloquejada automàticament per tots els nodes. Basar-se en un compte abans de bloquejar una adreça sospitosa també ajuda a evitar que el model d'atac 1 tingui lloc.

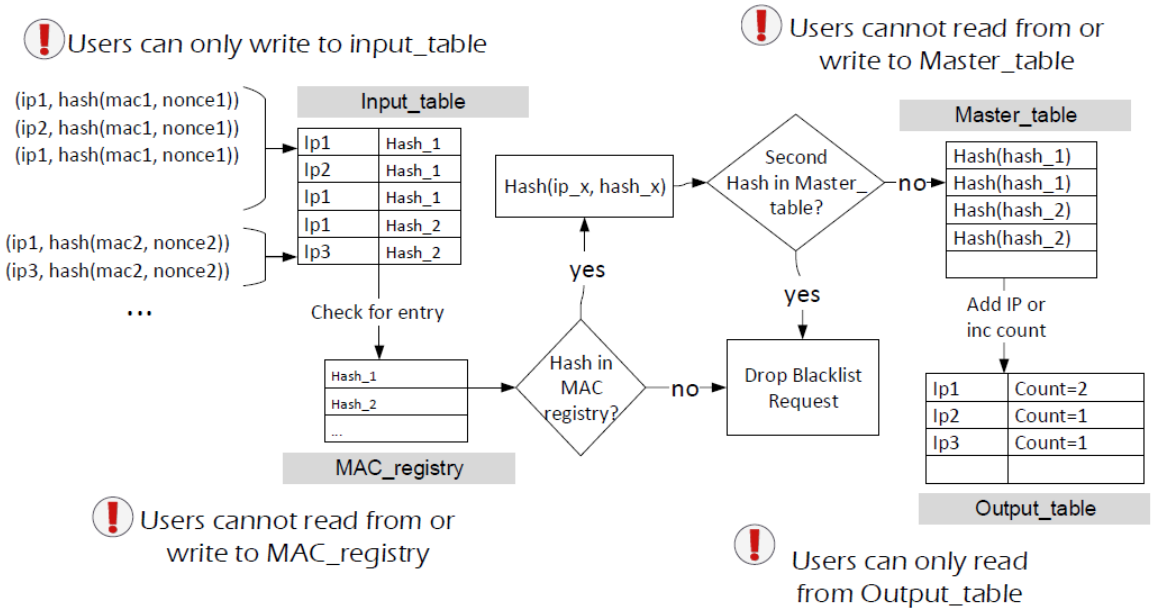


Figura 5: Community Outpost push and pull [1]

Community Outpost també proveeix d'un mètode per evitar el tràfic DDoS de sortida des de la subxarxa d'un node guardià cap a un objectiu extern DDoS. Aquest sistema és molt més senzill que l'anterior: en aquest cas, la llista de víctimes DDoS és editada per administradors de CommunityGuard, i només la lectura és permesa per als nodes guardians. Els administradors confirmarien l'existència d'un atac DDoS a través d'un canal alternatiu,

possiblement una tercera part DDoS treballant conjuntament amb ells, i afegirien les adreces IP rellevants o rangs CIDR (de l'anglès *Classless Inter-Domain Routing*) a la taula IPv4_ddos. Els nodes guardians utilitzen la tasca descrita a la Figura 4 per afegir o eliminar normes per evitar atacs DDoS de sortida.

5.3. Avaluació

L'avaluació del prototip de CommunityGuard es fa sobre dos aspectes: l'efectivitat de la seva operació prevista i el rendiment de la xarxa mentre el sistema està sent executat en línia.

5.3.1. Operació

Per comprovar l'efectivitat del sistema, s'avaluen dues capacitats bàsiques del CommunityGuard que són descrites més endavant. La preparació per a la prova inclou dos nodes guardians presents en diferents xarxes, que són usats per comprovar si les adreces IP en llista negra actualitzades per un node guardià són comunicades a l'altre. Els dos nodes guardians es connecten al servidor Community Outpost amb la mateixa periodicitat.

La primera capacitat a avaluar és la prevenció i registre del tràfic maliciós d'entrada. Es configuren normes Snort per bloquejar tràfic IP que contingui contingut maliciós. Hi ha milers de normes Snort disponibles que poden alertar i bloquejar aquest tràfic. En aquest cas, però, no es vol rebre tràfic maliciós real en les proves degut a les restriccions de les infraestructures i dels recursos. El que es fa és, escriure unes quantes normes Snort per tractar algun tràfic segur arbitrari com a maliciós i es configuren les normes per bloquejar aquest tràfic. La tasca periòdica dels nodes guardians (descrita a l'apartat 5.2.1), analitza el registre i actualitza la base de dades sobre la font des d'on possiblement el tràfic maliciós és generat i, depenent d'una majoria de vots entre tots els nodes guardians (majoria elevada artificialment en les proves ja que només hi ha 2 nodes), el Community Outpost procedeix a afegir les adreces font IP malicioses condicionalment a la llista de bloqueig. Els nodes guardians busquen noves adreces IP dolentes confirmades quan executen la tasca periòdica encarregada d'afegir les adreces IP a la llista negra.

La segona capacitat a avaluar és la prevenció d'atacs DDoS de sortida. Per comprovar aquesta capacitat, es poden afegir adreces IP "fictícies" (víctimes d'un suposat atac DDoS), a la llista de vigilància del Community Outpost. Les adreces IP poden ser de màquines posades expressament en una altra xarxa. Un cop afegides les adreces, es simula un atac TCP SYN DoS (*flood*) amb hping3. La tasca periòdica encarregada de comprovar actualitzacions en la base de dades cerca aquestes adreces IP cada minut i busca coincidències en la llista existent d'adreces IP DDoS mantinguda a cada node guardià. Si no es troba cap coincidència, es crea una nova norma Snort per a l'adreça en qüestió per evitar tot tràfic DDoS de sortida. A la Figura 6 es mostra un exemple de norma Snort i una fracció

de l'arxiu de registre corresponent creat durant l'atac. Cal destacar que, d'aquesta manera, la xarxa continua podent enviar tràfic legítim a les adreces IP atacades ja que les normes Snort estan configurades per analitzar el patró d'un atac DDoS abans de bloquejar el tràfic. És a dir, que es pot enviar tràfic benigne TCP, fet que es pot comprovar usant l'eina netcat per comunicar-se entre el sistema atacant i el sistema atacat mentre Snort està bloquejant els atacs TCP SYN DoS de sortida cap a l'adreça IP objectiu.

Snort rule

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (flags: S; msg:"Possible TCP DoS"; f
low: stateless; threshold: type both, track by_dst, count 70, seconds 10; sid:10
0001;rev:1;)
```

Snort log

```
"Possible TCP DoS",TCP,12/11-08:22:48.025236 ,192.168.3.4,41224,10.0.0.2,80
"Possible TCP DoS",TCP,12/11-08:22:58.021326 ,192.168.3.4,51812,10.0.0.2,80
"Possible TCP DoS",TCP,12/11-08:23:08.033561 ,192.168.3.4,16528,10.0.0.2,80
"Possible TCP DoS",TCP,12/11-08:23:18.019386 ,192.168.3.4,44599,10.0.0.2,80
```

Figura 6: Norma Snort DDoS de mostra i arxiu de registre corresponent [1]

5.3.2. Rendiment de la xarxa

Com ja s'ha comentat a l'apartat 4.1.1, el node guardià està dissenyat per a ser col·locat entre el Router i el cable Modem, el que implica que pot estar introduint un possible coll d'ampolla i estancant el tràfic d'entrada i sortida de la xarxa personal. Per això, és necessari verificar el rendiment de la xarxa (velocitats de baixada, de pujada, i latència) mentre el node guardià està en ple funcionament en diferents condicions de càrrega de la xarxa. Es va fer la prova per a 3 condicions de càrrega de diferents:

- Càrrega baixa (*Low load*):1-3 vídeos a temps real, uns pocs navegadors amb contingut flash mitjà, una pujada activa i pàgines de mitjans socials en 6 dispositius diferents, entre els quals hi havia tauletes, ordinadors i telèfons mòbils.
- Càrrega mitjana (*Average load*): 3-5 vídeos a temps real, 4-6 pàgines amb contingut flash elevat i 2-3 pujades actives repartides entre varis dispositius.
- Càrrega pesada (*Heavy load*): 6-8 vídeos a temps real, 8-10 pàgines amb contingut flash elevat i 4-5 pujades actives distribuïdes entre varis dispositius.

Els resultats són els que es mostren a la Figura 7, on Test és el cas en què CommunityGuard està en funcionament i Baseline és el cas en les mateixes condicions de càrrega sense CommunityGuard funcionant. Com es pot observar, les dades sobre el rendiment són bastant similars entre Test i Baseline. En mitjana, el cas Test sembla ser lleugerament inferior al cas Baseline, fet que es pot atribuir a les següents deficiències del hardware:

- L'adaptador USB a 10/100 Mbps Ethernet afegit al node guardià, necessari per al

seu funcionament, limita la velocitat de transferència degut al límit màxim en la velocitat màxima de conversió.

- El node guardià s'executa en un sistema operatiu Debian Linux muntat en una tarjeta SD, que presenta temps d'escriptura més lents en comparació a la resta de hardware.

Figura 7: Resultats dels test de rendiment Snort [1]

També es considera adequat analitzar la utilització de RAM en funció de les normes incloses. La Figura 8 mostra la utilització de la memòria RAM sense Snort, amb Snort actiu i 156 normes, i amb Snort actiu i amb 9863 normes. L'equip desenvolupador del prototip CommunityGuard creu que el conjunt de normes Snort incorporades podria ser optimitzat per a un millor ús del processador i de la memòria RAM. Cal destacar que l'increment de la velocitat del tràfic de 100 Mbps a 400 Mbps augmentaria el requisit de memòria RAM fins gairebé 1GB per a 100-10000 normes. La implementació de Snort en el node guardià usa molta menys memòria RAM ja que el tràfic de la xarxa està limitat a 100Mbps per la targeta Ethernet. Treballar amb velocitats de tràfic més elevades requeriria hardware especialitzat. En una implementació ideal, el node guardià tindria una targeta de xarxa amb característiques similars a la targeta de xarxa del Router corresponent.

RAM utilization at Guardian node without Snort instance

```
KiB Mem:    503912 total,   280868 used,   223044 free,    20912 buffers
KiB Swap:        0 total,        0 used,        0 free.    58264 cached Mem
```

RAM utilization at Guardian node with 156 Snort rules configured

```
KiB Mem:    503912 total,   479568 used,    24344 free,    20856 buffers
KiB Swap:        0 total,        0 used,        0 free.    58244 cached Mem
```

RAM utilization at Guardian node with 9863 Snort rules configured

