Universitat Oberta de Catalunya

uoc.edu

Intrusion Detection System to the IoT/Edge Computing

Miquel Comas Marti
Màster Universitari en Ciberseguretat i Privadesa
Seguretat en la internet of things

Consultor: Víctor Méndez Muñoz
RPA: Helena Rifà Pous

06/2022

FITXA DEL TREBALL FINAL

Títol del treball: **Intrusion Detection System to the IoT/Edge Computing**	Intrusion Detection System to the IoT/Edge Computing
Nom de l'autor:	Miquel Comas Marti
Nom del consultor/a:	Víctor Méndez Muñoz
Nom del PRA:	Helena Rifà Pous
Data de lliurament (mm/aaaa):	06/2022
Titulació o programa:	Màster Universitari en Ciberseguretat i Privadesa
Àrea del Treball Final:	Seguretat en la internet of things
Idioma del treball:	Català
Paraules clau	IOT, Seguretat, HIDS

Resum del Treball :

L'objectiu d'aquest treball és estudiar la problemàtica dels atacs als dispositius IoT, que cada cop són més presents en la nostra vida.

S'exposen les plataformes i capacitats dels dispositius per tal d'identificar els tipus d'atac als que són sotmesos en funció de les seves capacitats tècniques. Per cada plataforma es proposa un mètode per tal de detectar els possibles atacs i en alguns casos els mètodes per bloquejar-los.

Amb l'estat de l'art es mostra les opcions actuals en el camp dels HIDS de codi obert i la possibilitat d'utilitzar-los en un entorn tan específic com és el IoT.

Es proposa un model de desplegament i eines per tal de cobrir el màxim nombre de plataformes, independentment del fabricant o capacitats tècniques de cada una.

També es planeja la problemàtica de desplegar un sistema a gran escala, on es gestionin milions de dispositius, i s'aporten algunes possibles solucions.

Com a part del treball, es presenta una possible implementació utilitzant la plataforma Raspberry Pi/Linux com a dispositiu IoT, que són dos exemples de plataformes completament diferents.

Finalment, s'exposen les conclusions a les quals s'ha arribat i es proposen millores i línies futures d'investigació i millora.

Abstract :

The aim of this paper is to study the problem of attacks on IoT devices, which are becoming more and more present in our lives.

The platforms and capabilities of the devices are exposed in order to identify the types of attacks to which they are subjected based on their technical capabilities. A method is proposed for each platform to detect possible attacks and in some cases methods to block them.

The state of the art shows the current options in the field of open source HIDS and the possibility of using them in such a specific environment as the IoT.

A deployment model and tools are proposed to cover the maximum number of platforms, regardless of the manufacturer or technical capabilities of each.

The problem of deploying a large-scale system, where millions of devices are managed, is also exposed, and some possible solutions are provided.

As part of the work, a possible implementation is presented using the Raspberry Pi / Linux platform as IoT device, which are two examples of completely disparate platforms.

Finally, the conclusions reached are set out, and future improvements ,lines of research and improvement are proposed.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

L'Internet de les coses a poc a poc va formant part de les nostres vides: bombetes, endolls, càmeres, panys, assistents, timbres, etc. connectats ens faciliten el nostre dia a dia, i han demostrat sobradament la seva utilitat.

Amb la proliferació dels dispositius IoT, i donat la gran quantitat de dispositius i la varietat d'implementacions, cada cop són un objectiu més atractiu pels hackers, és per això que han crescut les amenaces aquest tipus de dispositius. Diversos estudis indiquen que en els primers 6 mesos del 2020 els atacs contra dispositius IoT han augmentat un 100 %, arribat als 1,5 Bilions.

Avui dia, cada fabricant és l'encarregat d'implementar les mesures de seguretat en la seva pròpia solució, i en cas de detectar algun atac o falla de seguretat, desplegar una nova versió del software amb les correccions. En molts casos, el fabricant no té la potestat de forçar instal·lació d'una nova versió del software, quedant en mans dels usuaris finals aquesta decisió. Això provoca que existeixin milions de dispositius, que sempre estan connectats a internet, sense les últimes correccions de seguretat publicades pels fabricants, deixant-los desprotegits d'avant nombrosos intents d'atac.

La firma Trendmicro ha informat[1] que dels 10,6 bilions d'intents de connexionisme a ports sospitosos de routers detectats el primer trimestre del 2020, 5,3 bilions involucren el port tcp/23, i la firma creu que provenen d'atacs de cuc sobre dispositius IoT i no de simples escanejos de ports.
El setembre del 2020, IBM X-Force va reportar[2] que els atacs provinents de dispositius IoT es van incrementar un 400 % en el període octubre 2019 – juny 2020 comparats amb els dos anys anteriors. L'informe «Threat Intelligence Report 2020» de Nokia també mostra la tendència a l'alça en el nombre de dispositius IoT infectats:

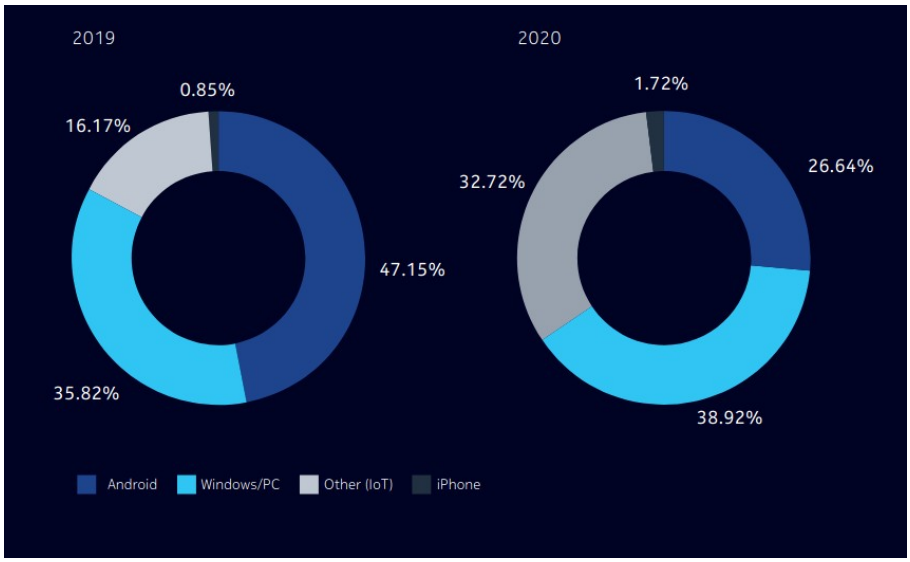


Figura 1: Infected device breakdown 2019 and 2020

Un dels atacs DDoS més famosos es va produir l'octubre del 2016, quan el malware Mirai, utilitzant una gran quantitat de dispositius IoT va fer caure el servei de DNS de Dyn[3], deixant sense funcionament GitHub, Twitter, Reddit, Netflix, Airbnb i molts altres. La firma Radware ha informat[4] que una nova variant de Mirai que anomenen Dark.IoT botnet està funcionant des de febrer del 2021.

L'objectiu d'aquest treball és proposar una solució, des del punt de vista de la detecció d'intrusos en els mateixos dispositius o xarxa IoT, que permeti detectar problemes de seguretat independentment de la versió de software que executi el dispositiu. El sistema ha de proporcionar alertes que puguin ser consumides tant pel fabricant del dispositiu com pel propietari d'aquest, i addicionalment bloquejar el possible atac.

La solució proposada ha de tenir en compte la gran diversitat de plataformes, hardware i software, que s'utilitzen per desenvolupar els dispositius IoT, així com les limitacions de potència (CPU) i memòria que comparteixen la majoria d'aquests dispositius. L'escalabilitat de la solució també és un factor clau, ja que es podrien arribar a gestionar milions de dispositius.

1.2 Objectius del Treball

- Investigar els tipus de dispositius, plataformes, SO, etc. per fer-se una idea de la disparitat de tecnologies en l'àmbit del IoT i identificar les característiques comunes d'aquest tipus de dispositius.

- Investigar els tipus d'atacs més rellevants sobre els dispositius IoT.

- Investigar si existeix alguna correlació entre els tipus d'atac i la plataforma, SO, etc.

- Fer un estat de l'art de sistemes HIDS en general i específicament en l'àmbit de IoT

- Recopilar informació sobre les metodologies per detectar atacs: ML, patrons, firmes, perfilat de tràfic de xarxa, etc.

- Definir les característiques que cal que tingui la solució proposada, com ara: escalabilitat, entorn d'aplicació, mètodes de comunicació, tipus d'atacs a detectar, etc. tenint en compte les especificitats de les plataformes IoT: poca memòria i CPU entre d'altres.

- Proposar una arquitectura per tal de satisfer les característiques explicades al punt anterior.

- Trobar, dins l'àmbit del codi obert possibles eines, llibreries, etc. relacionades amb els HIDS que ens puguin ajudar en la implementació.

- Proposar els mètodes de detecció així com les eines i llibreries, que s'usaran per a cada plataforma.

- Implementar l'arquitectura proposada en una prova de concepte, utilitzant la plataforma Rasperry pi.

- Proposar mètodes per implementar la resposta activa.

- Fer propostes de millora per continuar el treball.

1.3 Enfocament i mètode seguit

S'enfocarà aquest treball pensant en la possible implementació a gran escala de la solució proposada. La solució podria ser una mena «estàndard» per la implantació a milions de dispositius IoT que permeti una alerta global i no depenent de cap fabricant en concret. La idea general és que si es detecten suficients mostres d'atacs en dispositius IoT de diferents tipus i fabricants es pugui aixecar una alerta al més aviat possible.

Es tindran en compte els factors que afecten l'escalabilitat de la solució, així com els que poden afectar la seguretat d'aquesta.

La primera fase serà investigar l'estat de l'art, les plataformes, atacs, mètodes de detecció, etc. i fer una proposta de les característiques i l'arquitectura que idealment tindrà la solució. La proposta pot implicar continuar amb algun treball de recerca ja iniciat o aprofitar parts d'estudis en matèria tant dels HIDS com de IoT.

El següent pas, ja pensant en la prova de concepte, serà estudiar l'existència de solucions ja creades en l'àmbit dels HIDS i veure si és possible adaptar-les o reaprofitar-les del tot o parcialment per tal de satisfer les especificitats d'una xarxa IoT. En el cas de no trobar cap solució adequada, s'estudiarà si és possible utilitzar parcialment components de solucions ja existents, pertanyin o no a l'àmbit dels HIDS.

En últim cas, si no és possible trobar components ja existents que permetin implementar la proposta, es plantejarà un model teòric.
En qualsevol cas, es presentarà una implementació utilitzant Raspberry PI com a plataforma de dispositius IoT, i si és necessari s'eliminaran prestacions per tal de fer una demostració pràctica de la solució.

No es veu factible desenvolupar una solució des de zero, així doncs sembla molt més encertat investigar dins l'àmbit del codi obert per tal de trobar solucions i components ja creats, encara que no estiguin directament pensats per dispositius IoT.
Per dur a terme el treball se seguirà una metodologia en cascada, ja que en ser un sol individu i tenir les tasques una forta dependència de les predecessores aquest enfocament sembla el més adient.

En aquest treball no es consideraran dispositius IoT els aparells que tradicionalment han estat connectats a internet, com els routers, VDRs, Televisors, NAS, Ap wifi, etc.

1.4 Planificació del Treball

Investigació: D'entrada es recopilarà i analitzarà informació sobre tipus de dispositius, arquitectures, SO més usats, plataformes software, tipus d'atacs, etc. La hipòtesi és que depenent de les capacitats tècniques de cada dispositiu, els atacs són de diferents tipus.