```
KiB Mem:    503912 total,   481016 used,    22896 free,      568 buffers
KiB Swap:        0 total,        0 used,        0 free.    28896 cached Mem
```

Figura 8: Utilització de la RAM al node guardià [1]

6. Recreació prototip CommunityGuard

En aquest apartat s'explica amb detall la recreació del projecte original CommunityGuard, ubicant tots els elements en una mateixa màquina: un servidor de la Universitat de Colorado, Boulder, i utilitzant Ubuntu com a sistema operatiu dels nodes guardians (en lloc de Debian Linux original).

6.1. Accés al Servidor

Per implantar aquesta primera alternativa el primer pas es aconseguir accés a un servidor de la Universitat de Colorado, Boulder. Això s'ha fet amb la col·laboració del professor Eric Keller del departament de Computer Science. El servidor proporcionat, que empra el sistema operatiu Linux, està protegit pel firewall del Campus de la Universitat. Per creuar aquest firewall hi ha dues opcions:

a. Estar connectat a la xarxa Wifi del Campus.
b. Tenir accés a Internet i configurar la connexió amb el VPN (de l'anglès *Virtual Private Network*) de la universitat [2], que té l'adreça de servidor *vpn.colorado.edu*. Això només es pot fer si es té un compte de la Universitat de Colorado, que proporciona credencials per accedir-hi. El programa utilitzat per configurar aquesta connexió s'anomena Cisco AnyConnect.

Un cop sobrepassat el firewall, s'estableix una connexió amb el servidor de manera segura mitjançant SSH (de l'anglès Secure Shell). SSH es un protocol de xarxa criptogràfic per serveis operatius en xarxa en un entorn no segur. La seva aplicació més usual, que coincideix amb la que s'empra en aquest cas, és iniciar sessió remotament a un ordinador amb un usuari. Hi ha diverses maneres de fer aquesta connexió segons el tipus de sistema operatiu. En aquest projecte es consideren les connexions des de Linux i des de Windows.

En el cas de Linux [3], es poden seguir els següents passos:

1. Obrir un terminal de Linux i generar una nova clau SSH amb la comanda:

   ssh-keygen
2. Preguntarà pel lloc on desar la clau. Escriure el nom de l'arxiu i ubicació. Per defecte serà *~/.ssh/id_rsa.* Quan es genera la clau, es creen dos arxius: un públic i un privat. L'arxiu privat no s'ha de compartir mai amb cap altre usuari, mentre que l'arxiu públic s'ha de compartir amb el computador al que es vulgui accedir remotament.
3. També preguntarà si es vol establir una *Passphrase*. Una Passphrase es una contrasenya que s'utilitza en el moment d'iniciar cada connexió SSH. Aquest pas

és opcional.

4. Afegir la clau a l'agent SSH. Per fer-ho, es recomana comprovar que el procés *ssh-agent* estigui actiu. Això es pot fer amb la comanda:

   eval $(ssh-agent –s)

   Si el procés està actiu, es veurà per pantalla *Agent pid [NÚM]*. Un cop feta aquesta comprovació, es procedeix a fer l'addició amb la comanda següent:

   ssh-add ~/.ssh/id_rsa

   Si no s'ha escollit la ubicació i nom per defecte, la comanda anterior s'ha d'adaptar.
5. Abans de poder realitzar la connexió amb el servidor, és necessari enviar l'arxiu públic de la clau generada al professor Eric Keller, i que ell, des del servidor, l'afegeixi a la llista de claus conegudes. A més, també és necessari disposar de les credencials d'un usuari dins el servidor per tal de poder iniciar sessió en un terminal del servidor.
6. Finalment, es pot establir una connexió SSH amb el servidor, des d'un terminal, escrivint la següent comanda:

   ssh -X [nom_usuari]@[adreça_IP_servidor]

   En el cas que l'usuari del servidor tingui assignada una contrasenya en el servidor, s'ha d'introduir.

Després de seguir aquests passos, el terminal des d'on es fa la connexió passa a ser un terminal del servidor.

Pel cas de Windows, la connexió SSH es pot fer amb el programa Mobaxterm [4], seguint els passos següents:

1. Obrir Mobaxterm.
2. Iniciar un terminal local (*Start Local terminal*) i repetir els passos 1 a 4 per configurar SSH en el cas de Linux.
3. Crear una nova sessió tipus SSH amb els següents ajustos:
   a. *Basic SHH settings*: introduir adreça IP del servidor, nom d'usuari i port de connexió (port 22).
   b. *Advanced SSH settings*: marcar les caselles *X11-Forwarding*, *Display SSH-browser*, i *Use private key*. En aquesta última casella cal seleccionar l'explorador d'arxius i escollir la clau privada que s'hagi generat anteriorment. Com a *Remote environment*, es por escollir *Interactive shell*.
4. Repetir el pas 5 del cas de Linux.
5. Des de Mobaxterm, a la pestanya de *Saved sessions*, fer doble click amb el ratolí sobre la sessió creada anteriorment per establir la connexió SSH amb el servidor.

6.2. Preparació del servidor

Després de tenir accés al servidor, el següent pas es preparar-lo per poder allotjar el Community Outpost i els nodes guardians. Amb tal fi, es pot instal·lar KVM (de l'anglès *Kernel-based Virtual Machine*). KVM és un mòdul de virtualització per al Linux Kernel que el converteix en un "Hipervisor", és a dir, que es pot utilitzar KVM per executar múltiples sistemes operatius com Windows, Linux o CentOS usant màquines virtuals. Cada màquina virtual té el seu disc privat, targeta gràfica i de xarxa, memòria RAM, etc. La idea consisteix en utilitzar una màquina virtual per al Community Outpost, una per cada node guardià, i una altra per cada dispositiu IoT (simulat) que hagi de ser protegit per un node guardià.

Donat que el sistema operatiu del servidor és Ubuntu, s'explicaran les comandes d'instal·lació per aquest sistema operatiu. El primer pas és instal·lar VNC server (de l'anglès *Virtual Network Computing*) al servidor per tal de poder controlar remotament les màquines virtuals instal·lades al servidor. VNC aconsegueix això mitjançant la transmissió de les tecles polsades i del moviment i clics del ratolí a les màquines virtuals. Per instal·lar VNC server, s'han d'entrar les següents comandes en el terminal de Linux, un cop establerta la connexió SSH amb el servidor [5]:

sudo apt-get install ubuntu-desktop gnome-panel gnome-settings-daemon metacity nautilus gnome-terminal

sudo apt-get install vnc4server

sudo apt-get install xtightvncviewer

Tot seguit, es procedeix a instal·lar KVM [6] amb la comanda següent:

sudo apt-get install qemu-kvm libvirt-bin virtinst bridge-utils

Després de la instal·lació es creen els següents directoris d'interès:

- /var/lib/libvirt/boot: per emmagatzemar imatges ISO per a la instal·lació de sistemes operatius.
- /var/lib/libvirt/images: directori d'instal·lació de les màquines virtuals.
- /etc/libvirt: directori de configuració.

Abans d'instal·lar cap màquina virtual, s'ha de verificar que el servidor tingui els recursos necessaris per executar KVM:

sudo kvm-ok

En cas afirmatiu, es pot procedir a descarregar l'arxiu ISO del sistema operatiu que es vulgui implementar. Es començarà pel Community Outpost i, per tant, per CentOS [7]. Un arxiu ISO de CentOS es pot obtenir introduint al terminal:

cd /var/lib/libvirt/boot/

sudo wget https://mirrors.kernel.org/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal.1611.iso

Després de descarregar l'arxiu ISO, s'utilitza la comanda *virt-install* [6] amb les opcions detallades a la Taula 1 per instal·lar la màquina virtual. Es possible que en el moment d'introduir aquesta comanda aparegui una advertència. Per poder continuar amb la instal·lació, cal establir una connexió remota amb la màquina que s'està instal·lant i seguir les instruccions que apareguin des de la màquina. Per fer-ho, es poden introduir, en un altre terminal del servidor, les comandes següents. És important el nom d'usuari del Community Outpost, ja que s'ha d'introduir posteriorment en un fitxer de configuració.

- *sudo virsh vncdisplay [nom_màquina]*: retorna una adreça que serveix per establir connexió remota.
    - *vncviewer [adreça]*: estableix connexió remota amb la màquina virtual.

En quant acaba la instal·lació, es pot iniciar, apagar, editar o eliminar la màquina virtual. A continuació es mostren les comandes que ho permeten fer (des del servidor) i altres comandes d'utilitat:

- *sudo virsh list --all*: mostra les màquines virtuals i el seu estat.
- *sudo virsh start [nom_màquina]*: encén la màquina virtual especificada.
- *sudo virsh shutdown [nom_màquina]*: apaga la màquina virtual especificada.
- *sudo virsh destroy [nom_màquina]*: apaga la màquina virtual especificada de manera forçada.
    - *sudo virsh undefine [nom_màquina]*: elimina la màquina virtual. És recomanable, per tal d'eliminar completament la màquina virtual, eliminar els corresponents arxius ISO i la imatge d'instal·lació.
- *sudo virsh edit [nom_màquina]*: permet modificar característiques de la màquina virtual especificada, com ara la memòria RAM o les interfícies de xarxa.

Comanda i opcions	Descripció
sudo virt-install --virt-type=kvm \	Estableix KVM com hipervisor per instal·lar CentOS
--name centos7 \	Nom de la instància de la màquina virtual.
--ram 1024 \	Memòria RAM en MB.
--vcpus=1 \	Nombre de processadors virtuals.
--os-variant=rhel7 \	Optimitza la configuració per a una variant específica de sistema operatiu.
--hvm \	Necessita ús de virtualització completa.
--cdrom=/var/lib/libvirt/boot/CentOS-7-x86-64-Minimal-1611.iso \	Ubicació de l'arxiu ISO.
--network network=default, model=virtio \	Connecta la màquina virtual a la xarxa.
--graphics vnc \	Configura la consola virtual i l'exporta com VNC server al servidor.
--disk path=/var/lib/libvirt/images/centos7.img, size=20,bus=virtio	Estableix la ubicació i nom de l'arxiu d'instal·lació, així com l'espai de disc dur a utilitzar.

Taula 1: Comanda d'instal·lació de CentOS

Seguint el mateix procediment que en el cas del sistema operatiu CentOS, es poden instal·lar les altres màquines virtuals. En aquest cas, s'instal·len 4 màquines més amb sistema operatiu Ubuntu 16.04.2: 2 nodes guardians i 2 dispositius IoT que fan passar el seu tràfic en xarxa pel seu node guardià corresponent. L'arxiu ISO es pot descarregar fent:

cd /var/lib/libvirt/boot

sudo wget http://releases.ubuntu.com/16.04/ubuntu-16.04.2-desktop-amd64.iso

Es fa notar que les 4 màquines poden partir del mateix arxiu ISO, però han de tenir noms diferents. Això s'estableix en el moment de fer la instal·lació, a les opcions *--name* i *--disk* s'expliquen a la Taula 1. Com a exemple, es mostra una possible comanda d'instal·lació

d'una de les màquines:

sudo virt-install --virt-type=kvm --name ubuntu1 --ram 1024 --vcpus=1 --hvm --cdrom=/var/lib/libvirt/boot/ubuntu-16.04.2-desktop-amd64.iso --network network=default,model=virtio --graphics vnc --disk path=/var/lib/libvirt/images/ubuntu1.img,size=20,bus=virtio

Cal destacar que les màquines virtuals s'instal·len amb una distribució del teclat per defecte. Per ajustar la distribució del teclat al teclat d'un ordinador concret, s'ha de fer des de dos llocs diferents. En primer lloc, s'ha d'editar la màquina virtual amb la comanda *virsh edit* i afegir, a la línia que conté *<graphics type='vnc' port '-1' autoport='yes'/>*, *el terme keymap='es'* (entre els símbols "<" i "/>") en el cas del teclat espanyol, per exemple. En segon lloc, s'ha de tornar a especificar la distribució del teclat des de la pròpia màquina virtual. En el cas d'Ubuntu, es fa en ajustos del sistema, i en el cas de CentOS, es pot fer de la manera següent [8]:

sudo yum install kbd
localectl set-keymap es

Després d'haver instal·lat i ajustat el teclat de les màquines virtuals necessàries, cal configurar-les de la següent manera: el Community Outpost i els nodes guardians han de tenir accés a Internet, mentre que els dispositius IoT han de conduir tot el tràfic en xarxa a través dels seus nodes guardians associats. Abans de tot, però, cal tenir en compte que, de la manera en que s'han creat les màquines virtuals, se'ls ha donat accés a Internet mitjançant la xarxa NAT (de l'anglès *Network Address* Translation) *default*, especificada a l'opció *--network* de la comanda *virt-install*.

El primer pas, doncs, és deslligar els dispositius IoT de la xarxa *default*. Per fer-ho, una opció es editar-los amb la comanda que s'ha especificat anteriorment (és recomanable fer-ho amb la màquina apagada):

1. *sudo virsh edit [nom_màquina]*: s'obrirà un arxiu *.xml* que conté la configuració de la màquina virtual [9].
2. Cercar el block *<interface type='network'>...<source network='default'/>...</interface>* i eliminar-lo per complert. D'aquesta manera, s'elimina la interfície de la màquina virtual que estava connectada a la xarxa *default* i que li donava accés a Internet.
3. Desar l'arxiu i sortir.

Tot seguit s'ha de crear una xarxa privada (des del servidor) per a cada node guardià, que connecti el node en qüestió amb els seus dispositius IoT associats. La creació de la xarxa es pot fer mitjançant un arxiu .xml [9] on s'ha d'especificar el nom de la xarxa (en l'exemple *private1*), el nom del pont (en l'exemple *virbr1*), l'adreça IP del pont i el rang de possibles

adreces IP dels dispositius connectats al pont. A continuació es mostra un exemple d'arxiu .xml:

```
<network>
  <name>private1</name>
  <bridge name="virbr1"/>
  <ip address="192.168.152.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.152.2" end="192.168.152.254"/>
    </dhcp>
  </ip>
</network>
```

És important no repetir el nom de la xarxa, del pont, ni de l'adreça IP del pont respecte amb altres xarxes. Un cop editat l'arxiu .xml, es segueix amb les comandes següents [9]:

1. *sudo virsh net-define [nom_arxiu].xml*: defineix la xarxa.
2. *sudo virsh net-start [nom_xarxa]*: activa la xarxa especificada.
3. *sudo virsh net-autostart [nom_xarxa]*: fa que la xarxa s'activi automàticament.

En cas d'error en la creació de la xarxa, es pot editar, desactivar o eliminar amb les comandes següents, respectivament.

sudo virsh net-edit [nom_xarxa]

sudo virsh net-destroy [nom_xarxa]

sudo virsh net-undefine [nom_xarxa]

Després de qualsevol canvi, es recomanable reiniciar la xarxa amb la comanda:

service network restart

Una vegada les xarxes privades estan creades, s'han de crear interfícies als nodes guardians i als dispositius IoT i connectar-les a la xarxa privada corresponent. Per fer-ho, es pot utilitzar la comanda *virsh edit* des del servidor, per a cada una de les màquines virtuals i afegir un contingut similar al següent:

```
<interface type='network'>
  <mac address='52:54:00:bc:2f:9c'/>
  <source network='private1'/>
  <model type='virtio'/>
```

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
</interface>

Cal destacar que l'adreça MAC ha de ser única en cada interfície, i tampoc es pot repetir entre la resta de màquines. El paràmetre *slot* ha de ser un nombre únic entre la resta d'adreces definides dins la mateixa màquina.

Després que la interfície d'una màquina estigui creada i tingui una adreça MAC, se li pot assignar una adreça IP fixa, opcionalment, editant la xarxa privada (*sudo virsh net-edit [nom_xarxa])* i afegint tants camps *host mac* com adreces IP es vulguin fixar, al camp <*dhcp*>, com es mostra en forma d'exemple a continuació:

<dhcp>
<range start="192.168.152.2" end="192.168.152.254"/>
<host mac='[adreça_mac]' name='[nom_maquina]' ip='[adreça_IP_a_fixar]'/>
</dhcp>

Finalment, és necessari entrar a cada node guardià i a cada dispositiu IoT per acabar de configurar tot l'entorn. Als dispositius IoT cal introduir la següent comanda, que defineix la ruta estàndard d'accés a Internet (gateway) [10]. Per saber l'adreça IP del node guardià corresponent, es pot teclejar al terminal (des del node guardià) la comanda *ifconfig --a* i cercar l'adreça de la interfície connectada a la xarxa privada.

sudo ip route add default via [adreça_IP_node_guardià_associat en la xarxa privada]

La comanda anterior perd el seu efecte si es reinicien les màquines virtuals. Per definir la ruta estàndard d'accés a Internet de forma permanent (dispositius IoT), s'ha d'editar el fitxer */etc/network/interfaces* i afegir el contingut següent [11]:

auto [nom_interfície_privada]
iface [nom_interfície_privada] inet static
address [IP_dispositiu_IoT]
netmask 255.255.255.0
gateway [IP_node_guardia_associat]

D'altra banda, també s'ha d'assignar un servidor de DNS. En aquest cas s'estableix el servidor de Google introduint les següents comandes al terminal del dispositiu IoT [11]:

- *sudo apt-get install resolvconf*
- *sudo gedit /etc/resolvconf/resolvconf.d/base* → introduir la línia *nameserver 8.8.8.8*
- *sudo resolvconf -u*

Als nodes guardians, cal fer un reenviament de NAT. Això es pot fer de la forma següent [12]:

1. *sudo iptables --table nat --append POSTROUTING --out-interface [nom_interficie_default] -j MASQUERADE*
2. *sudo iptables --append FORWARD --in-interface [nom_interficie_privada] -j ACCEPT*
3. *sudo bash –c 'echo 1 > /proc/sys/net/ipv4/ip_forward'*

Alternativament a la comanda 3 i per tal de tenir efecte fins i tot després de reiniciar el sistema, es pot editar l'arxiu */etc/sysctl.conf/* i introduir (o editar) la línia *sudo net.ipv4.ip_forward=1*, i tot seguit introduir la comanda *sysctl –p /etc/sysctl.conf* o */etc/init.d/procps.sh restart*.

Finalment, es té l'entorn que es descriu a la Figura 9 (per a un node guardià i un dispositiu IoT), on Pont 1 correspon a la xarxa *default*, que proporciona accés a Internet, i Pont 2 a la xarxa privada, que interconnecta el node guardià amb el seu dispositiu IoT associat.

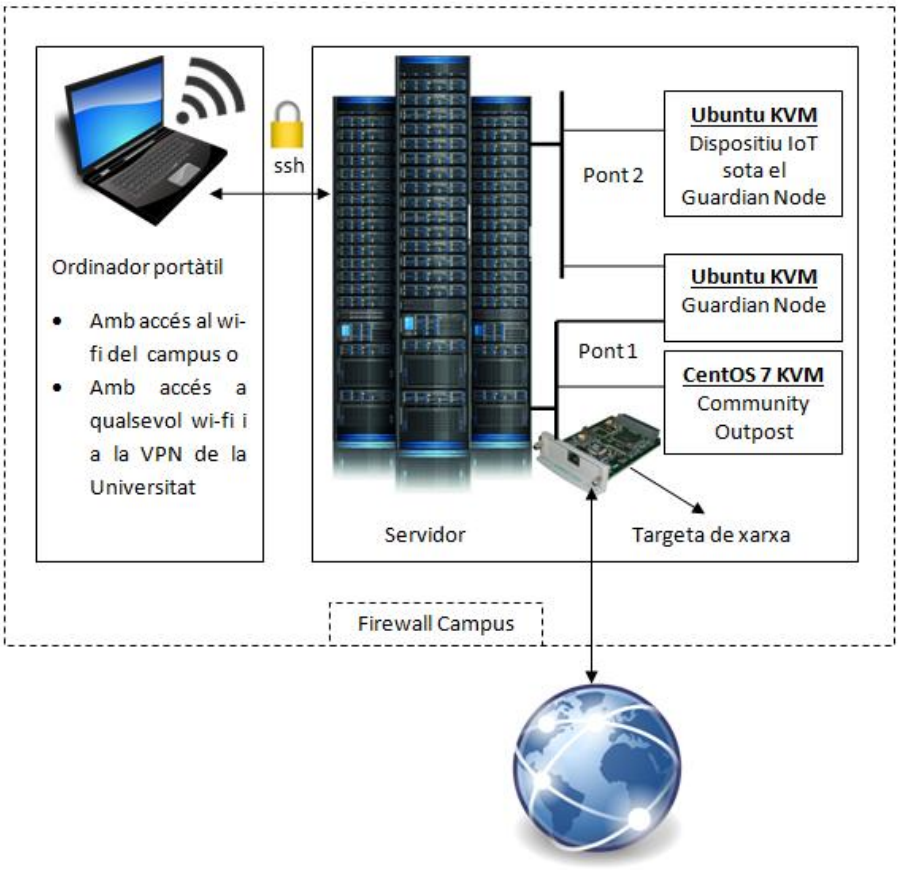


Figura 9: Esquema de la implementació bàsica de CommunityGuard al servidor

6.3. Instal·lació de CommunityGuard

La instal·lació de CommunityGuard és un procés tediós que s'ha de realitzar des del Community Outpost i des dels nodes guardians.

6.3.1. Configuració SSH

Abans de començar amb la instal·lació, però, és necessari crear claus SSH per a que la comunicació entre els nodes guardians i el Community Outpost sigui segura i autònoma, i situar-les a uns directoris concrets, per tal que els programes que s'encarreguin de la transmissió de dades les trobin. De manera semblant a com s'ha fet a l'apartat 6.1, es pot utilitzar la comanda *ssh-keygen* des de cada node guardià per crear una clau SSH, tot introduint un nom i ubicació, i sense posar *passphrase*, ja que els nodes guardians hauran d'accedir al Community Outpost autònomament. Tot seguit, es procedeix a afegir la clau a l'agent SSH amb la comanda:

eval $(ssh-agent –s)

D'altra banda, és necessari instal·lar un servidor SSH dins el Community Outpost, si és que no està instal·lat per defecte:

sudo yum –y install openssh-server

Tot seguit, cal configurar aquest servidor per a que els nodes guardians s'hi puguin connectar amb les claus creades. Per a fer això, cal copiar la part pública de les claus dels nodes guardians en un fitxer determinat, dins el Community Outpost. Per a fer la transmissió, l'opció més ràpida es utilitzar la comanda *scp* (de l'anglès *secure copy*). Aquesta comanda utilitza SSH per a fer la transmissió, de manera que per a que tingui efecte i no doni errors, s'ha de configurar el servidor SSH per no haver de necessitar les claus temporalment. Això es pot fer editant el fitxer (des del Community Outpost) ubicat a */etc/ssh/sshd_config* i editant certes línies, de forma que quedin de la següent forma:

- *PermitRootLogin without-password*: permet l'accés autònom dels nodes guardians al servidor sense la necessitat de posar una contrasenya.
- "*PermitEmptyPasswords* yes" i "*PasswordAuthentication yes"*: estableixen el mètode d'autenticació per contrasenya en comptes de per clau i permeten l'ús de contrasenyes buides.

Un cop editat i desat el fitxer, i per a que els canvis tinguin efecte, s'ha d'introduir la comanda *sudo service sshd restart*. Tot seguit, es pot realitzar la transferència de la clau pública (*id_rsa.pub* en aquest cas) des d'un node guardià al directori de l'usuari del Community Outpost, per exemple, amb la següent comanda

cd ~/.ssh/

scp id_rsa.pub [nom_usuari_Community_Outpost]@[IP_Community_Outpost]:

Un cop es té la clau pública del node guardià al Community Outpost, cal copiar-ne el contingut a un fitxer de claus autoritzades, reconegut pel servidor SSH. Per defecte, aquest fitxer es troba a *~/.ssh/authorized_keys*. En aquest cas, i després de realitzar vàries proves, s'observa que el servidor SSH al sistema operatiu CentOS no funciona correctament perquè no reconeix el fitxer de claus en la ubicació esmentada [13]. Això es deu a que CentOS pot encriptar per error el directori de l'usuari. Per solucionar aquest problema, es decideix copiar el fitxer a una altra ubicació i modificar els seus permisos:

mkdir /etc/ssh/aux

cp ~/.ssh/authorized_keys /etc/ssh/aux/

chown –R [usuari_Community_Outpost]:[usuari_Community_Outpost] /etc/ssh/aux

chmod 755 /etc/shh/aux

chmod 644 /etc/ssh/aux/authorized_keys

Després de realitzar aquest procés, l'últim pas és tornar a editar el fitxer de configuració del servidor SSH (*/etc/ssh/sshd_config*) per habilitar la connexió mitjançant clau. Les línies que s'han d'editar són les següents:

- *AuthorizedKeysFile /etc/ssh/aux/authorized_keys*: indica al servidor SSH a quin fitxer es troben les claus públiques dels nodes guardians.
- *PasswordAuthentication no*: es desactiva l'autenticació mitjançant contrasenya, forçant l'autenticació mitjançant clau.
- *#PermitEmptyPasswords yes*: es comenta aquesta línia, ja que l'autenticació mitjançant contrasenya esta desactivada.

Finalment, si es torna a introduir la comanda *sudo service sshd restart*, els nodes guardians ja estan preparats per accedir al Community Outpost mitjançant la seva clau SSH.

6.3.2. Community Outpost

Per configurar el Community Outpost, el primer pas és instal·lar les llibreries EPEL seguint un conjunt d'instruccions [14]. Tot seguit, s'han d'instal·lar *git* i *ansible* amb la comanda:

sudo yum install git ansible

Per poder executar els programes de configuració del Community Outpost, s'ha de configurar el servidor per a poder executar comandes com a superusuari sense haver de posar contrasenya. Això es pot fer teclejant pel terminal *sudo visudo* i afegint la línia *[nom_usuari] ALL=(ALL) NOPASSWD:ALL* a l'arxiu que s'obre. Abans de descarregar el codi, és necessari crear un compte d'usuari amb contrasenya a Bitbucket, i que algú dels desenvolupadors del projecte CommunityGuard proporcioni accés a aquest compte als repositoris. Aconseguit això, es pot clonar el codi en el servidor amb la comanda:

git clone https://bitbucket.org/ChaseEStewart/advnetsysfinal

El codi contingut inclou un directori amb fitxers que permeten configurar automàticament el software del Community Outpost, si aquest corre el sistema operatiu CentOS. Per executar aquests arxius es poden entrar les comandes que es mostren a continuació. S'ha d'introduir un nom d'usuari i contrasenya que s'utilitzaran en els futurs accessos a la base de dades que s'implementa al Community Outpost.

cd ~/advnetsysfinal/deploy

sudo ansible-playbook aws_main.yml --i hosts --e "user_name=[nom_usuari] extra_password=[la teva contrasenya MySQL]"

Cal destacar que els fitxers que permeten la configuració mitjançant la comanda anterior no estan del tot complets, ja que els falta la programació de la tasca periòdica que permet l'anàlisi de la informació tramesa per cada node guardià. Degut a això, s'ha de programar manualment de la següent manera:

Obrir fitxer *sudo crontab –e* → afegir la línia **/15 * * * * python /home/[Usuari_Community_Outpost]/advnetsysfinal/aws_cron.py*

La línia introduïda indica que, cada 15 minuts, totes les hores del dia i tots els dies de la setmana i de tots els mesos, s'executi el fitxer *aws_cron.py* que s'encarrega del processament de la informació. Fet això, el software del Community Outpost queda configurat.

6.3.3. Nodes guardians

Per a configurar els nodes guardians el procés a seguir és més complex, ja que cal instal·lar i configurar Snort [15] i és un procés guiat que depèn de cada implementació del CommunityGuard. Per instal·lar Snort el procés és el mateix en tots els casos, i està descrit pas a pas a l'Annex B.

Després de seguir les instruccions d'instal·lació de Snort, s'ha d'editar l'arxiu de configuració */etc/snort/snort.conf*. En aquest fitxer, per defecte, hi ha un conjunt de línies que indiquen a Snort que incloguin un conjunt d'arxius que contenen normes. Per tal de tenir el control de les normes que s'inclouen, es decideix comentar aquestes línies executant la següent comanda des del terminal:

sudo sed –i "s/include \$RULE_PATH/#include \$RULE_PATH/" /etc/snort/snort.conf

Tot seguit, s'obre l'arxiu de configuració Snort amb un editor de text qualsevol i s'editen les línies següents:

- Línia 45: A on apareix "*ipvar HOME_NET [adreça_IP]*", s'han d'introduir les adreces IP dels dispositius IoT associats al node guardià que s'estigui configurant.
- Línia 104 en endavant: Establir ubicacions d'arxius que contenen normes. Comentar (símbol # al principi) les línies que defineixin ubicacions diferents en les mateixes variables.
    - *var RULE_PATH rules*
    - *var SO_RULE_PATH so_rules*
    - *var PREPROC_RULE_PATH preproc_rules*
    - *var WHITE_LIST_PATH rules/iplists*
    - *var BLACK_LIST_PATH rules/iplists*
- Línia 119: afegir "*config policy_mode:inline*"
- Línia 159 en endavant: Editar les línies de la següent manera i especificar la mida de la cua d'anàlisi de paquets en xarxa per part de Snort, en MB:
    - *config daq: afpacket*
    - *#config daq_dir: <dir>*
    - *config daq_mode: inline*
    - *config daq_var: buffer_size_mb=[mida_cua_en_MB]*
- Línia 518: establir el nom i ubicació de l'arxiu de registre on s'emmagatzemaran les alertes generades per Snort temporalment, i el format en que es generaran:
    - *output alert_csv: [ubicació]/[nom_arxiu].log msg,proto,timestamp src,srcport,dst,dstport*
- Línia 546: incloure dos arxiu de normes, per defecte buits, que s'utilitzaran posteriorment per incloure les normes de tràfic DDoS i d'altres tràfics maliciosos per separat, segons convingui, afegint les línies següents. És possible que els dos fitxers s'hagin de crear manualment el primer cop.
    - *include $RULE_PATH/community_guard.rules*
    - *include $RULE_PATH/community_guard_ddos.rules*

Després de modificar i desar l'arxiu de configuració de Snort, es pot verificar que no hi hagi errors introduint una comanda de prova, indicant entre quines interfícies de xarxa es vol monitoritzar el tràfic:

sudo snort –T –i [interfície1]:[interfície2] –c /etc/snort/snort.conf

Un cop instal·lat i configurat Snort, cal seguir passos similars al cas del Community Outpost per obtenir la resta del software del node guardià:

- *sudo apt-get install git ansible* → Per instal·lar Git i Ansible
- *cd ~*
- *sudo git clone https://bitbucket.org/ChaseEStewart/advnetsysfinal*, fent servir el mateix compte de Bitbucket que en el cas del Community Outpost

En el cas del node guardià, també existeixen fitxers que permeten configurar automàticament el software del node. Tot i això, no es poden emprar aquests fitxers en aquest cas, ja que estan dissenyats pel sistema operatiu Debian. De manera alternativa, es poden executar les comandes següents des del terminal (Ubuntu):

- mkdir *~/advnetsysfinal/deploy/sshtunnel*
- *cd ~/advnetsysfinal/deploy/sshtunnel*
- *sudo git clone https://github.com/pahaz/sshtunnel*
- *sudo apt-get install python-setuptools*
- *python setup.py install* → Instal·la la llibreria de Python necessària per establir connexió SSH amb el Community Outpost.
- *sudo apt-get install python-configparser* → Instal·la un llibreria que implementa un llenguatge de configuració bàsic.
- *sudo apt-get install python-mysqldb*

Tot seguit, cal editar el fitxer de configuració inclòs en el codi del node guardià a *~/advnetsysfinal/bbb_user/bbb_cron_conf.cfg* de la següent manera:

- Camp *aws_ip*: introduir adreça IP del Community Outpost.
- Camp *private-key-path*: introduir la ubicació de la clau SSH del node guardià d'accés al servidor.
- Camp *log-location*: introduir la ubicació de l'arxiu de registre (.log) on Snort escriu les alertes, ubicació prèviament definida en la configuracio de Snort.
- Camp *ddos-ip-path*: introduir la ubicació de l'arxiu (.txt) on s'introduiran, a través del Community Outpost, les adreces IP que estan essent objectiu d'un atac DDoS.

- Camp *ddos-rule-path*: introduir la ubicació de l'arxiu (.rules) on es crearan les normes de bloqueig de tràfic DDoS. Aquesta ubicació es defineix prèviament en la configuració de Snort
- Camp *COusr*: introduir el nom d'usuari amb el que es treballa al Community Outpost. Serveix als nodes guardians per iniciar sessions SSH.

A més, s'ha de programar l'execució periòdica de les 3 tasques que el node guardià realitza periòdicament, explicades a l'apartat 5.2.1. De manera semblant al cas del Community Outpost, es pot introduir:

crontab –e

Afegir línia *0 * * * * /home/[Usuari_Node]/advnetsysfinal/bbb_user/bbb_ddos_cron.sh*

*Afegir línia */15 * * * * /home/[Usuari_Node]/advnetsysfinal/bbb_user/bbb_cron.sh*

La primera línia executa al minut zero de totes les hores de tots els dies de l'any un programa que s'encarrega de la cerca d'informació disponible a la llista de vigilància del Community Outpost i de la corresponent creació de normes.

La segona línia executa cada 15 minuts un programa que s'encarrega de les altres dues tasques periòdiques del node guardià. D'una banda, la transmissió d'informació sobre les alertes de tràfic maliciós detectades, a partir de l'arxiu de registre de Snort. D'altra banda, la cerca i actualització de la llista de direccions IP malicioses més recent al Community Outpost, i l'actualització de les normes del projecte de seguretat Emerging Threats de Snort.

Després de fer això, l'últim pas abans d'iniciar Snort per a que monitoritzi el tràfic és escriure normes en el fitxer *community_guard.rules*. Es poden copiar normes dels fitxers de Emerging Threats, que es troben presents al mateix directori que el fitxer, o bé introduir unes poques normes de prova.

Finalment, es pot iniciar Snort en mode IDS per a que monitoritzi el tràfic en xarxa entre dues interfícies amb les normes establertes i amb les normes DDoS que es creïn automàticament, amb la comanda següent. Cal destacar que Snort s'aturarà i es tornarà a iniciar en quant s'executi el fitxer *bbb_cron.sh*. Per a que monitoritzi les interfícies desitjades, cal editar aquest fitxer i introduir-les manualment

sudo snort –Q –c /etc/snort/snort.conf –i [interfície1]:[interfície2]

Cada cop que es segueixi el procediment anterior per crear i configurar un node guardià, s'ha de generar un hash SHA256 de l'adreça MAC del node i d'una *random nonce* que s'ha de crear i emmagatzemar al fitxer de configuració del software del node guardià. Un cop generat, s'ha d'introduir manualment a la taula *mac_address_registry* de la base de dades del Community Outpost per tal que processi les entrades provinents d'aquest node guardià.

Per fer tot això, es poden introduir un conjunt de comandes en Python en un fitxer executable amb el següent contingut, introduint l'adreça IP i usuari del Community Outpost, la ubicació del fitxer privat de la clau SSH i la ubicació del fitxer de configuració del software del node guardià (per defecte a *~/advnetsysfinal/bbb_user/bbb_cron_conf.cfg*):

```
#!/usr/bin/python
from uuid import getnode as get_mac
from sshtunnel import SSHTunnelForwarder
import random
import hashlib
import MySQLdb
with SSHTunnelForwarder(
        ('[Community_Outpost_IP]',22),
        ssh_private_key="[private_key_path]",
        ssh_username="[usuari_Community_Outpost]",
        remote_bind_address=('127.0.0.1',3306)) as server:
                my_mac = get_mac() #get MAC address
                salt_hash = random.getrandbits(64)
                mac_ip_hash = hashlib.sha256()
                mac_ip_hash.update(str(salt_hash))
                mac_ip_hash.update(str(my_mac))
                hash_val  = mac_ip_hash.hexdigest()
                con = None
                con = MySQL.connect(user='dbwriter', passwd='I!AmTheFly',
                db='adv_netsys_final',host='127.0.0.1',port=server.local_bind_port)
                cur = con.cursor()
                cur.execute("INSERT INTO mac_addr_registry (hash_val) VALUES
                (|""+self.hash_val+"\");")
                con.commit()
                f=open("/home/[nom_usuari_NG]/advnetsysfinal/bbb_user/bbb_cron_conf.cfg
", "r+")
                f.seek(9,0)
                f.write("salt-hash=%s|n" %(str(salt_hash)))
                f.close()
```

7. Proves amb Snort

Després d'haver analitzat i recreat CommunityGuard, es vol indagar en l'efecte de Snort en el rendiment de la xarxa. Les proves realitzades amb el prototip mesuren velocitats de pujada, de baixada i latència en diferents condicions de càrrega, i contrastant el cas en què Snort està actiu i el cas en què no. També s'ha valorat el consum de memòria RAM en funció de la quantitat de normes incloses. El que no s'ha fet, és, mesurar si la quantitat de normes incloses pot tenir un efecte negatiu i significatiu en el rendiment de la xarxa. En aquesta part s'experimenta amb Snort per respondre aquesta pregunta, ja que en cas que la quantitat de normes afecti al rendiment de la xarxa, la recerca d'un mètode d'optimització de normes tindria sentit.

7.1. Equipament i software

Per a fer les proves es necessiten dos dispositius configurables amb, com a mínim, dues interfícies Ethernet cadascun. Un dels dispositius ha d'executar Snort en mode IDS, monitoritzant el tràfic entre les seves dues interfícies i, per agilitzar les proves, pot tenir un script que, en executar-se, inclogui en un únic fitxer les normes Snort que es desitgin. L'altre dispositiu s'ha d'encarregar d'enviar tràfic per una de les seves interfícies, a través de l'altre dispositiu, de rebre el tràfic per l'altra interfície, i d'analitzar el rendiment de la xarxa per a cada enviament. Amb tal fi, es pot utilitzar el programa D-ITG (de l'anglès *Distributed Internet Traffic Generator*), una plataforma capaç de produir tràfic IPv4 i IPv6 replicant precisament la càrrega de treball d'aplicacions en Internet actuals, a la vegada que permet realitzar mesures en xarxa com la velocitat, el retard dels paquets o la pèrdua de paquets, per exemple. D-ITG pot generar tràfic seguint models estocàstics de la mida dels paquets o del temps entre paquets.

Com a dispositiu on s'executarà Snort s'utilitza una Raspberry Pi 2 (Figura 10) que, com a característiques més rellevants presenta:

- Processador ARM Cortex-A7 quad-core de 900MHz.
- 1GB RAM.
- 4 ports USB.
- 40 pins GPIO (de l'anglès *Global Purpose Input Output*).
- Port Ethernet.
- Ranura de targeta micro SD.

A aquest dispositiu se li afegeix un convertidor USB a Ethernet (*TRENDnet USB 3.0 to Gigabit Ethernet Adapter*, Figura 11) per aconseguir les dues interfícies Ethernet

necessàries.

L'altre dispositiu és un *Switch ADI* (*Gigabit SDN Development Kit*), que consisteix en un equip de desenvolupament que permet als manufacturers originals d'equipament avaluar, desenvolupar, i desplegar de manera ràpida noves classes de productes SDN (de l'anglès *Software-Defined Networking*). El dispositiu escollit corre *OpenSUSE* (basat en Linux) com a sistema operatiu i presenta un conjunt d'interfícies Ethernet que poden ser configurades per a enviar i rebre informació de la Raspberry Pi 2. A la Figura 12 es mostren dues imatges del mateix.

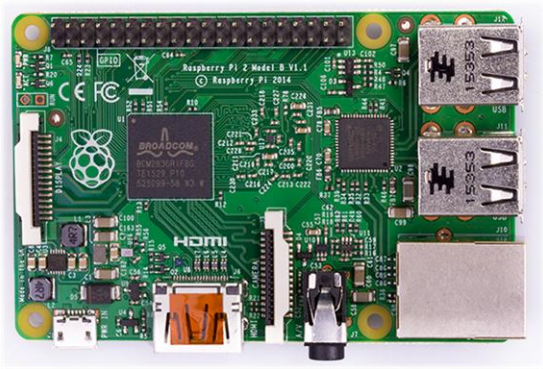


Figura 10: Raspberry Pi 2

Figura 11: TRENDnet USB 3.0 to Gigabit Ethernet Adapter

Figura 12: Gigabit SDN Development Kit

Apart d'aquests dos dispositius, són necessaris més elements que permeten configurar l'entorn per a realitzar les proves correctament. El *ADI Switch* ve amb un cable d'alimentació, diversos cables Ethernet i un cable USB a mini USB que permet el seu control mitjançant un altre dispositiu (un ordinador portàtil, per exemple), mentre que la Raspberry ve juntament amb diversos cables d'alimentació.

Per al control de la Raspberry, hi ha dues opcions: o bé utilitzar un segon convertidor de USB a Ethernet per aconseguir una tercera interfície mitjançant la qual fer una connexió SSH, o bé s'utilitza monitor amb cable HDMI i teclat. En aquest cas s'utilitza un monitor i teclat.

7.2. Preparació

Per a preparar les proves amb Snort cal configurar i connectar la Raspberry i el *ADI Switch* de forma que es puguin controlar des d'un ordinador o monitor i que tinguin accés a Internet, per tal d'instal·lar-hi el software necessari.

7.2.1. Raspberry Pi 2

El primer pas abans de configurar la Raspberry és instal·lar el sistema operatiu Debian [16] i configurar-ho de manera que es pugui accedir remotament amb SSH. Amb tal fi, és necessari disposar d'un monitor amb cable HDMI, teclat i ratolí, cable Ethernet, targeta micro SD i cable d'alimentació Raspberry.

Es comença introduint la targeta micro SD a un ordinador mitjançant un adaptador de micro SD a SD, i instal·lar el sistema operatiu amb el programa *Win32DiskImager* a partir d'una imatge. La imatge es pot descarregar en línia [17], escollint el sistema operatiu *Raspbian* (basat en Debian). Amb la imatge descarregada i el programa Win32DiskImager obert, només cal seleccionar el dispositiu micro SD (pestanya *Device*) i la imatge (pestanya *Image File*), i tot seguit seleccionar *Write*.

En finalitzar la instal·lació, es pot extreure la targeta micro SD i introduir-la a la Raspberry a

la ranura corresponent. Aleshores, la Raspberry és com un ordinador però sense les interfícies habituals com teclat, ratolí o pantalla, i s'inicia en el moment de connectar el cable d'alimentació.

Per poder controlar la Raspberry com qualsevol altre ordinador és necessari connectar-ne el monitor amb el cable HDMI, i el teclat i ratolí als ports USB. Amb aquestes connexions fetes, es visualitza al monitor l'escriptori del sistema operatiu Debian de Raspberry. Cal destacar que la Raspberry encara no té connexió a Internet, característica necessària per tal de connectar-s'hi remotament. Hi ha dues maneres d'aconseguir això. La primera, consisteix en connectar-ne un cable Ethernet directament des del router de casa. La segona, consisteix en connectar un cable Ethernet des de la Raspberry a un ordinador amb accés a Internet. En el cas de Windows, per permetre a la Raspberry l'accés a la xarxa, s'ha de fer el següent:

- Anar al Panell de Control → Xarxes i Internet → Centre de xarxes i recursos compartits.
- Seleccionar Internet, Wi-Fi (nom-xarxa-wifi-ordinador) → Propietats → Pestanya "Ús compartit" → Seleccionar "Permetre que els usuaris d'altres xarxes es connectin a través de la connexió a Internet d'aquest equip" i com a connexió de xarxa domèstica "Ethernet".
- Acceptar.

Fet això, si es reinicia la Raspberry, disposa d'accés a Internet. Per comprovar-ho, es pot obrir un terminal (des d'accessoris, al menú d'inici) i executar la comanda *ping 8.8.8.8* (servidor DNS de Google). Si els paquets de prova enviats obtenen resposta, és que es disposa d'accés a Internet.

Per tal d'accedir remotament a la Raspberry mitjançant SSH, no és necessari seguir el procés habitual basat en la creació de claus. En aquest cas, és suficient amb conèixer l'adreça IP de la Raspberry i l'usuari i contrasenya de l'usuari per defecte instal·lat en la mateixa. Per saber l'adreça IP, es pot teclejar pel terminal la comanda *ifconfig –a*. El nom d'usuari i contrasenya en aquest sistema operatiu són *pi* i *raspberry*, respectivament, per defecte.

Sabent això, es pot prescindir per a futures connexions del monitor amb cable HDMI, del teclat i del ratolí, ja que es pot obrir un terminal mitjançant una connexió SSH amb el programa PuTTY, disponible en línia [18]. *PuTTY* és una aplicació gratuïta emuladora de terminals, consola en sèrie i transmissora d'arxius en xarxa que suporta protocols en xarxa com SCP, SSH, Telnet, etc. i port sèrie. Si s'obre aquest programa, per a realitzar la connexió només cal iniciar la sessió SSH introduint l'adreça IP de la Raspberry. En realitzar la connexió, s'haurà d'introduir des del terminal que s'obri l'usuari i contrasenya associats.

Fet això, ja es pot accedir a un terminal en la Raspberry amb només un cable Ethernet. De forma opcional, es pot realitzar una connexió a escriptori remot. Per configurar-la, s'han d'introduir les comandes següents des de la Raspberry:

sudo apt-get install vim

sudo apt-get install tightvncserver

sudo apt-get install xrdp

Després d'introduir aquestes comandes, la connexió es pot realitzar, des de Windows, amb el programa *Connexió a Escriptori remot*, introduint l'adreça IP de la Raspberry.

Un cop es té accés a la Raspberry i si aquesta disposa de connexió a Internet, es pot efectuar la descàrrega i instal·lació de Snort seguint les instruccions de l'apartat 6.3. En aquest cas, quan s'editi el fitxer de configuració del mateix, també s'ha d'especificar el nom de l'arxiu on s'ubicaran les normes i incloure aquest arxiu. D'altra banda, s'han de descarregar les normes disponibles als repositoris de Snort i ubicar-les al directori adequat:

cd ~

sudo wget –q https://rules.emergingthreatspro.com/open/snort-2.9.0/emerging.rules.zip

sudo unzip –q emerging.rules.zip

Després de tenir les normes i Snort descarregats, la Raspberry ja no necessita accés a Internet per a les proves.

7.2.2. ADI Switch

Per accedir al *ADI Switch* el procés a seguir és el següent. Es connecta el cable d'alimentació per encendre el dispositiu i el cable USB a mini USB entre el mateix i un ordinador portàtil, per exemple. Des de l'ordinador portàtil, es pot fer servir el programa *PuTTY* per obrir un terminal del *ADI Switch* mitjançant la connexió en sèrie. Per realitzar la connexió, s'obre el programa *PuTTY*, és selecciona l'opció *Serial* com a tipus de connexió, i 115200 com a *Speed*. Al camp *Serial line* s'ha d'introduir *COM* seguit d'un número que depèn del port USB on s'hagi connectat el cable. Tot seguit es fa clic a *Open* i s'obre un terminal. El dispositiu *ADI Switch* té un usuari i contrasenya per iniciar sessió. En aquest cas, són *root* com a usuari i *linc* com a contrasenya.

Després d'accedir al dispositiu, se l'ha de proporcionar accés a Internet per instal·lar-hi el programa D-ITG. Per fer-ho, es connecta un cable Ethernet a una de les seves interfícies i s'introdueix manualment l'adreça IP d'aquesta interfície, editant l'arxiu

/etc/sysconfig/network/ifcfg-[nom_interfície], i introduint la línia *IPADDR=[Adreça IP]*.S'ha d'introduir una adreça vàlida segons la xarxa a la que s'hagi connectat el dispositiu. En aquest cas, la xarxa es la del campus de la Universitat de Colorado, i com a adreça IP s'escull 192.138.189.73. Tot seguit, s'estableix un servidor DNS predeterminat, introduint, per exemple, la línia *nameserver 8.8.8.8* al fitxer /etc/resolv.conf i reiniciant el sistema. Finalment, després de reiniciar el sistema, s'estableix la ruta per defecte d'accés a Internet, en aquest cas 128.138.189.1 (UCB Wireless). Això es pot fer amb la comanda:

sudo ip route add default via 128.138.189.1

Per instal·lar el programa D-ITG cal introduir les següents comandes [19]:

sudo zypper install gcc-c++

sudo wget http://www.grid.unina.it/software/ITG/codice/D-ITG-2.8.1-r1023-src.zip

sudo unzip D-ITG-2.8.1-r1023-src.zip

cd D-ITG-2.8.1-r1023/src

sudo make

En aquest punt, és possible trobar-se amb certes complicacions, degudes al sistema operatiu que empra el Switch ADI. Algunes d'aquestes complicacions es mostren a l'Annex C. Després d'haver obtingut i compilat D-ITG, el dispositiu ADI Switch ja no necessita accés a Internet.

7.3. Configuració

Un cop obtingut i instal·lat el software necessari als dos dispositius, s'han de connectar dos cables Ethernet entre ells. En aquest cas, per exemple, es connecta un cable entre la interfície *eth0* de la Raspberry Pi 2 i la interfície *eth2* del ADI Switch, i un altre cable entre les interfícies *eth1* (Raspberry) i *eth1* (ADI Switch). També s'han de realitzar les connexions necessàries per accedir als dos dispositius, comentades anteriorment (en aquest cas, les dues interfícies de la Raspberry Pi 2 estan ocupades, i es possible que es necessitin, o bé monitor i teclat, o bé un altre convertidor de USB a Ethernet).

Tot seguit, s'han de configurar les adreces IP de cada una de les interfícies que es facin servir, de forma que els dispositius es puguin reconèixer entre ells i es pugui controlar la manera en què s'envien dades amb el programa D-ITG. Les adreces IP assignades es mostren a la Taula 2. L'assignació d'adreces es pot fer de manera anàloga a l'apartat 7.2.2, només editant els fitxers */etc/sysconfig/network/ifcfg-eth1* i */etc/sysconfig/network/ifcfg-eth2* i

introduint a cadascun la línia *IPADDR=[adreça_IP_corresponent]*. Aquests canvis prenen efecte en el moment de reiniciar el dispositiu. En el cas de la Raspberry Pi 2, les adreces IP s'indiquen editant el fitxer */etc/network/interfaces*, i introduint els blocs següents [11]:

auto eth0
iface eth0 inet static
address 192.168.4.2
netmask 255.255.255.0

auto eth1
iface eth1 inet static
address 192.168.1.2
netmask 255.255.255.0

Per tal que aquests dos blocs prenguin efecte, caldria reiniciar la xarxa en el dispositiu. Abans de fer-ho, però, és necessari habilitar el reenviament de paquets, per tal de permetre que el tràfic que arribi a una interfície sigui redirigit a l'altra. Amb tal fi, s'edita l'arxiu */etc/sysctl.conf*, i s'introdueix (o es busca i modifica, si hi és) la línia *net.ipv4.ip_forward=1*. Després de fer això, es pot reiniciar la xarxa de la Raspberry amb la comanda:

sudo /etc/init.d/networking restart

Dispositiu:Interfície	**Adreça IP**
Raspberry:eth0	192.168.4.2
Raspberry:eth1	192.168.1.2
ADI:eth1	192.168.1.1
ADI:eth2	192.168.4.1

Taula 2: Assignació d'adreces IP

Després de l'assignació d'adreces IP i de l'habilitació de reenviament de tràfic per part de la Raspberry, en un principi, des del ADI Switch ja es podria forçar el tràfic per anar fins a l'adreça IP 192.168.1.1 a través de la interfície eth2, anant a través de la Raspberry (de eth0 a eth1) i tornant per la interfície eth1. Desafortunadament, el ADI Switch no és capaç d'enviar tràfic a una adreça IP pròpia a través d'un altre dispositiu. Hi ha, però, una manera d'"enganyar" al dispositiu i solucionar aquest problema, utilitzant "falses adreces IP" [20] i pre-encaminaments i post-encaminaments, de manera que en el moment d'enviar el tràfic,

es fa a una d'aquestes adreces falses i el sistema no la reconeix d'entrada com una adreça local, tot i que amb els encaminaments ho acabarà essent. Per aconseguir això, s'han d'introduir un conjunt de comandes llargues en el terminal, o executades des d'un script com el que es mostra a la *Figura 13* (veure comentaris per entendre el concepte). En aquest cas, es defineixen les adreces IP falses 192.168.1.100 per a la interfície eth1, i 192.168.4.100 per a la interfície eth2.

```
#!/bin/bash

#Reescriure la font 192.168.4.1 a 192.168.4.100 en els paquets de sortida
sudo iptables -t nat -A POSTROUTING -d 192.168.1.100 -s 192.168.4.0/24 -j SNAT --t
o-source 192.168.4.100

#Reescriure la destinacio 192.168.1.100 a 192.168.1.1 en els paquets d'entrada
sudo iptables -t nat -A PREROUTING -d 192.168.1.100 -i eth1 -j DNAT --to-destinati
on 192.168.1.1

#Reescriure la font 192.168.1.1 a 192.168.1.100 en els paquets de sortida
sudo iptables -t nat -A POSTROUTING -d 192.168.4.100 -s 192.168.1.0/24 -j SNAT --t
o-source 192.168.1.100

#Reescriure la destinacio 192.168.4.100 a 192.168.4.1 en els paquets d'entrada
sudo iptables -t nat -A PREROUTING -d 192.168.4.100 -i eth2 -j DNAT --to-destinati
on 192.168.4.1

#Forcar el trafic cap a l'adreca falsa de eth1 a traves de eth0 Raspberry
sudo ip route add 192.168.1.100 via 192.168.4.2

#Forcar el trafic cap a l'adreca falsa de eth2 a traves de eth1 Raspberry
sudo ip route add 192.168.4.100 via 192.168.1.2

#Per tal que la Raspberry sapiga com arribar a les adreces IP falses, cal activar
el proxi ARP
echo 1 | sudo tee /proc/sys/net/ipv4/conf/all/proxy_arp
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

Figura 13: Script de configuració d'adreces IP falses

Un cop executat aquest script, es poden enviar dades a les adreces 192.168.1.100 (equivalent a 192.168.1.1) i 192.168.4.100 (equivalent a 192.168.4.1), i automàticament el tràfic circularà a través de la Raspberry Pi 2, si aquesta està disponible. A la Figura 14 es mostra un esquema de la configuració establerta. Com a prova, es poden entrar les següents comandes des d'un terminal del ADI Switch:

ping 192.168.1.100 o *ping 192.168.4.100*

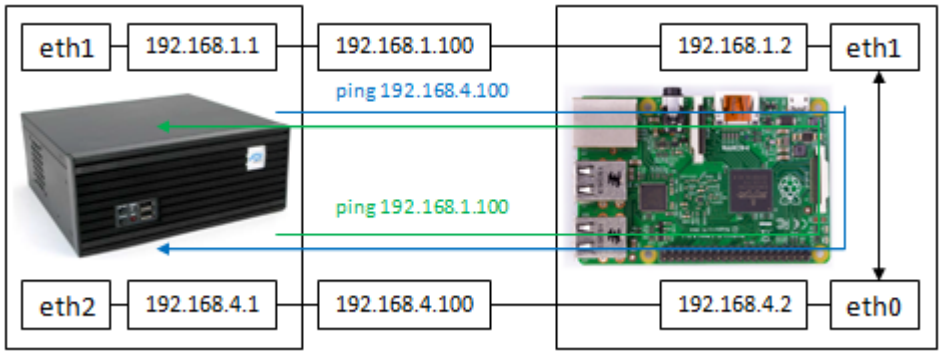


Figura 14: Flux de tràfic amb adreces IP falses

7.4. Proves

En aquests experiments es pretén determinar, en mitjana, la velocitat de transmissió de dades (*throughput*, en Kbits/s), el nombre mig de paquets enviats per segon (*packet rate*, pkt/s), la desviació estàndard de la latència (*jitter*, s) i el retard mig dels paquets (*delay*, s) en funció del nombre de normes Snort incloses i en diferents condicions. La durada de cada experiment és de 5 segons i el nombre de normes varia des de 0 (Snort no actiu) fins a 21024.

Els paquets a enviar són de tràfic TCP ja que la majoria de normes incloses estan pensades per aquest tipus de tràfic. També es consideren 3 mides diferents de paquets i que són bastant habituals: 64KB, 1500 Bytes i 64 Bytes. A més, amb el programa D-ITG es poden imposar, de manera orientativa, el nombre de paquets per segon que es pretenen enviar. Tenint en compte això, es planegen dos enviaments de dades per cada mida de paquet i nombre de normes Snort incloses: un enviament a plena velocitat (*FULL*) i un altre a mitja velocitat (*HALF*). A la Taula 3 es mostren els valors del nombre de paquets a enviar utilitzats segons cada cas.