Estat de l'art: Es buscaran treballs i articles sobre seguretat del IoT i en especial relacionats amb els HIDS. En ser la seguretat en el IoT un tema cada cop més important, es força probable que ja existeixin treballs iniciats en aquest camp.

Investigar mètodes per detectar cada tipus d'atac: En el món dels HIDS s'utilitzen diferents mètodes per detectar possibles atacs, com per exemple: basats en comportament, signatures, tècniques de ML, etc.

Crear un llistat de característiques desitjades: Partint del treball fer en els punts anteriors, cal fer un llistat de característiques desitjades per la solució. Algunes de les bàsiques són: seguretat, escalabilitat.

Buscar solucions preexistents o components reutilitzables: En l'àmbit del codi obert ja existeixen solucions HIDS força avançades, però es força evident que no han estat pensades per entorn IoT, en especial el que fa referència a l'escalabilitat i l'ús de recursos en dispositius amb poca CPU i memòria. Així i tot, cada dia apareixen nous projectes, per tant cal verificar-ho, potser algun es pot usar com a base per una nova solució. Un component interessant pot ser la «AWS IoT Device Defender library[5]».

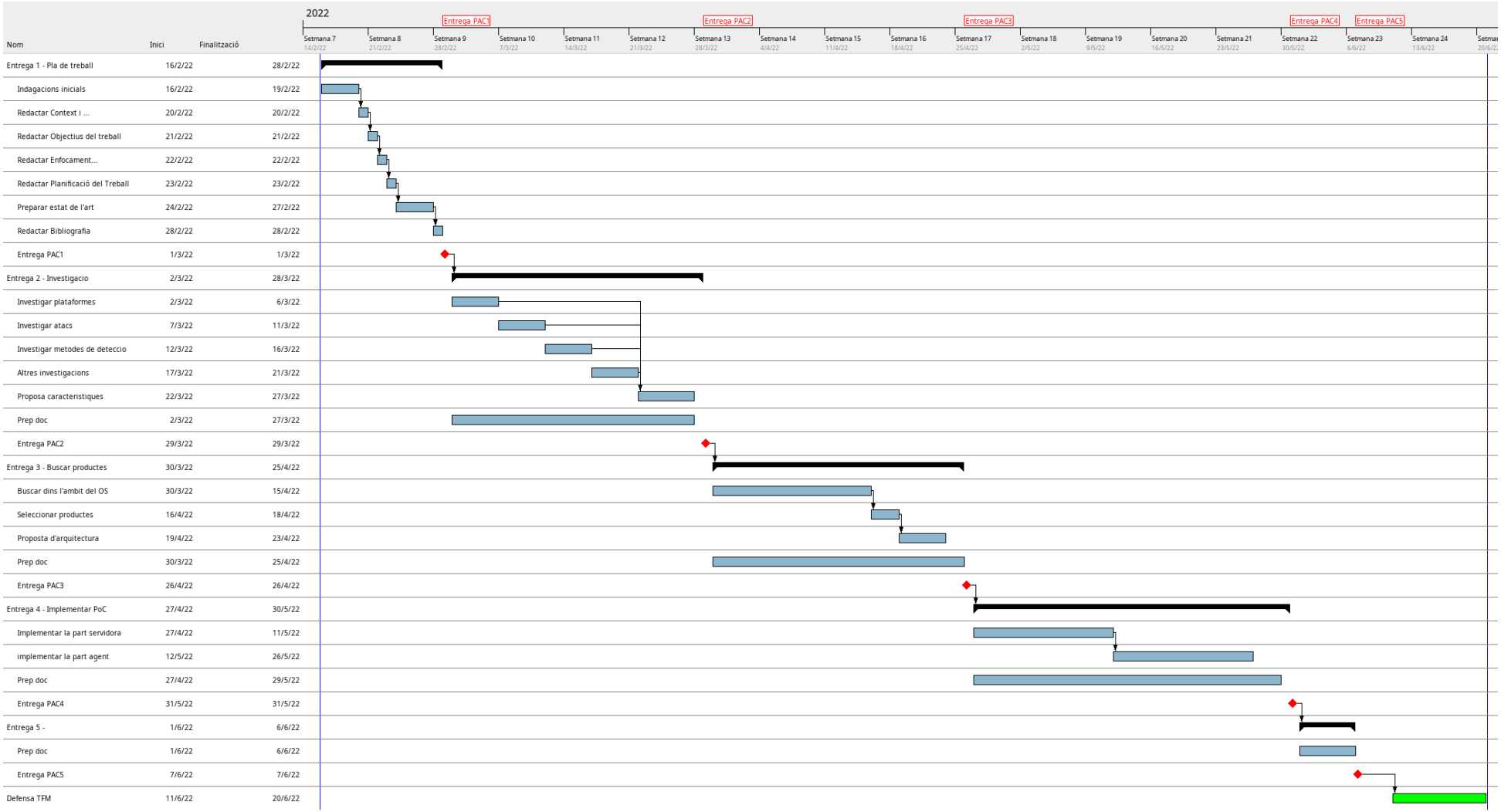
Proposar una arquitectura: D'entrada sembla que existirà un «servei» central que recollirà informació dels agents, però en una arquitectura a gran escala segurament caldran components auxiliars per ajudar amb l'escalabilitat. També caldrà veure les capacitats dels agents segons les especificacions tècniques de cada plataforma.

Creació del PoC: Utilitzant la Raspberry pi com a dispositiu IoT, i un laptop com a servidor, s'implementarà la solució proposada en el punt anterior. Depenent de les propostes fetes, potser caldrà usar algun servei en cloud.

Analitzar resultats: Un cop el PoC implementat, es veurà si s'han aconseguit els resultats desitjats, mostra evidències del funcionament. Es proposaran correccions i millores, així com esbossar com es podria continuar el treball fet.

Documentació: En totes les tasques esmentades en aquest apartat es documentaran els treballs realitzats i s'adjuntaran en aquest TFM.

El següent Gantt és una primera versió de la planificació, depenent dels resultats obtinguts en les fases d'investigació o comentaris del tutor, els continguts de cada entrega poden varia significativament, però no les dates d'entrega.



Nom	Inici	Finalització
Entrega 1 - Pla de treball	16/2/22	28/2/22
Indagacions inicials	16/2/22	19/2/22
Redactar Context i ...	20/2/22	20/2/22
Redactar Objectius del treball	21/2/22	21/2/22
Redactar Enfocament...	22/2/22	22/2/22
Redactar Planificació del Treball	23/2/22	23/2/22
Preparar estat de l'art	24/2/22	27/2/22
Redactar Bibliografia	28/2/22	28/2/22
Entrega PAC1	1/3/22	1/3/22
Entrega 2 - Investigacio	2/3/22	28/3/22
Investigar plataformes	2/3/22	6/3/22
Investigar atacs	7/3/22	11/3/22
Investigar metodes de deteccio	12/3/22	16/3/22
Altres investigacions	17/3/22	21/3/22
Proposa caracteristiques	22/3/22	27/3/22
Prep doc	2/3/22	27/3/22
Entrega PAC2	29/3/22	29/3/22
Entrega 3 - Buscar productes	30/3/22	25/4/22
Buscar dins l'ambit del OS	30/3/22	15/4/22
Seleccionar productes	16/4/22	18/4/22
Proposta d'arquitectura	19/4/22	23/4/22
Prep doc	30/3/22	25/4/22
Entrega PAC3	26/4/22	26/4/22
Entrega 4 - Implementar PoC	27/4/22	30/5/22
Implementar la part servidora	27/4/22	11/5/22
implementar la part agent	12/5/22	26/5/22
Prep doc	27/4/22	29/5/22
Entrega PAC4	31/5/22	31/5/22
Entrega 5 -	1/6/22	6/6/22
Prep doc	1/6/22	6/6/22
Entrega PAC5	7/6/22	7/6/22
Defensa TFM	11/6/22	20/6/22

1.5 Estat de l'Art

En aquest estat de l'art s'exposarà superficialment les problemàtiques i tendències tant del món dels HIDS així com de la seguretat en els entorns IoT. En les entregues posteriors es profunditzarà en temes més específics dels HIDS en plataformes IoT.

El terme «Internet of Things» se l'autoatribueix el Kevin Ashton el 1999 en un treball per l'empresa Procter and Gamble usant tags RFID per gestionar la cadena de subministraments, tal com ell mateix explica en un article[6] per la revista RfidJournal. El mateix 1999 ell i un grup de persones amb idees semblants varen formar l'Auto-ID Center research consortium.

ENISA defineix IoT com "cyber-physical ecosystem of interconnected sensors and actuators, which enable intelligent decision making". Aquesta definició ja implica que els sistemes IoT gestionen informació, la que recullen dels sensors o intercanvien amb altres dispositius IoT. També implica que existiran mecanismes (canals físics, protocols, etc.) per la comunicació entre sensors. El fet d'indicar que els dispositius poden prendre decisions deixa clar que disposen d'algun tipus de sistema de processament (microprocessador, microcontrolador, etc.).

Encara que ningú sap exactament el nombre total de dispositius IoT connectats a internet, s'estima que a final del 2021 existien al voltant d'11 bilions i el 2030 s'espera tenir-ne 25 bilions de connectats. Tot i que d'entrada l'IoT es va desenvolupar de pressa en l'àmbit domèstic, avui dia està creixent de forma molt ràpida l'adopció en la indústria i en projectes governamentals.

Diferents estudis posen de manifest que molts dels dispositius IoT, ja siguin per ús domèstic, mèdic[7], industrial[8], etc. tenen deficiències des del punt de vista de la seguretat.

D'acord amb Eurofins, un dels punts([9] punt 9) febles de seguretat dels dispositius IoT és el fet de no conèixer si han estat hackejats. D'acord amb aquesta firma, normalment quan els dispositius són hackejats continuen funcionant de forma normal, sense que l'usuari es pugui adornar de les diferències en els patrons de consum o en l'ús de l'amplada de banda, entre altres indicadors.

El que s'extrau dels informes i estudis és que la majoria de dispositius IoT no es dissenyen tenint present la seguretat, molts no tenen sistemes operatius tradicionals o prou memòria o capacitat de processament per incorporar característiques de seguretat. Com que el nombre de dispositius connectats a internet no para de créixer, amb aproximadament un milió de nous dispositius connectats cada dia, el resultat és que cada cop existeix una quantitat de dades més gran movent-se lliurement entre dispositius, entre xarxes, oficines remotes, treballadors remots i clouds públics, el que dificulta molt seguir i assegurar aquest flux de dades.

Existeix una preocupació creixent a causa del gran augment de dispositius i tecnologies en el món del IoT, en especial en temes relacionats amb seguretat i privadesa, en conseqüència empreses i governs estan treballant per solucionar-los mitjançant el desenvolupament d'estàndards, guies i marcs regulatius[10][11][12].

En el camp dels IDS/HIDS la història comença el 1980, amb un «paper» del James Anderson's, Computer Security Threat Monitoring and Surveillance, en aquest document neix la idea de la detecció d'intrusos. Més tard, entre els anys 1984 i 1986, en Dorothy Denning i Peter Neumann investiguen i desenvolupen el primer model d'IDS en temps-real. El primer prototip es va anomenar Intrusion Detection Expert System (IDES), que era un sistema basat en regles i entrenat per detectar activitat maliciosa ja coneguda.

El desenvolupament comercial de les tecnologies de detecció d'intrusos es va produir als inicis dels anys 1990. Haystack Labs va ser el primer venedor comercial d'eines IDS, amb la seva línia Stalker de productes basats en host. SAIC també estava desenvolupant una forma de detecció d'intrusions basada en host, anomenada Computer Misuse Detection System (CMDS).