Mida paquet	Situació	Paquets per segon a enviar
64KB	Plena velocitat	189 pkt/s
	Mitja velocitat	94 pkt/s
1500 Bytes	Plena velocitat	6562 pkt/s
	Mitja velocitat	3281 pkt/s
64 Bytes	Plena velocitat	11907 pkt/s
	Mitja velocitat	5953 pkt/s

Taula 3: Nombre de paquets per segon

Per cada enviament, és necessari introduir les comandes següents, on l'opció *–H* (mode passiu) permet treballar amb NAT, l'opció *–t* permet introduir el temps dels enviaments en milisegons, l'opció *–x* permet enregistrar estadístiques sobre els paquets rebuts, l'opció *–C* permet indicar el nombre de paquets per segon que es vol enviar, l'opció *–c* indica la mida dels paquets, i l'opció *–T* indica el protocol dels paquets a enviar [19]:

cd D-ITG-2.8.1-r1023/bin

./ITGSend –H –t 5000 –x receiver.log –C [#pkt/s] –c [Bytes/pkt] –T TCP &

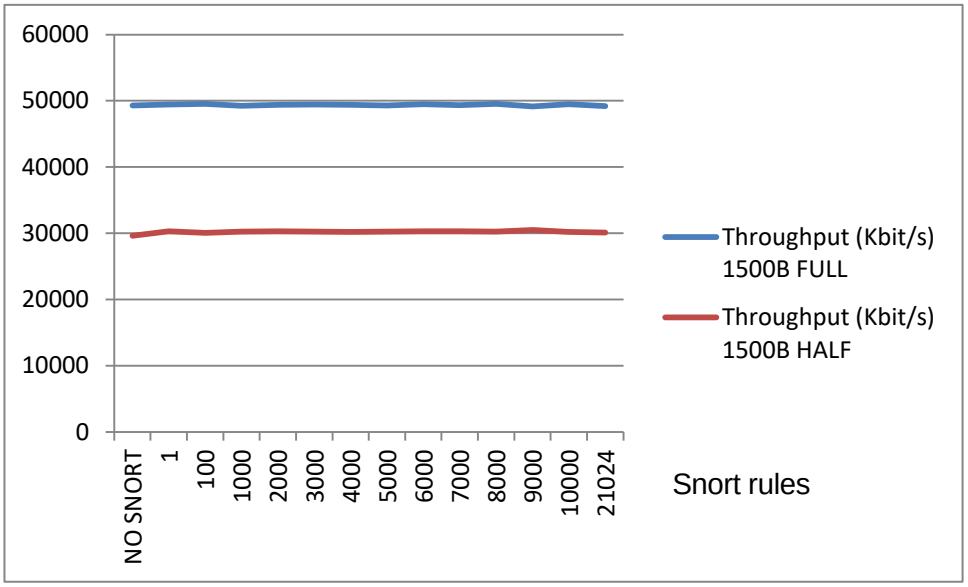
./ITGRecv –H 192.168.4.100

./ITGDec receiver.log

Els resultats obtinguts s'han introduït en 12 gràfics diferents, un per cada mida de paquet i per cada variable a mesurar. A la Figura 15 es mostren els resultats de la velocitat de transmissió, a la Figura 16 es mostren els resultats del nombre mig de paquets enviats per segon, a la Figura 17 es mostren els resultats de la desviació estàndard de la latència i a la Figura 18 es mostren els resultats del retard mig dels paquets.

Es pot apreciar que en tots els casos, excepte quan s'envien paquets de 64KB a plena velocitat, els resultats de totes les variables mesurades són pràcticament constants independentment de la quantitat de normes incloses. En el cas d'enviar paquets de 64KB a plena velocitat, es pot apreciar una diferència notable entre els resultats sense Snort actiu i amb Snort actiu (amb qualsevol quantitat de normes incloses). En el cas de la velocitat de transmissió, aquesta baixa d'uns 92500 Kbit/s a uns 82000 Kbit/s només en executar Snort, i

després es manté constant. La corba del nombre mig de paquets enviats per segon és en tots els casos proporcional a la corba de velocitat de transmissió. Les corbes de la desviació estàndard de la latència i del retard mig dels paquets presenten un augment notable en passar Snort d'estar inactiu a estar actiu i, després, es mantenen aproximadament constants com en el cas anterior (desviació de 0 a 2,7 ms i retard de 40 a 210 ms). Tenint en compte aquests resultats i els resultats de la Figura 8 obtinguts del prototip de CommunityGuard, no és necessari desenvolupar un mètode per optimitzar el conjunt de normes Snort incloses a cada node guardià.



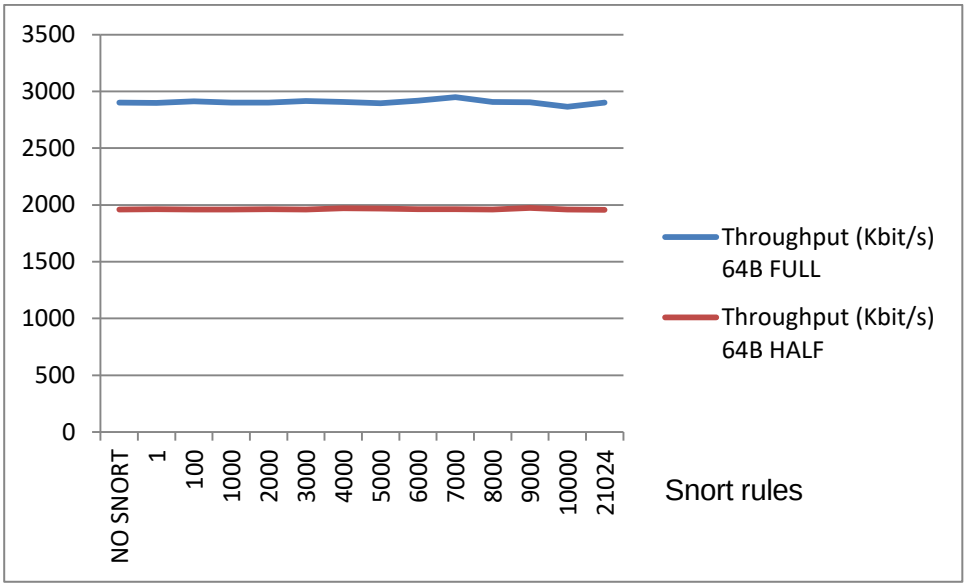
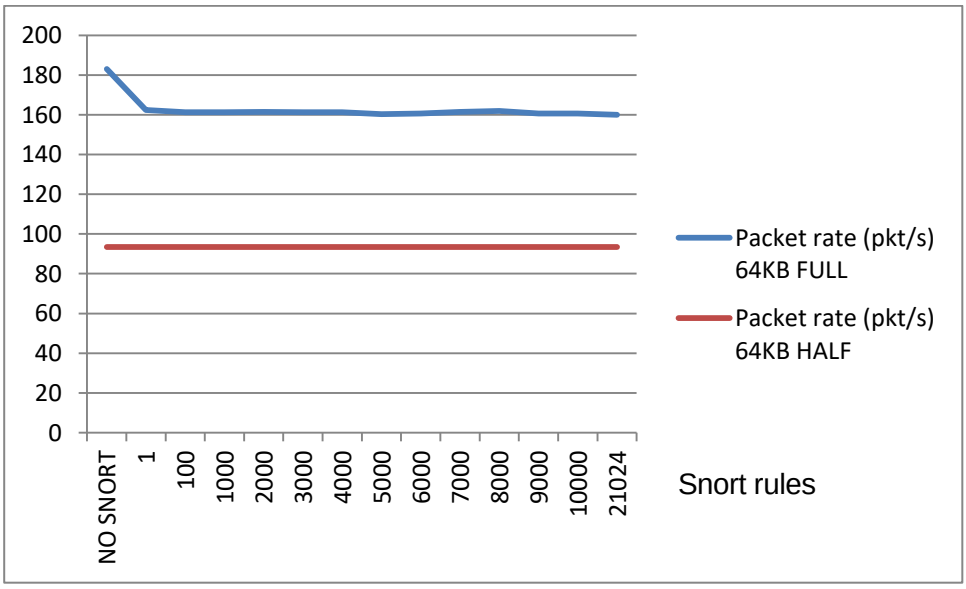


Figura 15: Velocitat de transmissió de dades



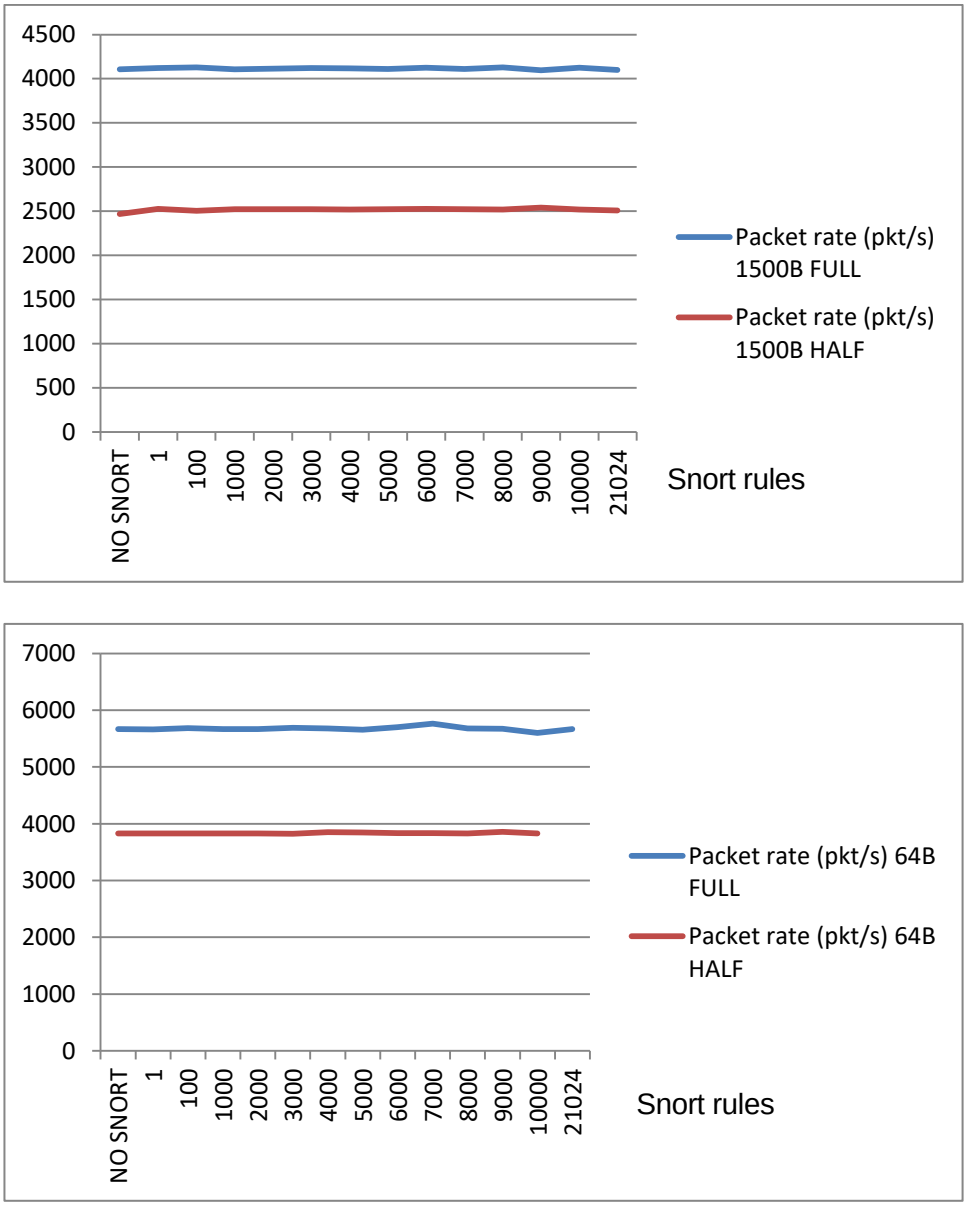
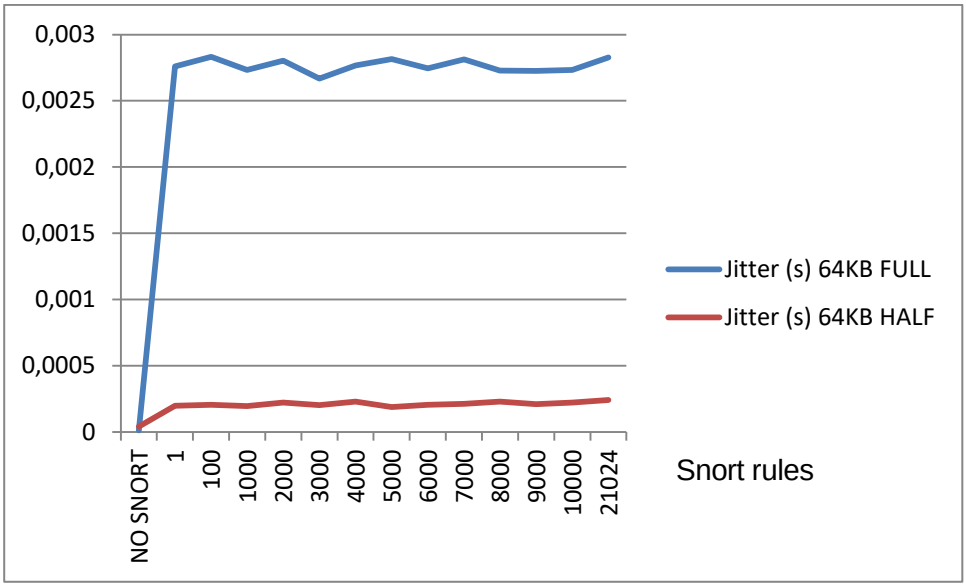
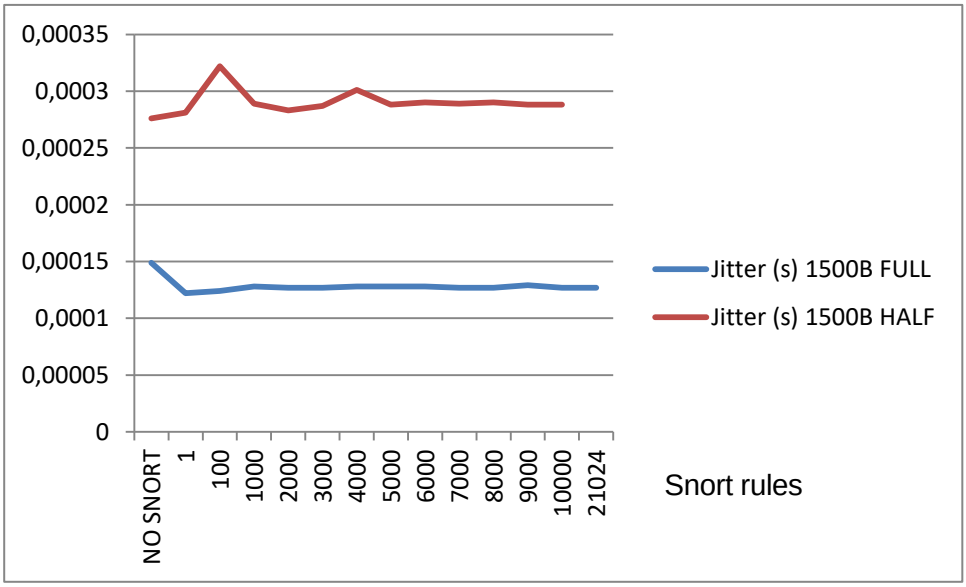


Figura 16: Nombre mig de paquets per segon enviats


0,003
0,0025
0,002
0,0015
0,001
0,0005
0
NO SNORT
1
100
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
21024
Jitter (s) 64KB FULL
Jitter (s) 64KB HALF
Snort rules


0,00035
0,0003
0,00025
0,0002
0,00015
0,0001
0,00005
0
NO SNORT
1
100
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
21024
Jitter (s) 1500B FULL
Jitter (s) 1500B HALF
Snort rules