La descripció més acceptada dels HIDS: un IDS basat en host (HIDS) supervisa les característiques d'un sol host i els esdeveniments que ocorren dins d'aquest per a detectar i aturar activitats sospitoses.

Els HIDS, en principi, són de naturalesa «passiva», la qual cosa implica que està destinada a detectar activitats sospitoses, no prevenir-les. Així i tot, és molt habitual que els mateixos HIDS o en combinació amb els IPS proporcionin funcions per mitigar els possibles atacs detectats.

Les tècniques més habituals per la detecció de les intrusions amb sistemes IDS/HIDS són:

Detecció basada en signatures: HIDS basat en signatures supervisa els paquets a la xarxa, fitxers, registres, etc. i els compara amb patrons d'atac preconfigurats i predeterminats coneguts com a signatures.

A diferència dels HIDS basats en signatures, els **basats en anomalies** es basen més en l'anàlisi del «comportament fiable» i utilitzen tècniques d'aprenentatge automàtic per marcar el comportament maliciós.
Dins dels HIDS basats en detecció d'anomalies existeixen diferents tècniques per detectar el «comportament no fiable» del dispositiu, basats en estadístiques, en coneixement previ, i més recentment en Machine Learning (ML).

Avui dia moltes de les implementacions[13] dels HIDS disposen de més d'un mecanisme de detecció d'intrusió: anàlisis de logs, detecció de rootkits, anàlisis de tràfic de xarxa, integritat de fitxers, etc.

S'estan proposant nous models pels sistemes IDS/HIDS aprofitant els avantatges d'Edge Computing, encara que el fet d'incrementar el nombre de

dispositius que intervenen en el funcionament del IoT també presenta els seus propis reptes des del punt de vista de la seguretat.

2. Investigació

2.1 Plataformes

En el procés de recerca sobre plataformes posaré el focus en els sistemes operatius, ja que seran aquest els que determinaran la «potència» de la plataforma:

Existeixen més d'una cinquantena de sistemes operatius per dispositius IoT:

No.	IoT operating system
1	Alpine Linux
2	Amazon FreeRTOS
3	Android Things
4	Apache Mynewt
5	Arch Linux ARM
6	ARM Mbed OS
7	Balena Yocto Linux
8	Batocera.linux
9	Bedrock Linux
10	BMC64
11	Brillo
12	Chromium OS
13	Contiki
14	Device OS
15	Devuan GNU
16	DietPi
17	Domoticz
18	Embedded Linux
19	FreeBSD

No.	IoT operating system
20	Fuchsia
21	Gentoo
22	Huawei LightOS
23	Kali Linux
24	Kano
25	Lakka
26	LibreELEC
27	Linutop
28	Micrium uC/OS
29	Minibian
30	Mongoose OS
31	Nano RK IoT
32	Nucleus RTOS
33	NuttX Real-Time
34	OpenELEC
35	OpenMediaVault
36	OpenSUSE
37	OS-IoT
38	OSMC

No.	IoT operating system
39	Particle Device OS
40	Raspbian
41	RaspBSD
42	RetroPie
43	Riot OS
44	RISC OS
45	Rokos
46	SARPi
47	Siemens MindSphere
48	Snappy
49	TinyOS
50	TizenRT
51	Ubuntu Core
52	Ubuntu Mate
53	VxWorks 653
54	Wind River VxWorks
55	Windows 10 for IoT
56	Zephyr

Figura 2: Cekerevac, Z. Top seven IoT operating systems in mid-2020

MEST Journal Vol. 8 No.2 pp. 47-68

Però els més utilitzats segons l'enquesta[14] anual de la fundació Eclipse són els següents:

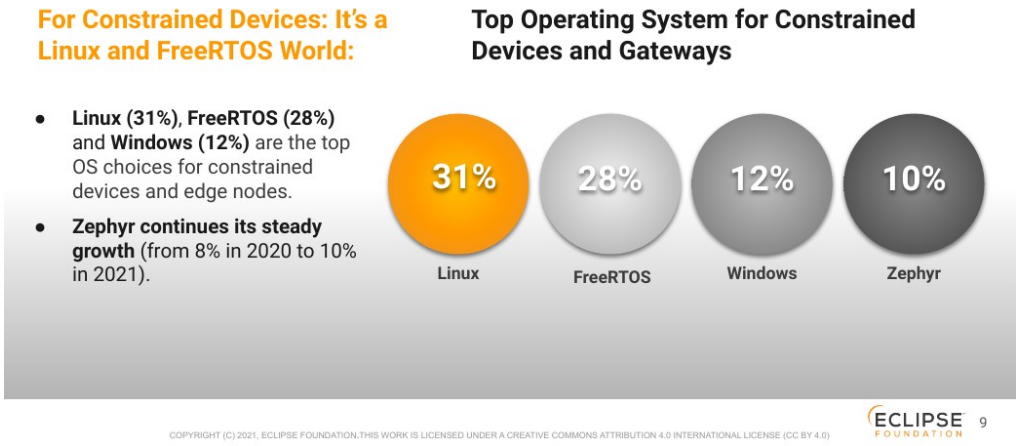


Figura 3: Eclipse IoT Developer Survey 2021 results.

Aquesta enquesta de la fundació Eclipse sols indica sobre quins sistemes operatius estan treballant els desenvolupadors de solucions IoT, i no pas el percentatge de dispositius IoT que estan funcionant amb els diferents sistemes operatius. Així i tot, ens dona una idea aproximada del que podríem trobar, i sí que sembla força clar que el Kernel Linux és el més usat amb diferència.

Com es pot veure, la suma de percentatges d'ús d'aquests 4 sistemes operatius acumula un total del 81% del total, així doncs a l'Annex6 s'analitzaren més detalladament aquests sistemes operatius per identificar els punts en comú i les diferencies.

Encara que sols s'han analitzat 4 sistemes operatius per dispositius IoT, i que tots tenen les seves especificitats, proposo la següent classificació en l'àmbit d'aquest treball:

Els dispositius IoT es poden dividir bàsicament en dues classes depenent del tipus sistema operatiu que utilitzin. La primera categoria seria els sistemes operatius que anomenaré «**complets**», amb kernel, libc i llibreries. Normalment funcionen amb dispositius amb processadors de 32/64 bits, disposen de sistema de fitxers, poden carregar llibreries de forma dinàmica i les aplicacions corren amb permisos d'usuari.

Anomenaré sistemes operatius «**simples**» aquells que executen un kernel, sense possibilitat de càrrega de llibreries dinàmiques, sense sistema de fitxers, normalment s'executen sobre microcontroladors i les aplicacions corren dins el mateix kernel.

Es veurà quan parlem dels tipus d'atacs que depenent del tipus de sistema operatiu els atacs són diferents, i com caldrà buscar mètodes de detecció diferenciats.

2.2 Atacs i problemes de seguretat coneguts

En aquest apartat comentaré els principals atacs soferts per dispositius IoT o utilitzant-los com a dispositius per llançar atacs.

L'atac més famós utilitzant dispositius IoT va ser el que es va dur a terme l'any 2016[15][16] amb el malware Mirai, la botnet creada amb Mirai va emprar centenars de milers de dispositius IoT segrestats per fer caure Dyn.
Mirai es propaga fent primer un escaneig ràpid on intenta fer connexions TCP SYN a adreces IPv4 pseudoaleatòries, als ports TCP 23 i 2323 de Telnet.

Un cop Mirai descobreix els ports Telnet oberts, intenta per força brutal esbrinar les credencials d'inici de sessió per tal d'infectar el dispositiu. Mirai intenta iniciar sessió amb una llista de deu combinacions de nom d'usuari i contrasenya. Aquestes deu combinacions es trien aleatòriament a partir d'una

llista preconfigurada de 62 credencials que s'utilitzen amb freqüència com a predeterminada per als dispositius IoT.

Després d'iniciar sessió correctament, Mirai envia la IP de la víctima i les credencials relacionades a un servidor d'informes. D'entrada, Mirai intenta avaluar i identificar l'entorn en què s'executa. Aquesta informació s'utilitza per descarregar càrregues útils de la segona etapa i programari maliciós específic del dispositiu. Per exemple, la càrrega útil d'un dispositiu basat en ARM serà diferent d'un de MIPS.

Després d'haver infectat un dispositiu amb èxit, Mirai cobreix les seves pistes suprimint el binari descarregat i utilitzant una cadena alfanumèrica pseudoaleatoria com a nom de procés. Com a resultat, les infeccions de Mirai no persisteixen després del reinici del sistema. Per reforçar-se, el programari maliciós també finalitza diferents serveis que estan vinculats a TCP/22 o TCP/23, incloses altres variacions de Mirai. En aquest punt, el bot espera les ordres del seu servidor d'ordres i control (C2) mentre busca altres dispositius vulnerables.
Aquesta àmplia gamma de metodologies permet que Mirai realitzi atacs DDoS com ara inundació UDP, HTTP i TCP juntament amb atacs de capa d'aplicació, atacs volumètrics i atacs d'esgotament d'estat TCP.

El codi font de Mirai es va pública i des de llavors han sorgit infinitat de versions i s'han expandit les arquitectures hardware que suporta, tal com es pot veure en l'experiment[17] dut a terme per CUJOAI.

Un software maliciós semblant a Mirai és «Gafgyt»[18] que també ha anat evolucionant amb els temps, les últimes versions conegudes d'aquest software utilitzen fins i tot el protocol TOR per connectar als servidors de C2.

Un altre software molt sofisticat per crear botnet utilitzant dispositius IoT i routers és Mozi[19]. Es calcula que aquest malware ha infectat 1,5 Milions de dispositius. És un software que crea una xarxa de botnet P2P utilitzant el protocol DHT, el que ens dona una idea de la complexitat d'aquest. Com els altres softwares comentats anteriorment, ataca dispositius exposats directament a internet.

Un dels últims malwares detectats ha estat el BotenaGo[20], que està escrit en el llenguatge de programació GO, cosa que dificulta la detecció per part d'antivirus. Se'l considera responsable de la infecció de milions de dispositius IoT.

Com es pot veure aquests malwares es propaguen utilitzant serveis d'accés o administració remota com són telnet o ssh, això implica que seran dispositius amb un sistema operatiu «**complet**».

Un zero day[21][22] detectar fa poc en dispositius «Nooie Baby Monitor» és interessant, ja que en un principi es considerava un problema de privacitat, però després es va veure que la combinació de diversos problemes de

seguretat feia que els atacants poguessin arribar a executar[23] codi en els dispositius.
Aquests dispositius utilitzaven MQTT sense autenticació per rebre l'adreça del servidor de vídeo on havien de publicar el que enregistraven. Els atacants sols havien de conèixer l'ID del dispositiu per enviar un missatge MQTT i fer que el dispositiu envies el vídeo a un servidor maliciós. Evidentment això suposava un problema de privacitat enorme, ja que aquest tipus de dispositius s'utilitzen dins de les llars per vigilar nens. El que és encara més greu és que més tard es va demostrar que la funció que llegia els missatges MQTT tenia un buffer overflow que permetia l'execució de qualsevol codi al dispositiu. Entre altres coses això permetia aconseguir les claus d'AWS i veure tots els vídeos guardats pel dispositiu al núvol. El que és més interessant és que aquests dispositius no estan directament exposats a internet (normalment estan darrere un router que fa NAT), però són controlats des del cloud, així doncs no és el típic mètode que tracta d'explotar autenticació dèbil de serveis com telnet o ssh.

Cal comentar que molts fabricant de sistemes de videovigilància han tingut problemes de seguretat greus, sobretot per què són sistemes exposats a internet amb sistema operatiu complets i que acostumen a tenir algun sistema d'administració remota com telnet o ssh. Alguns d'aquest fabricant son: Ring (ara Amazon), TRENDnet, Owlet, KrebsOnSecurity.com, Yi Technology,

Un altre producte amb serioses vulnerabilitats i que ha estat objectiu d'atacs[24], és la línia de sistemes de control d'accés «Nortek Linear eMerge E3-Series»[25]. Aquests dispositius s'utilitzen en els sectors comercial, industrial, bancari, mèdic, detallista i hostaleria per gestionar l'accés dels usuaris a instal·lacions o àrees específiques. Les nombroses[26] vulnerabilitats d'aquest producte permetien l'execució de codi amb els màxims privilegis. El que és interessant d'aquest cas és que aquests productes van ser dissenyats per controlar la seguretat de llocs físics i que no són productes enfocats al mercat domèstics.

Un dispositiu amb vulnerabilitats crítiques[27] era el «Samsung SmartThings Hub». El SmartThings Hub és un controlador central que supervisa i gestiona diversos dispositius IoT, com ara endolls intel·ligents, bombetes LED, termòstats, càmeres, etc. que normalment es desplegarien en una casa intel·ligent. Funciona com un controlador centralitzat per a aquests dispositius i permet als usuaris connectar-se i gestionar els dispositius de forma remota mitjançant un telèfon intel·ligent. El microprogramari que s'executa a SmartThings Hub està basat en Linux i permet les comunicacions amb dispositius IoT mitjançant una varietat de tecnologies diferents com Ethernet, Zigbee, Z-Wave i Bluetooth. El més interessant d'aquest cas és que atacant un sol dispositiu es tenia accés a la resta de dispositius IoT, que podien incloure des del pany de la porta d'entrada a les càmeres de videovigilància. Així doncs si la seguretat dels dispositius IoT és important de forma individual, encara ho és més la dels dispositius que actuen com a HUB.

Fa un temps[28] es va produir un atac a un dispositiu pensat per la gestió de benzineres. Aquest dispositiu és un Linux embedded que permet gestionar els diferents elements d'una benzinera com ara: els dispensadors, el sistema de

pagaments, etc. El dispositiu es connecta a internet per permetre la gestió remota, així doncs permet canviar els preus dels carburants, para els sortidors, etc. Aquest sistema també té l'opció de connectar-se a un servei cloud per tal d'obtenir reports i monitoratge centralitzats. Com s'indica a l'article la seguretat d'aquest dispositiu no difereix molt de la d'un router, així doncs té els mateixos problemes amb els sistemes de gestió remota (passwords coneguts, protocols insegurs, RCE, etc.). En aquest cas l'atac es va produir per tal de robar combustible, però tal com es pot llegir a l'article els atacants podien haver optat per accions molt més d'anyines.
El que crida l'atenció és que tot i ser un dispositiu molt especialitzat, amb un simple escaneig d'IP es van trobar més de 1000 dispositius d'aquest tipus.

2.3 Mètodes de detecció

En el procés de recerca s'ha pogut trobar[29][30][31][32][33], que els mètodes més usats pels IDS són:

Signature-based IDS, aquest mètode utilitza una base de dades de signatures d'atacs coneguts. Les signatures poden ser fitxers que apareixen en els sistemes atacats, modificacions de configuracions, canvis de permisos, sobre escriptura de binaris, entrades específiques en els logs, etc. Aquest mètode compara les signatures del sistema que està funcionant amb les que té guardades a la base de dades, si es detecta la coincidència d'alguna signatura, es dispara una alarma. El mètode també es pot usar amb tràfic de xarxa, encara que pot ser computacionalment costos depenent del tipus d'anàlisis que es vulgui dur a terme. Aquest mètode es força eficient per detectar atacs coneguts, ja que normalment les signatures les creen experts que busquen la manera òptima de detectar l'atac. El punt feble d'aquest mètode és que no pot detectar atacs que no té a la base de dades de signatures, a més els atacants poden variar l'atac per tal d'impedir el matching de la signatura. També cal esmentar que es poden patir els mateixos problemes que tenen els antivirus avui dia, la base de dades de signatures és tan gran que és costos (cpu i memòria) fer la verificació de totes les signatures. Tal com s'explica a l'estudi[29] de Pietro Spadaccino and Francesca Cuomo, s'estan estudiant millores d'aquest mètode de detecció, inclòs un dedicat als dispositius IoT [34]

Anomaly-based IDS, aquest mètode es basa a guardar la signatura (fase d'entrenament) del comportament del dispositiu en el seu funcionament normal, després es mesura el comportament de forma regular i si divergeix del guardat, s'aixeca una alarma. Aquest mètode té la possibilitat de detectar atacs nous, ja que qualsevol atac provocarà un canvi de comportament en el dispositiu atacat. Un punt a favor d'aquest mètode és que també ens permet detectar anomalies no relacionades amb atacs, com poden ser mals funcionaments, configuracions incorrectes o actualitzacions errònies. El gran inconvenient és el nombre de falsos positius provocats entre d'altres per actualitzacions del software, canvis d'ús per part de l'usuari, etc.

En ser sistemes basats en anomalies es poden usar molts paràmetres i esdeveniments per detectar l'atac, com ara patrons de tràfic de xarxa, l'ús de la CPU, processos al sistema, etc.

Existeixen 3 subcategories dins els AIDS:

- **Statistics-based**, basats en estadístiques. Es calculen les probabilitats de cada esdeveniment a la fase d'aprenentatge, i després es comparen amb els del sistema en funcionament. Si estadísticament es produeixen esdeveniments poc probables, s'aixeca una alarma.

- **Knowledge-based AIDS**, aquest mètode pressuposa la creació per part d'experts o del fabricant d'un perfil de tràfic de xarxa (poden ser altres tipus d'esdeveniments). Mentre el sistema està en funcionament es compara de forma regular el comportament amb el perfil generat, si difereixen, es procedeix a aixecar una alarma.

- **Machine Learning-based AIDS**, aquest mètode es tracta en identificar les «features»[35] per tal d'entrenar un model capaç de predir un comportament anormal del dispositiu.

De la mateixa manera que amb el mètode basat en signatures, s'està treballant en millores en els mètodes basats en anomalies, es pot trobar més informació a l'estudi[29] de Pietro Spadaccino and Francesca Cuomo, ja citat. També es poden trobar propostes per detectar atacs de tipus DoS i DDoS basat en detecció d'anomalies a l'estudi «A DDoS Attack Mitigation Framework for IoT Networks using Fog Computing» [36]

Encara que es poden utilitzar molt tipus d'esdeveniments, com l'ús de CPU o l'aparició de certes cadenes (patrons) als logs, sembla que la majoria d'estudis es focalitzen en el comportament del tràfic de xarxa.

Pel que fa a les implementacions pràctiques, els HIDS utilitzats en entorn servidor acostumen a implementar alguns o la totalitat dels següents mètodes:

- **Anàlisis de logs:** Aquest és el mètode més estès per detectar possibles atacs. Normalment s'utilitza una base de dades de signatures per buscar patrons (entrades al fitxer de log que són específiques de determinats atacs) ja coneguts o genèrics que poden prevenir en certa manera atacs desconeguts. És un mètode basat en **signatures.**

- **Detecció de Rootkits o malware:** Es basa a buscar modificacions en el sistema (executables en llocs i amb noms ja coneguts, permisos modificats, ports oberts) que es coneix que són utilitzats per software maliciós. Algunes implementacions són més sofisticades i detecten com a possibles problemes fitxers ocults, amb permisos poc freqüents, etc. En aquest cas les **signatures** són noms i llocs de fitxers, informació sobre ports coneguts, permisos, etc.

- **Integritat de fitxers i registre:** Aquest mètode crea una base de dades de **signatures** dels fitxers o entrades del registre (windows) que es consideren crítics (executables, fitxers de configuració, certificats), i duran el funcionament normal del dispositiu es verifica periòdicament que no s'ha modificat cap signatura. Normalment es verifica la signatura (SHA1 o SHA2, etc.) però també els permisos. Algunes HIDS utilitzen polítiques en comptes de signatures, sobretot en sistemes on es permet fer actualitzacions parcials del sistema.

- **Monitor de processos:** En un primer moment es registren en una base de dades informació sobre els processos que poden estar executant-se en el sistema. Periòdicament es revisen els processos que s'estan executant i si apareix algun procés nou, s'aixeca una alarma. També poden monitorar el **comportament** dels processos, com per exemple veure si obren un port de xarxa (TCP/UDP) que no està autoritzat per al procés. Tot i que utilitza una base de dades amb informació dels processos, aquest mètode es basa en detecció **d'anomalies**.

- **Anàlisis de tràfic de xarxa:** Dins d'aquest mètode podem trobar moltes variants. El funcionament principal és analitzar el tràfic d'entrada i sortida del dispositiu. Aquest mètode té l'avantatge que pot analitzar (és complex, cal poder accedir als certificats i/o claus) el tràfic de connexions xifrades (ssl, ipsec, etc.), ja que és un dels punts finals. Algunes implementacions utilitzen **signatures** per detectar patrons de tràfic associats a atacs coneguts. Altres implementacions analitzen el comportament (**anomalies**) del tràfic, com ara augments no previstos en l'ús de l'amplada de banda o connexions a ports o IPs no previstes.

En el següent gràfic es pot veure una classificació segons el tipus d'IDS i mètode de detecció:

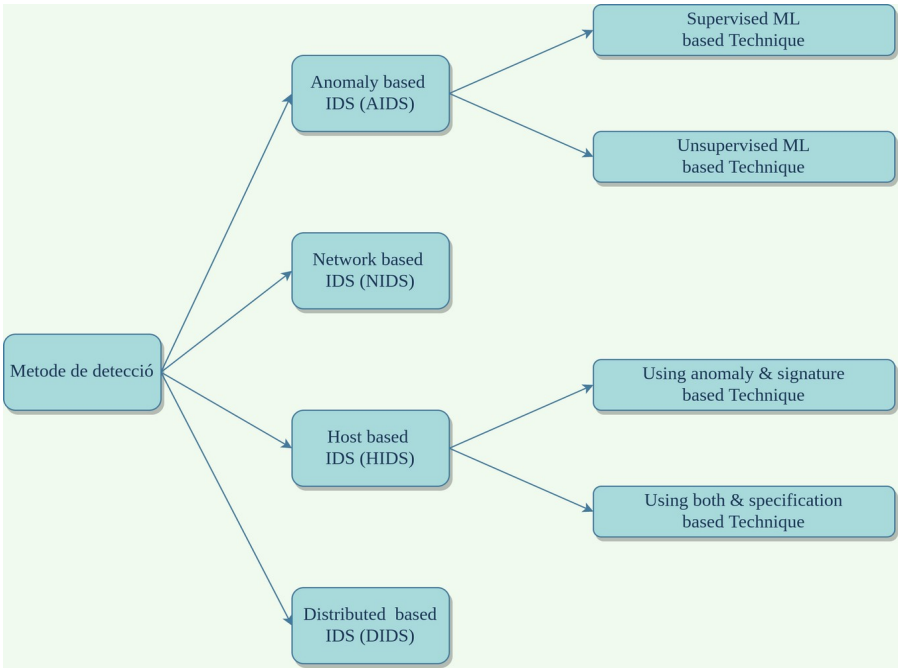


Figura 4: Mètodes de detecció

Com es pot veure, tots aquests mètodes requereixen alguna mena de coneixement previ o fase d'aprenentatge, i és en aquest terreny on s'està treballant per desenvolupar mecanismes per fer automàticament aquest procés, molts basats en alguna de les variants de la intel·ligència artificial (IA).
Els HIDS també poden utilitzar motors de correlació per combinar alguns d'aquests mètodes i obtenir una millor precisió en la detecció.