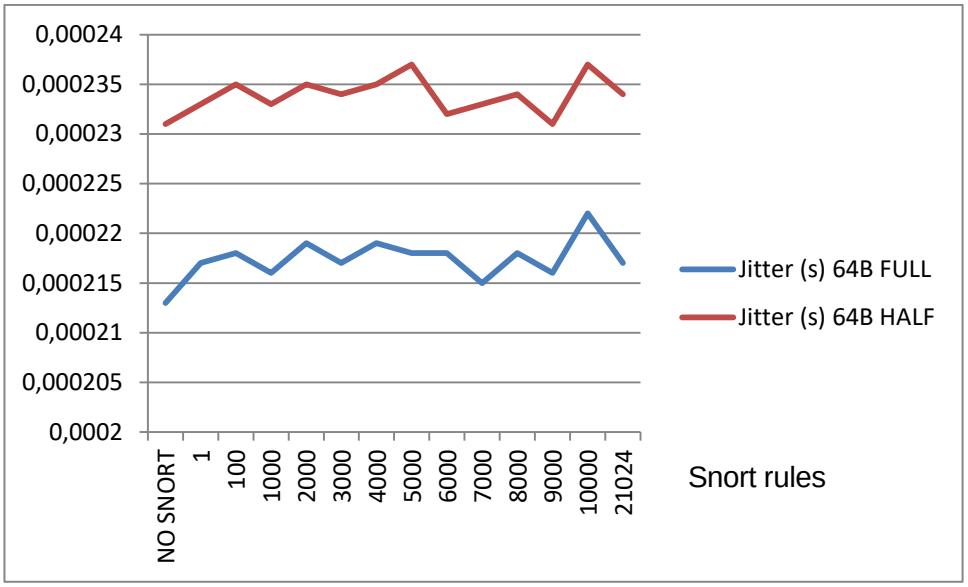


Figura 17: Desviació estàndard de la latència

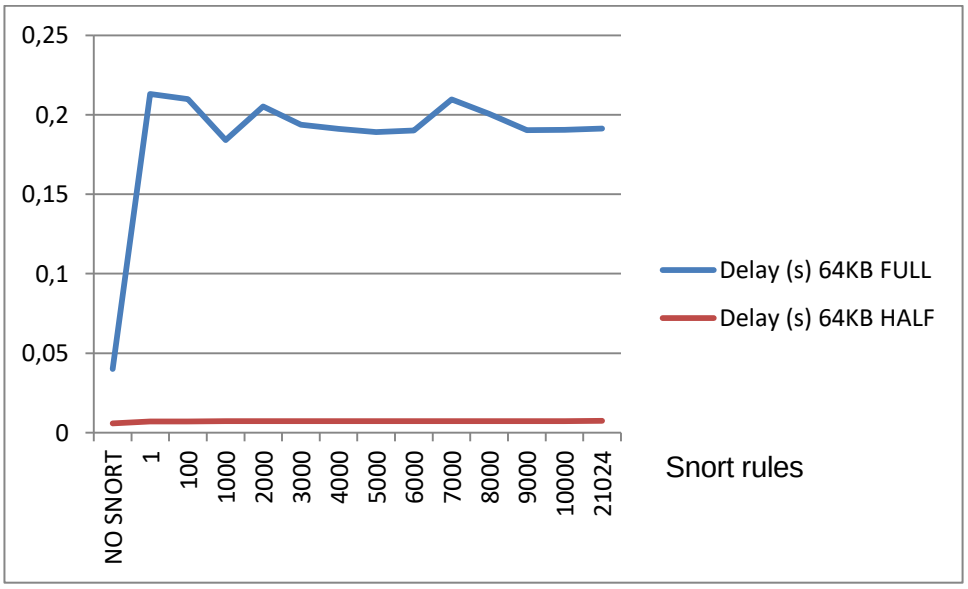


Figura 18: Retard mig dels paquets

8. Ampliació de CommunityGuard

Fins aquest apartat s'ha desenvolupat un sistema col·laboratiu de seguretat que actua a grups de xarxes amb diversos dispositius IoT, i que es basa en un conjunt de normes de seguretat per tal d'alertar o bloquejar tràfic sospitós. Aquest fet fa que l'efectivitat del sistema per classificar correctament els fluxos de dades (en maliciós o no maliciós) depengui de la qualitat de les normes utilitzades als nodes guardians, ja que no són perfectes i poden generar falses alarmes (alertant de tràfic sospitós que no és maliciós) o permetre tràfic indesitjat.

D'altra banda, un dels grans contribuïdors al risc de futurs atacs es el fet que gran part dels dispositius IoT presenten debilitats conegudes àmpliament que els fan vulnerables a l'explotació i control per part d'atacants remots. Encara pitjor, els venedors d'aquests dispositius freqüentment provenen de la indústria del hardware, però poden mancar d'habilitats o recursos per desenvolupament de software i sistemes de seguretat. Com a resultat d'això, els manufacturers de dispositius IoT poden arribar a crear dispositius que són extremadament difícils d'assegurar.

Tenint en compte les limitacions anteriors, es pretén desenvolupar un sistema de supervisió i control de les accions executades pels dispositius IoT. La majoria de dispositius IoT presenten en moltes ocasions sistemes operatius comuns, i no tenen per què disposar de la flexibilitat completa d'un sistema operatiu estàndard, fet que redueix la complexitat del sistema.

El sistema manté el concepte de node guardià, amb la diferència que l'únic requisit és que aquest tingui accés a tots els dispositius de la xarxa. L'objectiu és que cada dispositiu IoT, abans d'executar una certa "funció", hagi de sol·licitar permís al node guardià. La resposta proporcionada pel node guardià pot ser recordada pel dispositiu IoT per tal de minimitzar les sol·licituds entre node guardià i dispositiu IoT. Tenint en compte que aquesta resposta, a priori, ha d'ésser proporcionada per alguna persona, no es descarta la possibilitat d'incloure un gestor central semblant al Community Outpost, a través del qual una persona pot supervisar i controlar les accions que es duen a terme a un conjunt de xarxes amb els seus corresponents nodes guardians.

En els apartats següents s'implementen i es valoren un conjunt d'alternatives per aquest sistema.

8.1. Alternativa 1: LD_PRELOAD (Linux)

8.1.1. Procediment d'embolcall d'una funció

A Linux, les crides a sistema (o *system calls*) no son invocades directament, sinó mitjançant *wrapper functions* o funcions "d'embolcall" (funcions que criden a una altra funció original, i realitzen alguna tasca addicional) a *glibc* (La llibreria GNU C, coneguda comunament com *glibc*, és la implementació de la llibreria estàndard C de GNU Project. Malgrat el seu nom, també suporta directament C++ i, indirectament, altres llenguatges de programació). El *wrapper* glibc, en el fons, només copia arguments i el nombre únic de crida de sistema als registres on el kernel els espera, entrant a mode kernel i establint el *errno* (element de l'arxiu de capçalera *errno.*h a la llibreria estàndard de C que serveix per a reportar i recuperar condicions d'error) si la crida a sistema retorna un nombre d'error.

Existeixen diverses possibilitats per sobreescriure aquestes funcions i afegir codi propi a ser executat abans o després de l'execució de la funció original (concepte d'embolcall d'una funció). Una d'elles es utilitzar *LD_PRELOAD* [21]. *LD_PRELOAD* és una variable de Linux tipus *shell environment* (variable global a tot el sistema) en la qual es pot carregar una ubicació a una llibreria compartida, per tal que aquella llibreria sigui carregada abans que qualsevol altra llibreria.

Això permet, d'una banda, crear una funció amb el mateix nom que una altra predefinida a una llibreria, de manera que quan s'executi un programa que faci servir la funció, executarà la funció que s'ha creat. En aquest cas, a la funció que es creï, es vol executar (o no) la funció original. Per poder fer això, s'ha d'utilitzar la interfície de programació del "enllaçador dinàmic". Per entendre el concepte, a la Figura 19 es mostra un exemple d'aplicació que s'explica a continuació.

inspect_open.c:

```c
#define _GNU_SOURCE
#include <dlfcn.h>

typedef int (*orig_open_f_type)(const char *pathname, int flags);

int open(const char *pathname, int flags, ...)
{
    /* Some evil injected code goes here. */

    orig_open_f_type orig_open;
    orig_open = (orig_open_f_type)dlsym(RTLD_NEXT,"open");
    return orig_open(pathname,flags);
}
```

Figura 19: Exemple de wrapper function [21]

En aquest exemple, es pretén sobreescriure la funció *open*, que serveix per obrir un arxiu i obtenir el seu descriptor d'arxiu (indicador abstracte per accedir a un arxiu o un altre recurs

d'entrada/sortida) . La funció que sobreescriu només executa la funció original, però podria realitzar altres tasques addicionals si s'introdueixen tasques al seu interior (línies 8 a 12). Per exemple, es podria incloure l'arxiu *stdio.h* a dalt de tot, i introduir al cos de la funció la línia següent, que imprimeix per la pantalla un missatge indicant que s'ha usat aquesta funció per obrir un cert arxiu:

printf("The victim used open(...) to access '%s'!!!\n",pathname);

L'arxiu *dlfcn.h* (línia 2) es necessita per la funció *dlsym* (línia 11), mentre que la primera línia indica al compilador que habiliti una característica no estàndard que permet l'ús de *RTDL_NEXT* a *dlfcn.h* (línia 11). La comada *typedef* (línia 4) crea un àlies a un punter a una funció, i exactament els mateixos arguments que la funció original open (el nom de l'àlies és *orig_open_f_type*, el qual s'utilitza més endavant). Les tres línies presents al cos de la funció (10 a 12) fan el següent:

- Crear un nou punter a funció *orig_open* que apuntarà a la funció original *open*.
- Per trobar l'adreça de la funció *open* original, es demana a *dlsym* que trobi la següent funció "open" a la pila de biblioteques dinàmica.
- Finalment, es crida la funció (passant els mateixos arguments que han estat passats a la funció *open* "falsa") i es retorna el valor que retorni la funció original.

Per posar a la pràctica l'exemple, cal compilar el fitxer (*.c*) com una llibreria compartida, amb les següents banderes o *flags*:

gcc –shared –fPIC inspect_open.c –o inspect_open.so –ldl

S'afegeix *–ldl* per lligar la llibreria compartida a *libdl*, que proveeix la funció *dlsym*. Un cop es disposa de la llibreria, el següent pas es executar qualsevol programa que faci servir la funció *open* carregant a *LD_PRELOAD* la llibreria creada:

LD_PRELOAD=$PWD/inspect_open.so [nom_programa]

Com a resultat, el programa executarà l'embolcall de la funció *open* a cada crida a *open* que intenti fer.

8.1.2. Prototip

Amb aquest concepte de base, s'implementa un sistema de prova format per un node guardià i un dispositiu IoT, l'arquitectura del qual es mostra a la Figura 20. Al dispositiu IoT hi ha una llibreria compartida (*wrap_random5.so*) que realitza l'embolcall de la funció *rand()* (funció que genera nombres aleatoris),de manera que quan s'executa un programa que fa servir la funció *rand()* precarregant la llibreria compartida en qüestió, el dispositiu realitza una sol·licitud de la següent manera:

- Contacta amb un servei (programa en c) present al mateix dispositiu IoT (*service*) i

envia la paraula clau "*rand"* mitjançant un *Unix socket* [22], i resta a l'espera de rebre una resposta per part del servei. Un *socket* és un mitjà de comunicació en dos sentits, que pot ésser utilitzat per a comunicació en una gran varietat de "dominis". En aquest cas, els *sockets* de *Unix* poden ser emprats per a comunicació entre processos del mateix sistema *Unix*, i el socket es un arxiu compartit a l'ordinador en qüestió.

- El servei rep la paraula clau *rand*, i l'envia al node guardià mitjançant un *Internet socket* [23]. Un *Internet socket* és un altre tipus de *socket* amb utilitat en el domini en xarxa (IP), en particular pel protocol de control de transmissió (TCP).
- En el moment de rebre la paraula clau *rand*, un programa en c al node guardià (*server6_GN1)* construeix la sol·licitud concatenant l'adreça IP del dispositiu IoT i la paraula clau (en aquest cas "192.168.152.51rand").
- Després, cerca la sol·licitud en qüestió a dos fitxers diferents. Un d'ells conté una llista d'accions autoritzades i l'altre d'accions denegades prèviament.
    - Si no troba la sol·licitud a cap dels dos fitxers, indica per pantalla que ha arribat una nova sol·licitud, i si es vol acceptar o no. Per acceptar, l'usuari només ha d'introduir la lletra "y", i per denegar, la lletra "n". Tot seguit el programa escriu la sol·licitud al fitxer corresponent per recordar la decisió presa.
    - Si troba la sol·licitud a la llista d'accions autoritzades, s'accepta directament.
    - Si troba la sol·licitud a la llista d'accions denegades, es denega directament.
- Per acceptar o denegar la sol·licitud, el node guardià envia al dispositiu IoT "y" o "n", respectivament, a través del *Internet socket*.
- Quan el dispositiu IoT ho rep, el servei transforma aquests valors en "1" o "-1", respectivament, i els retorna com a resposta a través del *Unix socket*. El dispositiu IoT també recorda les respostes rebudes per tal de minimitzar el flux de dades. En el cas que hi hagi un error en l'enviament de la resposta, el servei retorna un "0" al programa original i s'avorta l'execució de la funció, sense recordar aquesta acció.

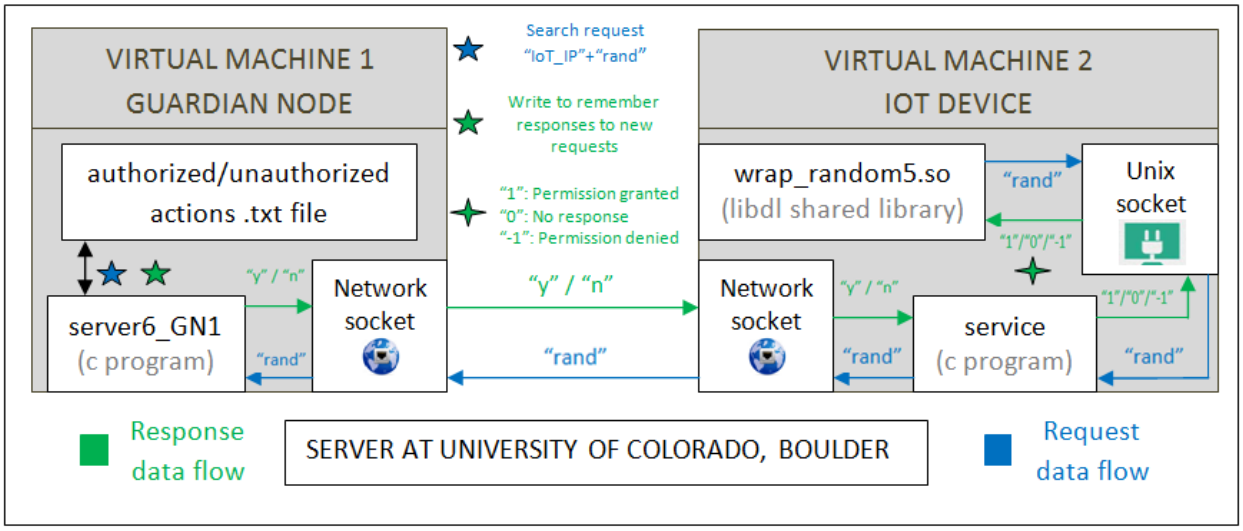


Figura 20: Arquitectura del prototip (LD_PRELOAD)

Cal destacar que prèviament a aquest procés, el node guardià i el dispositiu IoT han de tenir els processos *server6_GN1* i *service* actius i connectats entre ells. El procediment per provar el prototip és el següent (veure codi a l'annex D):

- Editar els fitxers font *server6_GN1.c* i *service.c* per introduir l'adreça IP del node guardià a la línia que comença per *#define GN_IP_ADDRESS*. Al fitxer *server6_GN1.c* introduir la ubicació dels fitxers d'accions autoritzades/denegades (*#define GOOD_FILE_PATH* i *#define BAD_FILE_PATH*). Als fitxers *service.c* i *wrap_random5.c* introduir la ubicació del *Unix socket* (*#define UNIX_SOCK_PATH*).
- Compilar els programes executant les següents instruccions:
  - *gcc server6_GN1.c –o server6_GN1* (des del node guardià)
  - *gcc service.c –o service* (des del dispositiu IoT)
  - *gcc –shared –fPIC wrap_random5.c –o wrap_random5.so –ldl* (des del dispositiu IoT)
  - *gcc random_nums.c –o random_nums* (des del dispositiu IoT). És un programa creat per executar la funció *rand()* 10 cops i imprimir els resultats per la pantalla.
- Des del node guardià, executar el programa *server6_GN1*.
- Des del dispositiu IoT, executar el programa *service*.
- Des del dispositiu IoT, executar la el programa *random_nums* precarregant la llibreria *wrap_random5.so*:

  LD_PRELOAD=$PWD/wrap_random5.so random_nums

A les Figura 21 i Figura 22 es mostren els resultats obtinguts en seguir aquest procediment
(s'ha executat el programa *random_nums* dos cops per verificar que es recorda l'acció).