2.4 Altres investigacions/conclusions

Després de realitzar la investigació sobre els dispositius i els tipus d'atacs es poden fer les següents observacions:

- La majora d'atacs a dispositius IoT es produeixen per a usar-los en xarxes botnets, un cop es controlen suficients dispositius, s'utilitzen per realitzar diferents tipus d'atacs com poden ser DDoS, enviament massiu de correu, etc.
- La majoria d'atacs a dispositius es produeixen a routers/gateway i dvr/cameres, això és així per què normalment aquest tipus de dispositius disposen de força recursos (CPU i memòria), sistema operatiu complet i estan connectats directament a internet (sense NAT en molts casos).
- Hi ha molt pocs atacs a dispositius IoT simples, ja que aquests dispositius no acostumen a tenir una connexió directa a internet, normalment es connecten algun tipus de HUB mitjançant protocols específics de IoT, com pot ser Zigbee, Z-Wave, Bluetooth, BLE, etc.
- Els atacs a dispositius simples són molt més complexos de realitzar, normalment utilitzen sistemes operatius i protocols de xarxa molt variats i que requereixen entendre el seu funcionament. És probable que l'esforç que requereixi no compensi els possibles beneficis que es puguin obtenir del fet de controlar un dispositiu molt limitat pel que fa a potència.
- Els atacs a dispositius simples són per fer DDoS, robar credencials o moure's lateralment per atacar altres dispositius de la xarxa, en tenir poca potència les possibilitats són limitades.
- La majoria de problemes són deguts a l'ús de telnet/ssh i passwords febles. Molt usuaris no són conscients que tenen els ports per la gestió remota exposats a internet.
- La majoria d'atacs són a dispositius amb Linux i SO complets, això és així atès que l'ecosistema de Linux és obert i molt extens. Per un costat els fabricants no han de pagar costos de llicència i disposen de molt software ja produït, així que els és econòmicament rendible utilitzar-los als seus productes. Per l'altre costats els atacants tenen disponible totes les eines i el codi font de la majora d'elles, amb el que poden experimentar i buscar problemes de seguretat de forma molt més senzilla.
- Encara que un dispositiu no estigui exposat directament a internet, es poden utilitzar els serveis (cloud) que comuniquen amb els dispositius per atacar-los (sobretot en entorns amb NAT).
- Alguns servidors de C2 de botnets utilitzen vpn i tor per comunicar-se amb els dispositius infectats, així doncs pot ser complicat utilitzar les IP per rastrejar els servidors C2.

Atac	Capa	Objectiu	Ús	Atacs reals	Exemple
Físics al dispositiu	Dispositiu	DoS, Controlar dispositiu	Baix	Molt pocs	Nest Thermostat
Encriptació de les dades	Dispositiu	Robar dades o credencials	Baix	Molt pocs	
DoS (Denial of Service)	Dispositiu	Danyar reputació	Baix	Molt pocs	
Segrest de firmware	Dispositiu	Crear xarxa Botnets	Baix	Molt pocs	Stuxnet
Botnets	App/SO	Fer atacs DDoS, spam ,etc.	Molt gran	Molts	Mirai, Gafgyt, Mozi, BotenaGO
Man-in-the-Middle	Xarxa	Robar dades o credencials	Baix	Molt pocs	
Ransomware	App/SO	Robar dades, demanar rescat	Baix	Molt pocs	Flocker
Eavesdropping	Xarxa	Robar dades o credencials	Baix	Molt pocs	Smart Baby Monitors
Escalat de privilegis	App/SO	Controlar dispositiu	Mitja	Alguns	Nortek Linear eMerge E3-Series
Força bruta al password	App/SO	Controlar dispositiu	Gran	Molts	Brickerbot
Cloud	Cloud	Controlar dispositiu, robar dades	Baix	Alguns	Nooie Baby Monitor

Taula 1: Atacs comuns als dispositius IoT

Al llistat de la Taula 1 es pot veure els atacs més comuns als dispositius IoT, encara que si tenen SO simple pot no permetre alguns dels atacs.

La columna «Capa» indica a quin nivell es realitza l'atac, així «Dispositiu» indica que pot ser un atac físic o un atac que requereixi accés a totes les parts que conformen el dispositiu. «APP/SO» es refereix a atacs al sistema operatiu o a l'aplicació que dona funcionalitat al dispositiu. «Xarxa» ens indica que és un atac al protocol de comunicacions, i normalment es duen a terme des de fora del dispositiu atacat. Per últim «Cloud» es refereix als serveis cloud que utilitza el dispositiu per implementar algunes de les funcions que ofereix.

La columna «Objectiu» indica un cop perpetrat l'atac amb èxit, quin us acostumen a fer del dispositiu els atacants. En el cas de «controlar el dispositiu» ens indica que l'atacant podrà utilitzar-lo per qualsevol cosa que vulgui fer.

La columna «Ús» dona informació sobre les preferències dels atacants per cada tipus d'atac. Així doncs com es pot veure els atacants utilitzen per sobre de tot mètodes automatitzats i que els permetin accés a un gran nombre de dispositius de forma simultània.

La columna «Atacs reals» resumeix les estadístiques que es poden trobar quant a atacs detectats a dispositius IoT i que no són sols estudis teòrics. Com es pot veure, el nombre d'atacs reals és molt semblant amb les preferències dels atacants.

A l'última columna «Exemple» es posen alguns noms de dispositius o atacs que són coneguts per cada tipus d'atac.

2.5 Proposta de característiques

Les característiques que aquí es mencionen són una proposta teòrica del que es considera que seria una bona aproximació per un sistema HIDS en entorns IOT, tant en dispositius amb sistema operatiu complet com simple.

Gairebé tots els atacs que s'han analitzat involucren **canvis de comportament** del dispositiu, sobretot des del punt de vista de les comunicacions. Així doncs plantejar un HIDS utilitzant la detecció d'anomalies en el tràfic de xarxa (Anomaly-based IDS) sembla un bon plantejament.

Una aproximació interessant pot ser separar la recollida de dades de l'anàlisi i presa de decisions. Així doncs un **agent (AG)** o conjunt d'agents poden recollir dades sobre el tràfic de xarxa en cada un dels dispositius. Depenent del tipus de dispositiu (potencia, capacitats del sistema operatiu, etc.), l'agent pot recollir sols metadades sobre el tràfic o arribar a fer deep inspection.

Alguns dispositius de HUB o Gateways/routers de xarxa, podrien recollir dades no sols del seu tràfic de xarxa, sinó també dels dispositius que gestionen o dels que encaminen el tràfic. Aquest últim punt podria ser molt interessant per no dependre d'una sola font de dades a l'hora d'identificar un possible dispositiu amb comportament anòmal. Aquests agents també podrien recollir dades de xarxes específiques del IoT (Zigbee, Z-Wave, Bluetooth, BLE, etc.) i no sols del tràfic internet.

Els agents han de tenir com a mínim la capacitat de signar la informació que envien, i si és possible també xifrar-la.

Encara que d'entrada els agents publicarien informació sobre el tràfic de xarxa, també seria interessant deixar oberta la possibilitat de publicar qualsevol altra informació dels dispositius que pogués servir per detectar un comportament anòmal, com per exemple: Ús mitjà de la CPU, canvis de contexts en un període de temps, etc.

Un altre component serà el que anomenaríem **«detector d'anomalies» (DA)**, que rebria les dades proporcionades pels agents i aplicaria les regles per tal de detectar el comportament anòmal. Aquest component seria l'encarregat de l'anàlisi de les dades i la presa de decisions, que després podria comunicar a altres components, com poden ser dashboards o sistemes actius de prevenció.

Els DA podrien executar-se en dispositius amb suficients recursos per fer l'anàlisi, però el plantejament és poder fer-ho de forma jeràrquica. Així doncs, els DA més propers (HUB, routers, equip HIDS dedicat) als dispositius IoT poden fer anàlisi més simple o específics, una capa intermèdia (edge o fog)

podria fer una anàlisi més generalitzada i per últim al Cloud es podria fer una anàlisi big data utilitzant el gran volum de dades recollit de tots els dispositius IoT.

La separació en capes pot implicar que els DA tindran diferents prestacions segons la capa i les capacitats on es trobi desplegat.

Una proposta de funcionament pels DA és que utilitzin **«perfils»** de dispositiu per determinar que es considera un comportament anòmal d'aquest. Aquests perfils poden ser creats i publicats tant pels fabricants com per experts que decideixin crear-los independentment dels fabricants. La premissa és que dispositius semblants tenen comportament del tràfic de xarxa semblant. Així doncs un perfil per una bombeta IoT pot indicar que no es comunica amb altres dispositius que no siguin de tipus «HUB», tenir una llista d'IPs valides del núvol on es pot connectar, o un límit del nombre de connexions que pot tenir obertes.

Poden existir perfils «globals» que apliquin a tots els dispositius amb una certa capacitat, com per exemple llistes d'IPs conegudes pel seu ús per part d'atacants, etc. Els dispositius amb capacitat de filtratge (moure més endavant) poden filtrar connexions i així mitigar un possible atac.

Els perfils es podrien distribuir des de les capes més altes (cloud) dels DA i anar baixant en la jerarquia de DA fins a arribar als DA més propers al dispositiu IoT final. Cada DA podria decidir quina part del perfil aplicar depenent de les seves capacitats.

El DA pot tenir la capacitat de «contrastar» les dades sobre un mateix dispositiu que siguin publicades per més d'un agent. Un exemple pot ser que un dispositiu publiqui estadístiques sobre el seu tràfic de xarxa i alhora el router també publiqui les mateixes estadístiques, si no coincideixen pot ser indicatiu que algun dels dos dispositius ha estat manipulat.
Els DA de més alt nivell (núvol) poden tenir la capacitat de detectar atacs o infeccions a gran escala contra un tipus concret de dispositiu, ja que agreguen tota la informació de tots els dispositius IoT que tenen agents desplegats. Així doncs es podria arribar a observar un ús anòmal d'un conjunt de dispositius agregant les dades de tot el grup. Per exemple si la suma d'amplada de banda usada per un grup de dispositius del mateix tipus sobrepassa un cert límit (proporcional), pot indicar que s'està utilitzant els dispositius en una botnet, encara que individualment l'augment d'ús d'amplada de banda no sigui significatiu.

Un altre component opcional podria ser un **«agent de filtrat» (AF),** aquest component es desplegaria en dispositius amb capacitat de firewall, sigui local o de xarxa. Depenent del tipus de dispositiu i dels perfils aplicats, aquest mòdul filtraria connexions a IPs, xarxes o protocols específics.

Tots els routers i molts dispositius IoT amb sistema operatiu complet tenen implementades funcions de firewall al propi kernel, així doncs aquest mòdul podria fer de «configurador» del firewall del kernel i els perfils. Alguns sistemes

operatius simples com ara Contiki disposen d'implementacions de firewall que encara que són molt simples, poden ser viables.

El component de **monitorització (MO)** podria mostrar tant les alertes generades pel DA com les dades en cru, enviades pels AG. En principi aquest mòdul no disposaria de capacitat d'anàlisi, ja que aquesta part la farien el DA, però si funcions de reporting amb capacitat de gestionar grans volums de dades.

Les comunicacions entre els diferents mòduls, estiguin a la capa que estiguin han de ser autenticades, sobretot amb el mòdul AF, ja que aquest té capacitat per realitzar accions disruptives. Caldrà implementar algun mecanisme per què els AG, DA i AF es puguin localitzar entre ells, definir com s'agrega i/o redunda la informació, com es reparteixen les funcions de detecció de cada nivell de DA, etc.

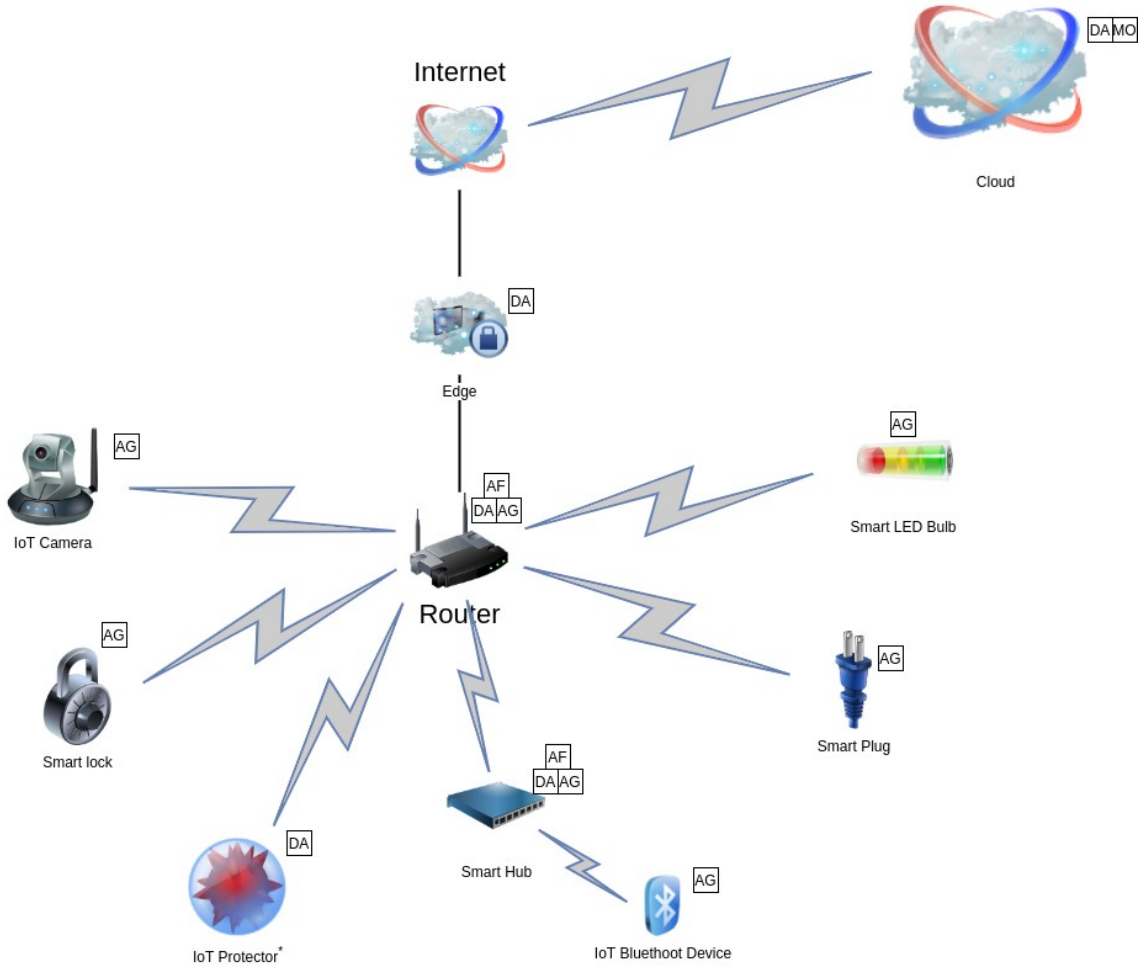


Figura 5: Visió general de l'arquitectura proposada

A l'exemple de la Figura 5 es pot veure com depenent del tipus de dispositiu pot disposar de més d'un mòdul, fins i tot es podria tenir un dispositiu «IoT Protector» amb un DA que pogués ser gestionat per l'usuari final. En l'apartat 4 d'aquest treball es desenvoluparà l'arquitectura de cada servei.

El fet de separar la recol·lecció de dades (data plane) de l'anàlisi i presa de decisions (control plane), permet aplicar noves regles de detecció d'atacs sense que calgui actualitzar els equips IoT finals. També permetria que diferents proveïdors d'accés a internet despleguessin els seus propis AG i AF per tal d'oferir millors prestacions als seus clients.

Existeixen desenes de papers amb propostes per molts dels aspectes proposats en aquest apartat, des de com detectar el tipus de dispositius de forma automàtica, fins com utilitzat l'Edge computing per millorar les capacitats de detecció. En els propers apartats es buscarà dins de l'àmbit de l'open source si existeixen ja creades eines o llibreries que ja implementin del tot o parcialment la proposta detallada aquí.

3. Buscar productes

3.1 Introducció

En aquest apartat es descriuen alguns dels principals HIDS dins l'àmbit de l'open source així com algunes llibreries que ens podrien ser d'utilitat per implementar la proposta que faig a l'apartat 2.5.
No es tracta de fer un estudi a fons sobre cada producte, sinó en veure si existeixen implementacions dels HIDS que s'adaptin a la proposta, així doncs, no es descriuran en profunditat les característiques de cada producte. La idea és buscar producte/s que puguin servir de base per a implementar el sistema HIDS pseudodistribuït que s'ha proposat. Alguns dels productes analitzats es consideren NIDS, però el comentaré igualment atès que la proposta és detectar comportaments anòmals en el tràfic de xarxa des del punt de vista del dispositiu (host)

3.2 HIDS opensource

WAZUH/OSSEC: Wazuh és un fork d'Ossec amb moltes millores, així que es comentarà sols aquest software. La descripció que fa el fabricant del producte és la següent:

«Els principals components de Wazuh són l'agent, el servidor i Elastic Stack.

***L'agent** lleuger de Wazuh està dissenyat per fer una sèrie de tasques amb l'objectiu de detectar amenaces i, quan calgui, activar respostes automàtiques. Pot funcionar en diverses plataformes, incloses Windows, **Linux**, macOS, AIX, Solaris i HP-UX. Aquests agents poden ser configurats i administrats des del servidor de Wazuh.*

*El **servidor** de Wazuh s'encarrega d'analitzar les dades rebudes dels agents, processar els esdeveniments a través de descodificadors i regles i utilitzar la intel·ligència d'amenaces per buscar els coneguts IOC (Indicadors de Compromís). Un sol servidor Wazuh pot analitzar les dades de centenars o milers dagents, i escalar de manera **horitzontal** quan es configura en mode cluster.*

El servidor també s'utilitza per gestionar els agents, configurant-los i actualitzant-los a distància quan calgui. A més, el servidor és capaç d'enviar ordres als agents, per exemple, per activar una resposta quan es detecta una amenaça.

Les alertes generades per Wazuh són enviades a Elasticsearch, on són indexades i emmagatzemades. El plugin Wazuh Kibana proporciona una interfície d'usuari potent per a la visualització i l'anàlisi de dades, que també es pot utilitzar per gestionar i supervisar la configuració i l'estat dels agents.

La interfície d'usuari de la web de Wazuh inclou taulers llestos per utilitzar per al compliment de la normativa (per exemple, PCI DSS, GDPR, CIS), aplicacions vulnerables detectades, supervisió de la integritat dels fitxers, avaluació de la configuració, esdeveniments de seguretat, supervisió de la infraestructura del núvol i altres.»

Wazuh és una solució completa on s'implementen tots els elements per a la gestió d'una xarxa d'agents, així i tot, sorgeixen alguns dubtes pel que fa a l'escalabilitat. Encara que tota la solució i en especial l'agent estan pensats per ser multiplataforma, inclús existeixen paquets per la distribució de raspberry pi, no sembla prou lleuger per a executar-se amb plataformes amb pocs recursos. No s'han trobat indicis de versions «light» de l'agent que es puguin desplegar en dispositius amb SO simples. Cal recalcar que les capacitats d'anàlisis de tràfic de xarxa recauen en components externs com ara suricata.

Un punt interessant de Wazuh és que és modular, i no sembla molt complicat afegir-hi mòduls per implementar noves funcionalitats o protocols. Les alertes que emet poden ser en format JSON, el que és molt idoni per compartir-les amb altres sistemes. El motor de regles, i les regles ja escrites per tal de detectar comportaments sospitosos són molt potents, encara que estan molt focalitzades en l'anàlisi de fitxers de logs. També té un mòdul per generar de forma automàtica regles del firewall per aturar de forma proactiva possibles atacs.

Suricata, és una combinació de funcionalitats que el seu fabricant descriu de la següent manera:

Suricata és el principal motor independent de detecció d'amenaces de codi obert. En combinar la detecció d'intrusions (IDS), la prevenció d'intrusions (IPS), el monitoratge de seguretat de la xarxa (NSM) i el processament PCAP, Suricata pot identificar, aturar i avaluar ràpidament fins i tot els atacs més sofisticats.»

Més enllà de la descripció de marketing que fa el fabricant, es tracta d'un software molt bo per l'anàlisi de tràfic de xarxa, pot detectar tant anomalies del tràfic com signatures dels atacs de xarxa. Pot utilitzar tant les seves signatures com les que proporciona SNORT, però sembla que té millor rendiment que aquest últim.

Suricata no és un software distribuït, així doncs el procés d'anàlisi es fa en el mateix host que fa la captura del tràfic de xarxa. Aquest software té diverses maneres per captura el tràfic de xarxa, però cap d'elles sembla possible implementar-la en dispositius amb SO simple, el fet que s'analitzi el tràfic en el propi host tampoc és útil en dispositius amb poca potència (CPU i memòria).

Té funcions avançades, com ara la detecció del protocol independentment del port on escolti el servei, detecció d'aplicacions, extracció de fitxers, etc. Pot emetre les alertes en format JSON igual que WAZUH.

Snort, és un sistema de detecció d'intrusos en xarxa, lliure i gratuït. Ofereix la capacitat d'emmagatzematge de bitàcoles en fitxers de text i en bases de dades obertes, com MySQL. Implementa un motor de detecció d'atacs i escaneig de ports que permet registrar, alertar i respondre davant de qualsevol anomalia prèviament definida.

Snort es pot configurar en tres modes principals:

1. Sniffer: El programa llegirà paquets de xarxa i els mostrarà a la consola.

2. Registre de paquets: En aquest mode el programa registrarà paquets al disc.

3. Detecció d'intrusions a la xarxa: el programa supervisarà el trànsit de la xarxa i l'analitzarà en funció d'un conjunt de regles definides per l'usuari. A continuació, el programa realitzarà una acció específica en funció del que s'ha identificat.

Aquest software és molt usat per l'anàlisi de tràfic de xarxa i és per aquest motiu que té un conjunt de regles molt extens i que fins i tot utilitzen altres programes com Suricata.

Snort fa la captura i l'anàlisi del tràfic en la mateixa màquina, així doncs no té funcions distribuïdes. Tampoc s'han trobat evidències que pugui funcionar en equips amb pocs recursos, cal tenir en compte que la BBDD de regles/signatures ocupa desenes de MB.

Zeek (anteriorment Bro), és un marc d'anàlisi de xarxa de programari lliure i de codi obert. Està dissenyat per ser un monitor de seguretat de xarxa (NSM), però també es pot utilitzar com a sistema de detecció d'intrusions de xarxa (NIDS) juntament amb una anàlisi en directe addicional d'esdeveniments de xarxa. Zeek és un conjunt d'eines i scripts molt modular i configurable.