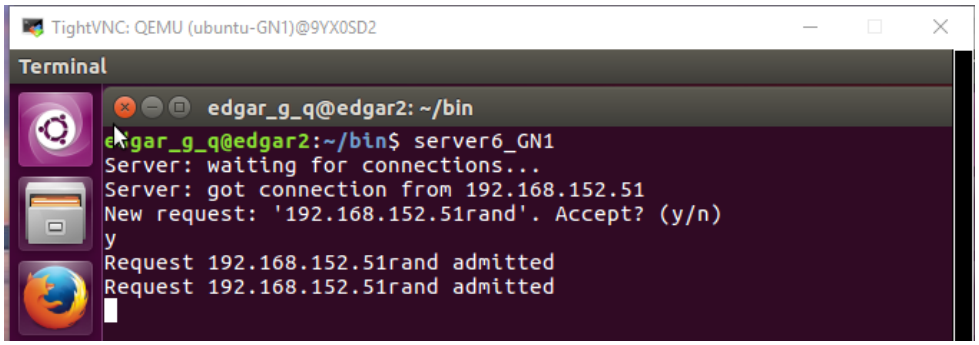


Figura 21: Resultats al node guardià (LD_PRELOAD)

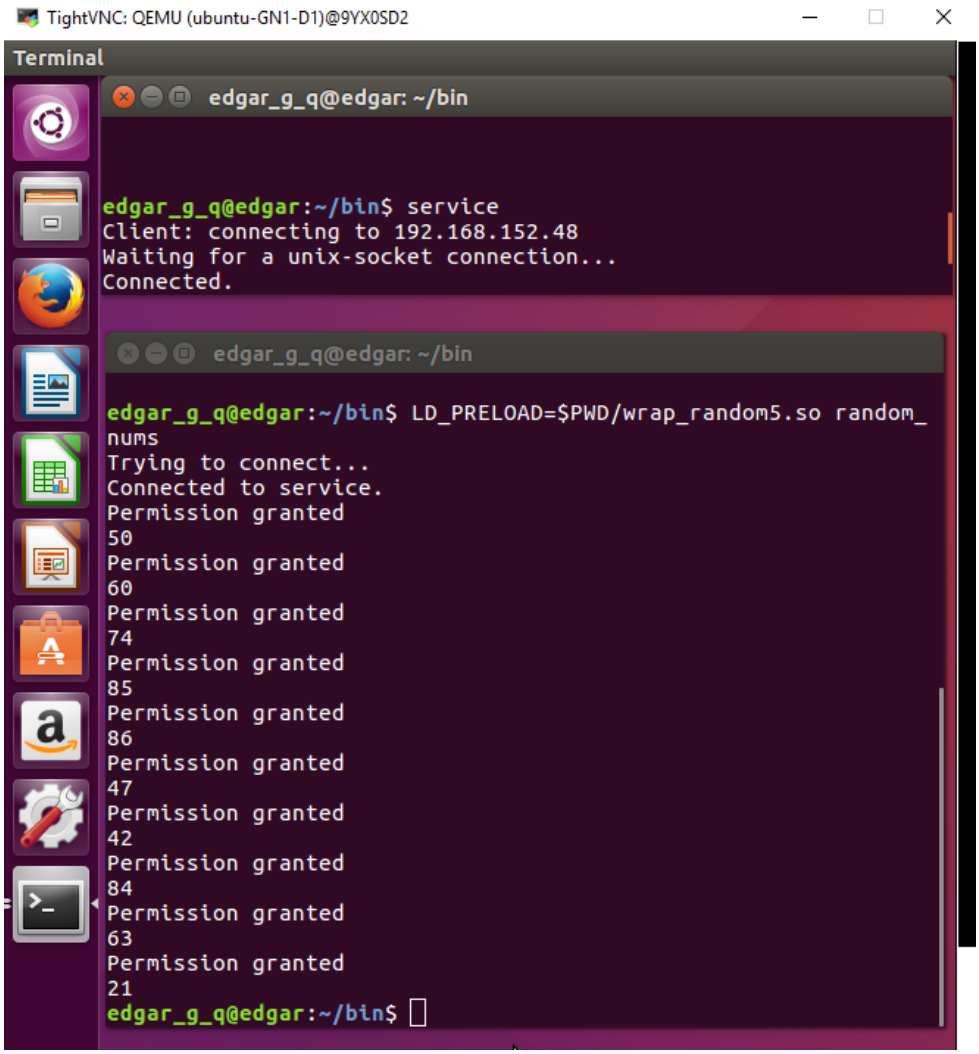


Figura 22: Resultats al dispositiu IoT (LD_PRELOAD)

8.1.3. Valoració

Mitjançant la variable d'entorn de Linux *LD_PRELOAD* i la funció *dlsym [26]*, ha estat possible crear un sistema de seguretat per tal que qualsevol acció nova dins un dispositiu IoT hagi de rebre autorització per part del node guardià.

D'una banda, *LD_PRELOAD* és una característica de l'enllaçador dinàmic i està disponible a la majoria de sistemes Unix i, per tant, permet que el sistema s'apliqui a una bona varietat de sistemes operatius.

D'altra banda, la funció *dlsym* pot obtenir l'adreça de qualsevol símbol definit a través d'un objecte accessible a través d'una crida a *dlopen [27]*, i *dlopen* és una funció que pot fer que un arxiu objecte executable, especificat per un paràmetre *file* (string de text), estigui disponible per al programa que crida la funció. El tipus d'arxius elegibles per aquesta operació són típicament llibreries compartides, arxius reubicables o programes.

Els símbols introduït per *dlopen* i disponibles a través de *dlsym* són, com a mínim, aquells que són exportats com a símbols d'abast global per l'objecte. Típicament, dits símbols haurien de ser aquells en els que es va especificar, en codi font C (per exemple), un enllaç extern (*extern linkage*).

Tenint en compte tot això, es pot afirmar que el sistema creat dóna una cobertura prou àmplia quan a sistemes operatius i funcions susceptibles de ser interceptades, però podria arribar a necessitar cobrir accions més bàsiques en el cas d'un sistema operatiu molt més senzill i simplificat.

8.2. Alternativa 2: Kprobes (Kernel module)

8.2.1. Procediment d'embolcall d'una funció

Una altra possibilitat per a realitzar un embolcall de crides a sistema es *kprobes* ("sondes" al Kernel) [24]. *Kprobes* és una eina del Linux Kernel que permet irrompre en qualsevol rutina del kernel i obtenir informació de manera no disruptiva. Pot realitzar la irrupció a pràcticament qualsevol adreça del codi kernel, especificant una rutina a ser executada quan s'arriba al punt en qüestió. Hi ha tres tipus diferents de sondejos: *kprobes*, *jprobes* i *kretprobes*. Una *kprobe* pot ésser insertada a virtualment qualsevol instrucció al kernel. Una *jprobe* és inserida a l'entrada d'una funció kernel, i proveeix accés als arguments de la funció. Una *kretprobe* actua quan una determinada funció retorna. En aquest cas s'utilitza *jprobes*, doncs es vol que abans d'executar la rutina, el dispositiu IoT demani permís al node

guardià.

Una *jprobe* està implementada usant una *kprobe* que és ubicada al punt d'entrada d'una funció. Quan una *kprobe* és enregistrada, *kprobes* realitza una còpia de la instrucció sondejada i reemplaça els primers bytes de la instrucció sondejada amb una instrucció tipus punt d'interrupció (breakpoint). Quan la CPU arriba a aquesta instrucció, una irrupció te lloc, els registres de la CPU es desen, i el control passa a *kprobes*. Kprobes executa la "pre-rutina" associada amb el *kprobe*, passant a la rutina les adreces de l'estructura *kprobe* i els registres desats. Després, *kprobes* desglossa en passos senzills la seva copia de la instrucció sondejada. Tot seguit, *kprobes* executa la "post-rutina", si n'hi ha, associada al *kprobe*. L'execució continua amb la instrucció després del punt de sondeig.

Per posar a la pràctica aquest mètode és necessari crear un mòdul de Linux kernel. Els mòduls són peces de codi que poden ser carregades i descarregades del kernel segons es necessiti, i estenen la funcionalitat del kernel sense la necessitat de reiniciar el sistema. Abans d'això, però, cal establir unes banderes (*flags*) de configuració per utilitzar *kprobes* (cercar al fitxer ubicat a */usr/src/linux[versió_kernel]/.config*.). Les banderes que s'han d'activar són *CONFIG_KPROBES, CONFIG_ MODULES, CONFIG_ MODULE_UNLOAD, CONFIG_KALLSYMS* i *CONFIG_KALLSYMS_ALL*.

Per crear el mòdul i provar *jprobes* es pot editar un fitxer en llenguatge C amb el contingut que es mostra a la Figura 23. La funció *init_module* és la que s'executa al moment d'inserir el mòdul, mentre que la funció *cleanup_module* és la que s'executa en eliminar el mòdul. A l'estructura *my_jprobe* s'indica quina és la rutina que s'executarà quan es vulgui executar la funció a sondejar, que en aquest cas és *_do_fork*, una funció que es crida cada cop que es crea o es deriva un procés nou. La funció *jdo_fork* és la rutina en qüestió.

El procés que té lloc quan s'insereix el mòdul és el següent. S'executa la funció *kallsyms_lookup_name* per a cercar l'adreça de la instrucció *_do_fork*. Aquesta instrucció és una de les moltes que es troben al fitxer */proc/kallsyms* (fitxer que conté símbols de mòduls carregats dinàmicament i codi estàtic). Després d'obtenir l'adreça, intentar enregistrar el jprobe amb la funció *register_jprobe*, i comprova que no hi hagi errors. Durant el procés, escriu missatges informatius al fitxer */var/log/syslog* amb la funció *printk*.

Mentre el mòdul està actiu, si en cap moment qualsevol programa crida directa o indirectament la funció *_do_fork*, la rutina *jdo_fork* s'executarà, indicant amb *printk* que s'executarà *_do__fork* (i amb quins paràmetres) i retornant tot seguit a la execució de la funció original. Quan s'elimina el mòdul, s'anul·la el registre de la "sonda" introduït a l'inici.

```c
/*jprobe-example.c */
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/uio.h>
#include <linux/kprobes.h>
#include <linux/kallsyms.h>

/*
 * Jumper probe for do_fork.
 * Mirror principle enables access to arguments of the probed routine
 * from the probe handler.
 */

/* Proxy routine having the same arguments as actual do_fork() routine */
long jdo_fork(unsigned long clone_flags, unsigned long stack_start,
              struct pt_regs *regs, unsigned long stack_size,
              int __user * parent_tidptr, int __user * child_tidptr)
{
        printk("jprobe: clone_flags=0x%lx, stack_size=0x%lx, regs=0x%p\n",
               clone_flags, stack_size, regs);
        /* Always end with a call to jprobe_return(). */
        jprobe_return();
        /*NOTREACHED*/
        return 0;
}

static struct jprobe my_jprobe = {
        .entry = (kprobe_opcode_t *) jdo_fork
};


    int init_module(void)
    {
            int ret;
            my_jprobe.kp.addr = (kprobe_opcode_t *) kallsyms_lookup_name
    ("_do_fork");
            if (!my_jprobe.kp.addr) {
                    printk("Couldn't find %s to plant jprobe\n", "_do_fork");
                    return -1;
            }

            if ((ret = register_jprobe(&my_jprobe)) <0)
            {
                    printk("register_jprobe failed, returned %d. "
                           "trying dereferenced address\n", ret);
                    my_jprobe.kp.addr = *((kprobe_opcode_t  **)my_jprobe.kp.addr);
                    if ((ret = register_jprobe(&my_jprobe)) <0)
                    {
                            printk("register_jprobe failed again, returned %d\n",
                                    ret);
                            return -1;
                    }
            }
            printk("Planted jprobe at %p, handler addr %p\n",
                   my_jprobe.kp.addr, my_jprobe.entry);
            return 0;
    }

    void cleanup_module(void)
    {
            unregister_jprobe(&my_jprobe);
            printk("jprobe unregistered\n");
    }
    MODULE_LICENSE("GPL");
```

Figura 23: Exemple d'aplicació de jprobes [24], adaptat Kernel 4.8.0-49-generic

Per poder provar el mòdul cal crear un arxiu de compilació i introduir un conjunt de comandes. L'arxiu de compilació és pot anomenar *Makefile* i contenir el següent:

```
obj-m := jprobe-example.o

KDIR := /lib/modules/$(shell uname –r)/build

PWD := $(shell pwd)

default:

    $(MAKE) –C $(KDIR)  SUBDIRS=$(PWD) modules

clean:

    rm –f *.mod.c *.ko *.o
```

Un cop editat l'arxiu, es pot construir el mòdul introduint la comanda *make* al terminal. Tot seguit, per insertar o eliminar el mòdul s'utilitzen les comandes següents, respectivament:

```
sudo insmod jprobe-example.ko

sudo rmmod jprobe-example.ko
```

8.2.2. Prototip

De manera anàloga a l'apartat 8.1.2, amb el concepte anterior de base, s'implementa un sistema de prova format per un node guardià i un dispositiu IoT, l'arquitectura del qual es mostra a la Figura 24. El sistema té com a objectiu obtenir informació sobre les crides al sistema de *_do_fork* al dispositiu IoT i fer arribar aquesta informació a la pantalla del node guardià. Per aconseguir-ho, el procés que té lloc és el següent:

- Al dispositiu IoT hi ha un mòdul (*base_mod_3.ko*) que, en ésser inserit, sondeja la crida a sistema *_do_fork* per obtenir la informació desitjada i resta a l'espera de rebre un missatge d'identificació per part d'un servei (programa en c, *service3*). En aquest cas, aquesta comunicació es realitza mitjançant un *Netlink socket* [25]. Un *Netlink socket* és un tipus de socket especial que permet la transmissió d'informació entre el Kernel i processos de l'espai de l'usuari.
- Quan el mòdul rep el missatge per part del servei, obté i emmagatzema el seu nombre identificador de procés (*PID*), el qual li permet adreçar-se a aquest servei i enviar-li la informació desitjada.
- Si en qualsevol moment es deriva o es crea un nou procés al dispositiu IoT, la crida a *_do_fork* serà interceptada i s'enviarà un missatge informatiu al servei *service3*. Aquest, en rebre el missatge, el reenviarà al node guardià mitjançant un *Internet socket*.

- Finalment, el node guardià, després de rebre el missatge del dispositiu IoT, imprimirà per la pantalla un missatge format per l'adreça IP del dispositiu IoT en qüestió i pel missatge informatiu que ha rebut (*server7_GN1*).

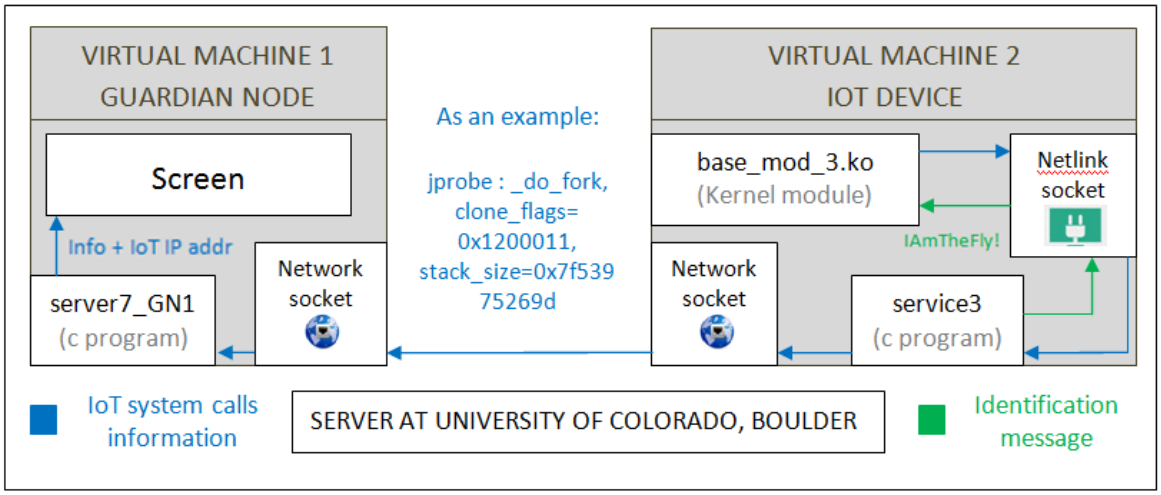


Figura 24: Arquitectura del prototip (Kprobes)

Les condicions per a que el procés anterior funcioni correctament són que el node guardià tingui el procés *server7_GN1* actiu, i que el dispositiu IoT tingui inserit el mòdul *base_mod_3.ko* i activi el procés *service3* per a que realitzi la connexió amb el node guardià i el mòdul anterior. El procediment per provar el prototip és el següent (veure codi a l'annex E):

- Editar els fitxers font *server7_GN1.c* i *service3.c* per introduir l'adreça IP del node guardià a la línia que comença per *#define GN_IP_ADDRESS*.
- Compilar els programes executant les següents instruccions:
  - *gcc server7_GN1.c –o server7_GN1* (des del node guardià)
  - *gcc service3.c –o service3* (des del dispositiu IoT)
  - *make* (compila el mòdul *base_mod_3.c* al dispositiu IoT)
- Des del node guardià, executar el programa *server7_GN1*.
- Des del dispositiu IoT, inserir el mòdul *base_mod_3.ko* amb la comanda *sudo insmod base_mod_3.ko*.
- Des del dispositiu IoT, executar el programa *service3*.
- Per a que hi hagi una crida a *_do_fork*, no cal crear cap programa en especial. És

suficient amb realitzar qualsevol acció dins la màquina virtual del dispositiu IoT (Obrir un nou terminal, per exemple).

A les Figura 25 i Figura 26 es mostren els resultats obtinguts en seguir aquest procediment.