Zeek captura el tràfic de xarxa i el passa a un motor d'esdeveniments que els accepta o refusa, els acceptats es passen a un script de polítiques.
El motor d'esdeveniments analitza el trànsit de xarxa en directe per generar esdeveniments neutrals. Genera esdeveniments quan passa «alguna cosa», pot significar activitat del mateix sistema (arrancar o parar el servei, llegir la configuració, etc.) com que s'ha trobat alguna cosa (nova connexió TCP o HTTP) en el tràfic de xarxa que s'està analitzant. Zeek utilitza ports comuns i detecció de protocols dinàmics (que inclou signatures i anàlisi de comportament) per fer una millor estimació a l'hora d'interpretar els protocols de xarxa. Els esdeveniments són políticament neutrals, ja que no són bons ni dolents, sinó que simplement indiquen que ha passat alguna cosa.

Els esdeveniments es gestionen mitjançant scripts de polítiques, que s'analitzen per crear polítiques d'acció. Per defecte, Zeek simplement registra informació sobre esdeveniments als fitxers, tanmateix, es pot configurar per dur a terme altres accions com ara enviar un correu electrònic, generar una alerta, executar una ordre del sistema, actualitzar una mètrica interna i fins i tot cridar a un altre script Zeek.

Tot i que l'arquitectura de Zeek és molt modular i que suporta el funcionament en mode cluster, no és una solució distribuïda i sols permet l'anàlisi de grans volums de tràfic distribuint-lo entre els diferents nodes. És molt interessant el sistema d'scripts per l'anàlisi dels esdeveniments, ja que permet configurar noves funcions sols amb simples scripts del seu propi llenguatge.

El sistema de recollida del tràfic de xarxa està basat en PCAP, que sense cap modificació no sembla adequat per sistemes amb pocs recursos.

Samhain, proporciona comprovació de la integritat dels fitxers i el seguiment/anàlisi dels fitxers de registre, així com la detecció de rootkits, la supervisió de ports, la detecció d'executables SUID delictius i processos ocults.

Samhain ha estat dissenyat per supervisar múltiples hosts amb sistemes operatius potencialment diferents, proporcionant registre i manteniment centralitzats, encara que també es pot utilitzar com a aplicació autònoma en un sol host. Samhain és una aplicació multiplataforma de codi obert per a sistemes POSIX (Unix, Linux, Cygwin/Windows).

Tal com la seva descripció indica (per sistemes POSIX) Samhain és un HIDS amb la configuració i motorització centralitzada igual que WAZUH/OSSEC, i com aquests, no està pensat per ser usat en dispositius amb pocs recursos.

Sagan, és un motor de correlació i anàlisi de registres en temps real de codi obert, multifils, d'alt rendiment i que s'executa en sistemes operatius Unix. Està escrit en C i utilitza una arquitectura multifil per oferir un registre d'alt rendiment i anàlisi d'esdeveniments. L'estructura i les regles de Sagan funcionen de

manera similar al motor Sourcefire Snort IDS/IPS. Això permet que Sagan sigui compatible amb els programaris de gestió de regles Snort o Suricata i li permeti relacionar-se amb les dades IDS/IPS de Snort.

Sagan admet diferents formats de sortida per a informes i anàlisis, normalització de registres, execució d'scripts en detecció d'esdeveniments, detecció/alerta de GeoIP i alertes sensibles al temps.

Aquesta eina és molt semblant a Snort o Suricata i el seu funcionament, avantatges i inconvenients són els mateixos que aquestes.

Existeixen conjunts d'eines com ara «Security Onion» o «Sguil» que incorporen alguns del software mencionats en aquest apartat amb algunes millores o sistemes que permetin el funcionament conjunt per tal d'oferir millors resultats.

Existeixen alguns papers de HIDS especificadament dissenyats pel món IoT, com ara «SVELTE intrusion detection system»[37] o «SENTINEL framework»[38], però ha estat impossible trobar cap implementació funcional.

Tal com es pot veure, no existeix un producte amb les funcionalitats descrites en l'apartat 2.5 d'aquest document, però pot ser possible utilitzar alguns d'aquests softwares com a DA i MO.

3.3 Llibreries de captura

Libpcap, és una de les llibreries més usades per la captura de tràfic de xarxa, i s'utilitza sobretot en sistemes UNIX. També té funcions de filtrar i un format (PCAP) binari per a emmagatzemar el tràfic capturat en fitxers. És una llibreria molt potent que abstrau les interaccions amb el SO i el hardware, però no està pensada per funcionar en dispositius amb pocs recursos.

AWS IoT Device Defender, és un component del framework AWS IoT Device SDK, que facilita el desenvolupament de dispositius IoT utilitzant els serveis al núvol d'AWS. D'acord amb la descripció d'AWS: *«AWS IoT Device Defender és un servei de seguretat que us permet auditar la configuració dels vostres dispositius, supervisar els dispositius connectats per detectar comportaments anormals i mitigar els riscos de seguretat.»*
En el nostre cas no estem interessats a utilitzar els serveis d'AWS, però aquest framework disposa de components opensource que ens poden ser d'utilitat. Les implementacions per defecte (python i C) permeten capturar informació sobre el tràfic IP, així com algunes mètriques (CPU) no relacionades amb la xarxa. Cal esmentar que aquesta llibreria està inclosa (https://www.freertos.org/iot-device-defender/index.html) en el codi de Freertos (també propietat d'AWS) i pot publicar informació sobre la pila IP de dispositius que executin aquest SO. Encara que la llibreria està pensada per usar mqtt com a protocol de comunicacions, no obliga a utilitzar aquest protocol i es pot integrar fàcilment amb qualsevol altre.

Existeixen multitud de sniffers de tràfic de xarxa i també eines que «exporten» informació i estadístiques de la pila IP, però gairebé la totalitat d'elles requereixen un SO complet basat en UNIX o Windows.

3.4 Llibreries de comunicacions

Mqtt, és un protocol de missatgeria de publicació/subscripció dissenyat per a comunicacions M2M lleugeres en xarxes amb pocs recursos. El client MQTT publica missatges a un broker MQTT al que estan subscrits altres clients. Cada missatge es publica a una adreça, coneguda com a «topic» (tema). Els clients poden subscriure's a diversos «topics» i reben tots els missatges publicats a cada tema. MQTT és un protocol binari i normalment requereix una capçalera fixa de 2 bytes amb càrregues de missatges petits, fins a una mida màxima de 256 MB. Utilitza TCP com a protocol de transport i TLS/SSL per a la seguretat. Per tant, la comunicació entre client i intermediari està orientada a la connexió. Una altra característica de MQTT són els seus tres nivells de qualitat de servei (QoS) per a un lliurament fiable de missatges.
MQTT és adequat per a grans xarxes de dispositius amb pocs recursos que s'han de supervisar o controlar des d'un servidor. No està dissenyat per a la transferència de dispositiu a dispositiu, ni per a dades multicast a molts receptors. És un protocol de missatgeria molt bàsic que ofereix només unes poques opcions de control. Existeixen versions especialitzades del protocol com ara les que funcionen sobre UDP o MQTT-SN (MQTT for Sensor Networks).

CoAP, és un protocol M2M lleuger que admet tant l'arquitectura de sol·licitud/resposta com de recursos/observació (una variant de publicació/subscripció). CoAP està desenvolupat sobretot per interoperar amb HTTP i el web RESTful mitjançant servidors intermediaris simples. A diferència de MQTT, CoAP utilitza l'identificador de recursos universal (URI) en lloc de «topics». L'editor publica dades a l'URI i el subscriptor se subscriu a un recurs concret indicat per l'URI. Quan un editor publica dades noves a l'URI, tots els subscriptors reben una notificació sobre el nou valor tal com indica l'URI. CoAP és un protocol binari i normalment requereix una capçalera fixa de 4 bytes amb càrregues de missatges petits fins a una mida màxima depenent del servidor web o de la tecnologia de programació. CoAP utilitza UDP com a protocol de transport i DTLS per a la seguretat. Així, clients i servidors es comuniquen mitjançant datagrames sense connexió amb menys fiabilitat. Tanmateix, utilitza missatges «confirmables» o «no confirmables» per proporcionar dos nivells diferents de QoS. On, els missatges confirmables han de ser reconeguts pel receptor amb un paquet ACK i els missatges no confirmables no. CoAP ofereix més funcionalitats que MQTT, ja que admet la negociació de contingut.

AMQP, és un protocol de missatgeria corporatiu dissenyat per a la fiabilitat, la seguretat, i la interoperabilitat. AMQP admet tant l'arquitectura de sol·licitud/resposta com de publicació/subscripció. Ofereix una àmplia gamma de funcions relacionades amb la missatgeria, com ara una cua fiable,

missatgeria de publicació i subscripció basada en «topics», encaminament flexible i transaccions. El sistema de comunicació AMQP requereix que l'editor o el consumidor creï un «intercanvi» amb un nom donat i després emeti aquest nom. Els editors i els consumidors utilitzen el nom d'aquest intercanvi per descobrir-se. Després, un consumidor crea una "cua" i l'adjunta a l'intercanvi alhora. Els missatges rebuts per l'intercanvi s'han de relacionar amb la cua mitjançant un procés anomenat "enllaç". AMQP intercanvia missatges de diverses maneres: directament, en forma de fanout, per tema o en funció de les capçaleres. És un protocol binari i normalment requereix una capçalera fixa de 8 bytes amb càrregues útils de missatges petits fins a una mida màxima depenent del broker/servidor o de la tecnologia de programació. AMQP utilitza TCP com a protocol de transport predeterminat i TLS/SSL i SASL per a la seguretat. Així, la comunicació entre el client i el corredor està orientada a la connexió. La fiabilitat és una de les característiques bàsiques d'AMQP i ofereix dos nivells de Qualitat de Servei (QoS).

HTTP, és sobretot un protocol de missatgeria web, que va ser desenvolupat originalment per Tim Berners-Lee. Més tard, va ser desenvolupat per l'IETF i el W3C conjuntament i es va publicar per primera vegada com a protocol estàndard el 1997. HTTP admet l'arquitectura web RESTful de sol·licitud/resposta. De manera anàloga a CoAP, HTTP utilitza l'identificador de recursos universal (URI) en lloc dels «topics». El servidor envia dades a través de l'URI i el client rep dades a través d'un URI particular. HTTP és un protocol basat en text i no defineix la mida de les càrregues útils de la capçalera i dels missatges, sinó que depèn del servidor web o de la tecnologia de programació. HTTP utilitza TCP com a protocol de transport predeterminat i TLS/SSL per a la seguretat. Així, la comunicació entre client i servidor està orientada a la connexió. No defineix explícitament QoS, però ofereix altres funcions, com ara connexions persistents, canalització (tunneling) de sol·licituds i codificació de transferència fragmentada.

Tant MQTT com amb AMQP es poden muntar estructures multinivell on el «broker» es connecten amb altres «broker», de forma que es poden crear grans xarxes de dispositius.

4. Arquitectura

La proposta exposada a l'apartat 2.5 implica un model distribuït de components software que es comuniquen entre ells de forma bidireccional. Donat que alguns dels dispositius que executaran els agents disposen de molt pocs recursos i en alguns casos no disposen de SO complet, cal utilitzar un protocol de comunicacions «lleuger». A l'arquitectura proposada s'utilitzarà MQTT, ja que es força lleuger, està molt implantat en el món IoT, té moltes implementacions i permet desplegaments amb diferents topologies.

MQTT té com a component central l'anomenat broker, que és l'encarregat de rebre els missatges enviats pels publicadors, i reenviar-los als subscriptors. Depenent de la implementació i configuració, el broker pot realitzar moltes altres funcions, com ara persistència dels missatges, QOS, comunicacions inter- broker, etc.

El concepte de broker pot donar a entendre que aquest protocol té una topologia centralitzada, però en realitat els brokers són prou lleugers per a executar-se conjuntament amb el software client.