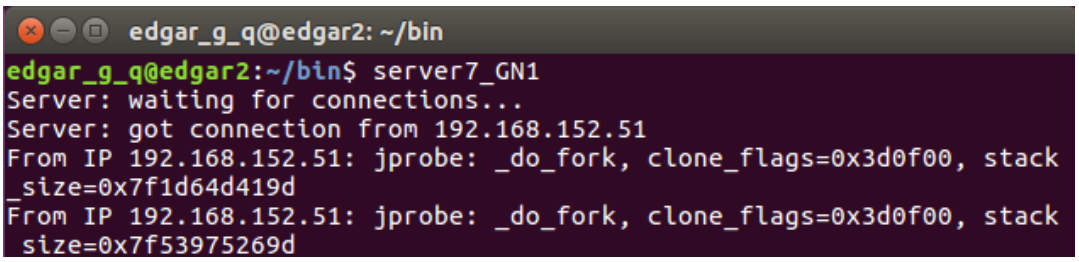


Figura 25: Resultats al node guardià (Kprobes)

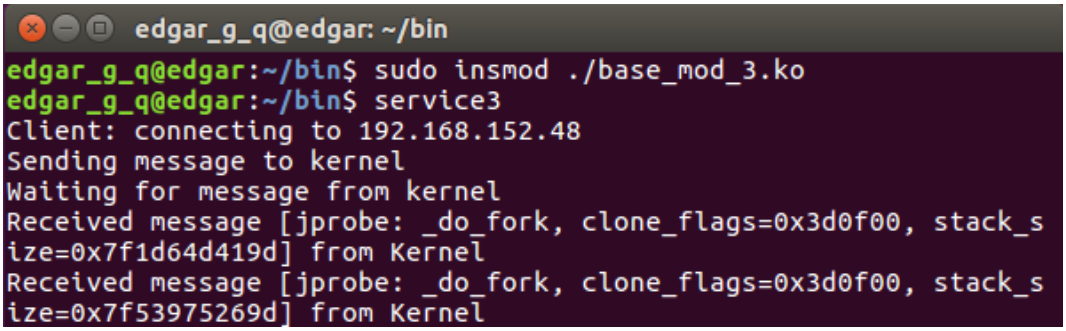


Figura 26: Resultats al dispositiu IoT (Kprobes)

8.2.3. Valoració

Per a aquest sistema, no ha estat possible que els dispositius IoT hagin de demanar autorització per a efectuar accions. Això es deu a la limitació per part del mètode utilitzat: *kprobes* (*jprobes*). Aquest mètode està pensat per a crear mòduls que puguin afegir informació "depuradora" (*debug*) al kernel. A més, la funció que s'implementa com a *jprobes* ha d'acabar amb la crida a *jprobe_return()*, que té com a principal conseqüència la execució de la funció original.

Per tal que el sistema funcionés tal i com es desitja, la mòdul dins el dispositiu IoT hauria de restar a l'espera de rebre una resposta dins la funció *jprobe*, per tal de decidir si cridar *jprobe_return()* (executar funció original) o no, però això presenta dos problemes. D'una banda, en cas de resposta negativa per part del node guardià, la *jprobe* no acabaria amb la crida obligatòria *jprobe_return()* i el sistema es quedaria bloquejat. D'altra banda, s'ha de tenir en compte el procediment per rebre la resposta mitjançant un *netlink socket*. El mòdul creat al dispositiu IoT, per a rebre dades mitjançant aquest socket, es basa en una rutina especial que s'executa cada cop que rep un missatge. Així doncs, mentre el sistema

estigués a la rutina *jprobe* esperant una resposta, la rutina associada a rebre dades s'intentaria executar i el sistema quedaria també bloquejat.

Malgrat aquestes limitacions, ha estat possible que el sistema creat pugui informar al node guardià de les accions realitzades des del dispositiu IoT, ja que la funció *jprobes* no presenta cap problema si abans de cridar *jprobe_return()* s'envien dades mitjançant el *netlink socket*.

Aquest sistema és aplicable a qualsevol sistema operatiu amb Linux Kernel, donant una cobertura semblant al sistema de l'apartat 8.1.

Quant a les accions interceptables, això són pràcticament qualsevol rutina del Kernel (símbols de */proc/kallsyms*), i per tant suposa una millora important amb respecte al sistema anterior. L'únic símbol que *kprobes* no pot sondejar és ell mateix, fet que implica que hi ha certes funcions que *kprobe* no pot sondejar (possible bucle recursiu). *Kprobes* gestiona dites funcions com una llista negra, la qual compara amb l'adreça que vol interceptar abans de registrar la sonda.

9. Pressupost

A continuació es mostrarà el càlcul del cost de realització del projecte. Es detallarà el cost de cada component utilitzat, de les eines utilitzades per recrear CommunityGuard i per realitzar proves, de l'amortització de l'ordinador utilitzat i de les hores dedicades.

MATERIAL			
Element	Quantitat	Preu Unitari	Cost Total (€)
Amortització Servidor	5 mesos	300,00 €	1500,00
Pantalla HDMI	1	200,00 €	200,00
Raspberry Pi 2	1	32,50	32,50
Adaptador USB a Ethernet	1	14,89	14,89
Material auxiliar (teclat, ratolí, cables Ethernet i HDMI)	1	100,00	100,00
Total			**1.847,39**

DESPESES DE PERSONAL			
Tipus de feina	Hores	Cost Horari (€/h)	Cost Total (€)
Estudi previ	80	60,00	4.800,00
Programació	120	60,00	7.200,00
Muntatge	20	60,00	1.200,00
Proves	30	60,00	1.800,00
Redacció memòria	80	60,00	4.800,00
Total			**19.800,00**

CONSUM DE RECURSOS			
Tipus de recurs	Hores	Cost Horari (€/h)	Cost Total (€)
Consum elèctric del PC i plaques (100W)	270	0,14	37,80
Consum elèctric de l'espai de treball	270	0,14	37,80
Total			**75,60**

COST TOTAL DEL PROJECTE	
Part	Cost (€)
Material	1.847,39
Temps de treball	19.800,00
Consum de recursos	75,60
Total	**21.722,99**

10. Impacte ambiental

En aquest capítol es farà un anàlisi de l'impacte ambiental del projecte. És necessari tenir en compte dos aspectes clarament diferenciats: la normativa RoHS (de l'anglès *Restriction of Hazardous Substances*), que afecta a la indústria sencera de l'electrònica i de molts productes elèctrics també, restringint l'ús de diferents materials perillosos, i la influència de les transmissions per radiofreqüència en les persones i en altres elements electrònics propers als transmissors, regulada per la normativa europea ICNIRP (de l'anglès *International Commission on Non-Ionizing Radiation Protection*)

La normativa RoHS imposa restriccions sobre les substàncies que es mostren a la Taula 4. En aquest projecte, els dispositius emprats com a node guardià han estat el BeagleBone Black i la Raspberry Pi 2, i ambdós compleixen amb les restriccions d'aquesta normativa. Qualsevol hardware emprat com a Community Outpost (servidor en el núvol) ha de complir amb la mateixa.

Quant a les transmissions per radiofreqüència, aquestes són generades en cada xarxa pel router, o bé pel dispositiu que fa de node guardià, si aquest és un BeagleBone Black o una Raspberry Pi 2 amb un adaptador Wi-fi (no és necessari, el presenten algunes versions). En qualsevol cas, les emissions generades són de 900 MHz o de 2,4 GHz. Segons la normativa europea, en aquest rang de freqüències, els espectres electromagnètics que es generen no poden causar danys com dolors o cremades, entre d'altres, a les persones.

Substància	Quantitat màxima (ppm)
Plom (Pb)	1000
Mercuri (Hg)	100
Cadmi (Cd)	100
Crom hexavalent (Cr VI)	1000
Bifenils polibromats (PBB)	1000
Èters de difenil polibromats (PBDE)	1000
Bis (2-etilhexil) ftalat (DEHP)	1000
Benzil butil ftalat (BBP)	1000
Ftalat de dibutil (DBP)	1000
Ftalat de diisobutil (DIBP)	1000

Taula 4: Restriccions sobre substàncies perilloses, RoHS

Conclusions

La principal conclusió del projecte és que ha estat possible complir amb els objectius proposats i que el sistema desenvolupat compleixi amb totes les especificacions del projecte. S'ha aconseguit analitzar i documentar CommunityGuard i s'ha entès amb quines eines ha estat desenvolupat i per quins motius. A més, s'ha recreat CommunityGuard en l'entorn que s'havia proposat, usant només màquines virtuals, i modificant el sistema operatiu dels nodes guardians per a analitzar la seva flexibilitat. En realitzar tot això, s'observa que CommunityGuard no està del tot acabat, i que està pensat per a uns sistemes operatius concrets (Debian Linux pel node guardià i Centos pel Community Outpost). per tant, s'ha hagut d'acabar d'implementar el codi de CommunityGuard i adaptar-lo al sistema operatiu Ubuntu pel cas del node guardià.

D'altra banda, s'ha analitzat amb èxit la necessitat d'optimització del software inclòs al node guardià (el conjunt de normes Snort). En base als resultats de les proves realitzades amb Snort, s'observa clarament que no és necessari realitzar aquesta optimització. Si el resultat hagués estat afirmatiu, l'ampliació de CommunityGuard hagués consistit en la recerca d'un mètode per optimitzar les normes Snort incloses, i possiblement no en la recerca d'un sistema de seguretat per als dispositius IoT.

Quant a aquest sistema, s'ha arribat a dues alternatives diferents quant al rang d'accions que pot supervisar o controlar, i el tipus de resposta que es pot donar en cada cas. En qualsevol cas, el sistema actua directament sobre els dispositius IoT i resideix en aquests i en els nodes guardians, assolint els objectius proposats. La primera alternativa dóna cobertura principalment a tots aquells símbols per als quals s'ha especificat, en codi font C, un enllaç extern, mentre que la segona alternativa dóna cobertura a qualsevol rutina del Kernel. L'altra diferència, però, és que la primera alternativa permet acceptar o denegar una acció per part del dispositiu IoT, i la segona només informa al node guardià de l'activitat als dispositius IoT, deixant de banda una possible resposta, que en principi hauria de ser proporcionada per una altra entitat.

Finalment, ambdós sistemes es poden aplicar a la majoria de dispositius IoT amb sistemes operatius de la família Unix. Això suposa una bona flexibilitat si es té en consideració que els dispositius IoT no tenen la necessitat de presentar la funcionalitat completa d'un computador normal, i per tant, és més probable que presentin sistemes operatius d'aquest tipus.

Bibliografia

[1] Chase E. Stewart, Anne Maria Vasu. CommunityGuard: A Crowdsourced Home Cyber-Security System [en línia]. Eric Keller. ACM Digital Library, 2017. 24 de Març de 2017. ISBN: 978-1-4503-4908-6. [Consulta: Gener de 2017]. Disponible a <https://dl.acm.org>.

[2] University of Colorado Boulder, Office of Information Technology. VPN (Virtual Private Network [en línia]. 25 d'Abril de 2017. [Consulta: Febrer de 2017]. Disponible a <https://oit.colorado.edu/services/network-internet-services/vpn>.

[3] Github. Connecting to GitHub with SSH [en línia]. [Consulta: Febrer de 2017]. Disponible a <https://help.github.com/articles/connecting-to-github-with-ssh/>.

[4] Mobatek. MobaXterm: Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more [en línia]. Versió 9.4. [Consulta: Febrer de 2017]. Disponible a <http://mobaxterm.mobatek.net/>.

[5] Phil Zona. Install VNC on Ubuntu 16.04 [en línia]. 21 de Juny de 2016. [Consulta: Febrer de 2017]. Disponible a <https://www.linode.com/docs/applications/remote-desktop/install-vnc-on-ubuntu-16-04>.

[6] NixCraft. How to install KVM on Ubuntu 14.04 LTS Headless Server [en línia]. 27 de gener de 2016. [Consulta: Febrer de 2017]. Disponible a <https://www.cyberciti.biz/faq/how-to-install-kvm-on-ubuntu-linux-14-04/>.

[7] Linux Kernel Organization. Linux Kernel Archives [en línia]. [Consulta: Febrer de 2017]. Disponible a <https://mirrors.kernel.org/centos/7/isos/x86_64/>.

[8] Linuxconfig. How to change system keyboard keymap layout on CentOS 7 Linux [en línia]. 21 de Juliol de 2015. [Consulta: Març de 2017]. Disponible a <https://linuxconfig.org/how-to-change-system-keyboard-keymap-layout-on-centos-7-linux>.

[9] Jamie Nguyen. NAT-based network [en línia]. 16 de Desembre de 2016. [Consulta: Març de 2017]. Disponible a <https://jamielinux.com/docs/libvirt-networking-handbook/nat-based-network.html>.

[10] WikiSYsop. Configuring Guest Networking [en línia]. 20 d'Octubre de 2016. [Consulta: Març de 2017]. Disponible a <http://www.linux-kvm.org/page/Networking>.

[11] Debian Wiki. NetworkConfiguration [en línia]. 8 de Març de 2016. [Consulta: Març de 2017]. Disponible a <https://wiki.debian.org/NetworkConfiguration>.

[12] Ganesh35. Step-By-Step Configuration of NAT with iptables [en línia]. [Consulta: Març 2017]. Disponible a <https://www.howtoforge.com/nat_iptables>.

[13] SuaSwe. Re: OpenSSH troubles:“Permission denied (publickey)” [en línia]. 27 de Febrer de 2012. [Consulta: Març-Abril 2017]. Disponible a <https://ubuntuforums.org/showthread.php?t=1932058>.

[14] Ravi Saive. How to Enable EPEL Repository for RHEL/CentOS 7.x/6.x/5.x [en línia]. 24 de Juny de 2014. [Consulta: Març-Abril 2017]. Disponible a <http://www.tecmint.com/how-to-enable-epel-repository-for-rhel-centos-6-5/>.

[15] Cisco and/or its affiliates. Snort [en línia]. Versió 2.9.9.0. [Consulta: Febrer-Abril de 2017]. Disponible a <https://snort.org/>.

[16] Jacklangstone. How to install Debian on Raspberry Pi [en línia]. 2 de Juny de 2012. [Consulta: Abril 2017]. Disponible a <http://www.thefruitycomputer.com/forums/tutorials/article/2-how-to-install-debian-on-raspberry-pi/>.

[17] Raspberry Pi Foundation. Downloads [en línia]. [Consulta: Abril 2017]. Disponible a <https://www.raspberrypi.org/downloads/>.

[18] Simon Tatham. PuTTy [en línia]. Versió 0.69. 29 Abril 2017. [Consulta: Abril 2017]. Disponible a <http://www.putty.org/>.

[19] Universita’ degli Studi di Napoli “Federico II” (Italy), Dipartimento di Ingegneria Elettrica e delle Tecnologie dell’Informazione (DIETI). D-ITG, Distributed Internet Traffic Generator [en línia]. Versió 2.8.1. [Consulta: Abril de 2017]. Disponible a <http://www.grid.unina.it/software/ITG/documentation.php>.

[20] Cmcginty. Force local IP traffic to an external interface [en línia]. [Consulta: Abril de 2017]. Disponible a <https://serverfault.com/questions/127636/force-local-ip-traffic-to-an-external-interface>.

[21] Rafał Cieślak. Dynamic linker tricks: Using LD_PRELOAD to cheat, inject features and investigate programs [en línia] 2 Abril 2013. [Consulta: Maig de 2017]. Disponible a <https://rafalcieslak.wordpress.com/2013/04/02/dynamic-linker-tricks-using-ld_preload-to-cheat-inject-features-and-investigate-programs/>.

[22] Brian “Beej Jorgensen”. Unix Sockets [en línia]. Versió 1.1.3. 1 Desembre de 2015. [Consulta: Maig de 2017]. Disponible a <http://beej.us/guide/bgipc/output/html/multipage/unixsock.html>.

[23] Brian “Beej Jorgensen”. Beej’s Guide to Network Programming Using Internet Sockets [en línia]. Versió 3.0.21. 8 de Juny de 2016. [Consulta: Maig de 2017]. Disponible a <http://beej.us/guide/bgnet/>.

[24] Jim Keniston, Prasanna S Panchamukhi, Masami Hiramatsu. Kernel Probes (Kprobes) (en línia). [Consulta: Maig de 2017]. Disponible a <https://www.kernel.org/doc/Documentation/kprobes.txt>.

[25] Kevin Kaichuan He. Kernel Korner – Why and How to Use Netlink Socket (en línia). 5 de Juny de 2005. [Consulta: Maig-Juny de 2017]. Disponible a <http://www.linuxjournal.com/article/7356?page=0,0>.

[26] The Open Group. Dlsym (en línia). [Consulta: Maig-Juny de 2017]. Disponible a <http://pubs.opengroup.org/onlinepubs/009695399/functions/dlsym.html>.

[27] The Open Group. Dlopen (en línia). [Consulta: Maig-Juny de 2017]. Disponible a <http://pubs.opengroup.org/onlinepubs/009695399/functions/dlopen.html>.

Bibliografia complementària

[28] William E. Shotts, Jr. The Linux Command Line [en línia]. Verbatim. 29 de Juliol de 2016. [Consulta: Febrer-Juny de 2017]. Disponible a <http://linuxcommand.org/index.php>.

[29] Tutorials Point. Python Tutorial [en línia]. [Consulta: Febrer-Abril 2017] Disponible a <https://www.tutorialspoint.com/python/>.

[30] Projecte OpenSUSE. OpenSUSE [en línia]. [Consulta: Abril de 2017]. Disponible a <https://es.opensuse.org/>.

[31] Tutorials Point. C Tutorial [en línia]. [Consulta: Abril-Juny de 2017]. Disponible a <https://www.tutorialspoint.com/cprogramming/>.

[32] Autor desconegut. Building External Modules [en línia]. [Consulta: Maig-Juny de 2017]. Disponible a <https://www.kernel.org/doc/Documentation/kbuild/modules.txt>.