A la proposta del punt 2.5 podem distingir 4 tipus de components: agent (AG), «detector d'anomalies» (DA), «agent de filtrat» (AF), monitorització (MO)

L'agent (AG) és l'encarregat de recollir les dades del dispositiu, és un component lligat al hardware IoT i al SO, ja que requeries interactuar amb el sistema operatiu per tal de recollir les dades que després enviarà als detectors d'anomalies. Amb dispositius amb recursos pot fer ús de llibreries com libpcap per recollir informació més detallada (deep inspection) del tràfic que rep el dispositiu. Un cop recollida la informació s'enviarà utilitzant el protocol MQTT a un o més DA's depenent de la configuració. Cada dispositiu disposarà d'un topic MQTT per enviar la informació, així doncs el DA podrà distingir la font de la informació sols veient per quin topic entra. Addicionalment existirà un topic que l'agent podrà llegir i per on el DA comunicarà a l'agent la informació que requereixi, com per exemple la configuració addicional del perfil (si existeix).

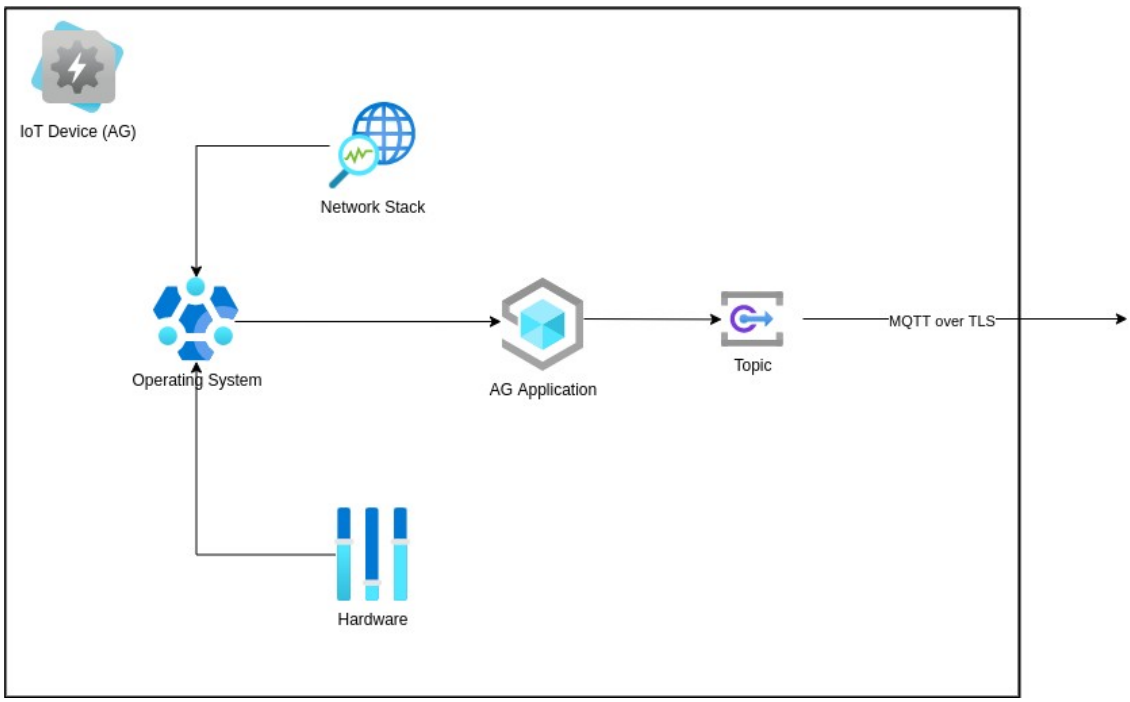


Figura 6: Arquitectura de l'agent (AG)

Des d'un punt de vista de serveis software l'aplicació de l'AG disposaria de 4 serveis, Configuració, Missatgeria, Recol·lector, Seguretat i un bucle principal que anomenaré Controller. En el següent diagrama es poden veure les interaccions entre els diferents serveis:

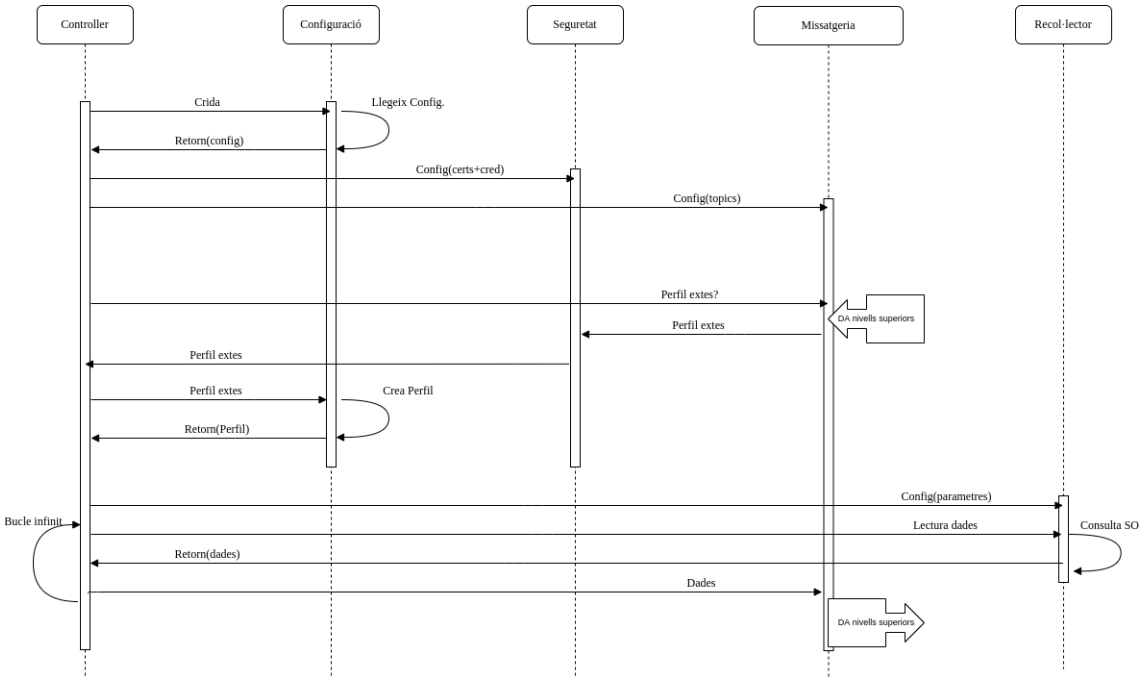


Figura 7: Diagrama de serveis del AG

Com es pot veure a diagrama de la Figura 7, el Controlador crida al servei de Configuració per obtenir la configuració inicial (certificats, credencials, adreces del DA de nivell superior, perfil base, etc.) Aquesta configuració pot ser la de «fabrica» o la guardada l'últim cop que es va reconfigurar l'AG.
Amb aquesta configuració es poden inicialitzar els serveis de Seguretat i Missatgeria, aquests dos serveis permeten sol·licitar (mitjançant el servei de missatgeria) al/s DA de nivell superior versions esteses del perfil, el mòdul de seguretat verificarà la integritat de la informació rebuda mitjançant firma digital amb certificats.

Amb el perfil base i les extensions rebudes del/s DA de nivell superior, el servei de configuració generarà el perfil definitiu que servirà tant per configurar (temps entre bucles, validacions, etc.) el bucle principal del Controller com el Recol·lector de dades.

El bucle principal es limita a repetir sempre el mateix, crida al servei de Recol·lecció de dades, i després crida al servei de Missatgeria per enviar els resultats al DA o altres mòduls interessats.

El DA és el component més complex de l'arquitectura, ja que serà l'encarregat de rebre tota la informació dels dispositius, analitzar-la i decidir si cal notificar o prendre accions per mitigar un atac. En l'arquitectura proposada el DA inclourà el broker MQTT, i podrà tant llegir la informació com reenviar-la a DA de més nivell. Un altre component del DA serà l'encarregat de llegir la informació dels diferents tòpics, analitzar i publicar en un altre tòpic les possibles alertes. Depenent de tipus de DA, caldrà que pugui guardar informació històrica dels dispositius per tal de prendre decisions a partir del comportament històric, o també entrenar models ML per la detecció de les anomalies. Els DA d'alt nivell (cloud, edge o fog) recol·lectaran dades a gran escala i faran anàlisi amb informació agregada de dispositius que pertanyen a clients diferents, aquest podran publicar alertes a tòpics que després podran ser llegits per als DA descendents. Un dels components del DA serà el motor de regles, que serà l'encarregat de processar la informació i decidir si el comportament del dispositiu o grup de dispositius és l'esperat.

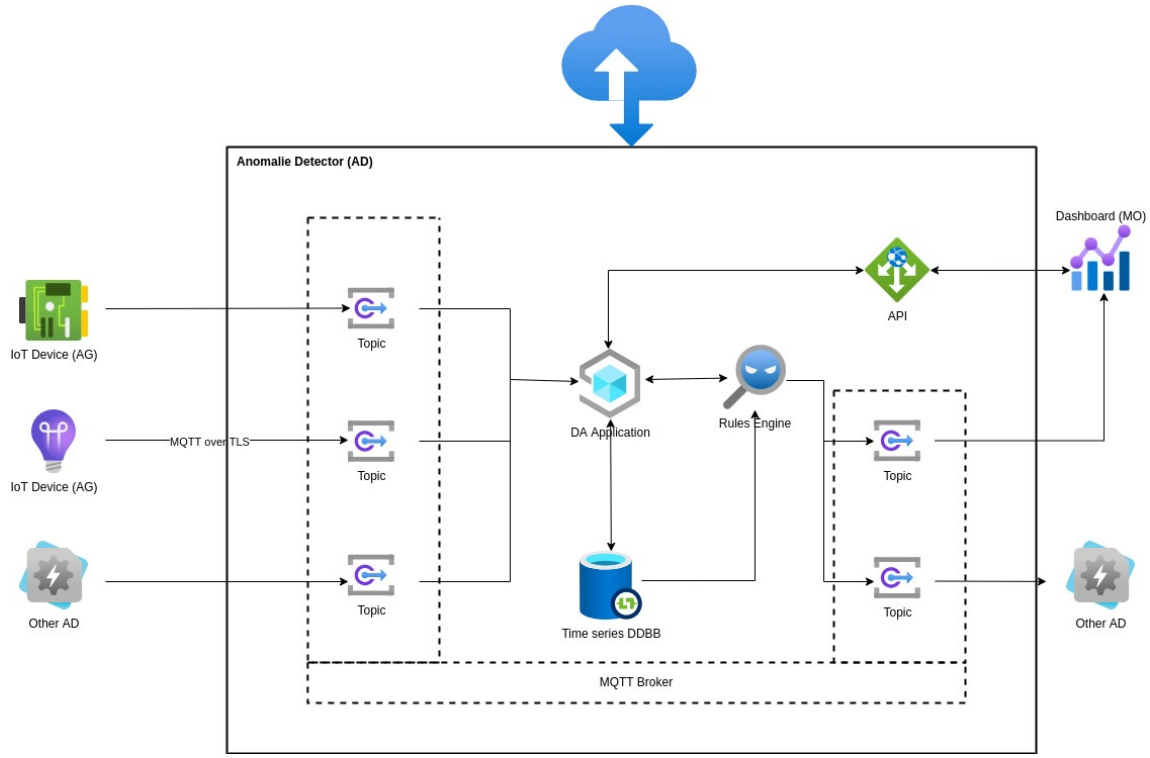


Figura 8: Arquitectura del detector d'anomalies (DA)

Des d'un punt de vista de serveis software l'aplicació del DA disposaria de 5 serveis, Configuració, Missatgeria, Anàlisis, Seguretat, Storage, API i un bucle principal que anomenaré Controller. En el següent diagrama es poden veure les interaccions entre els diferents serveis:

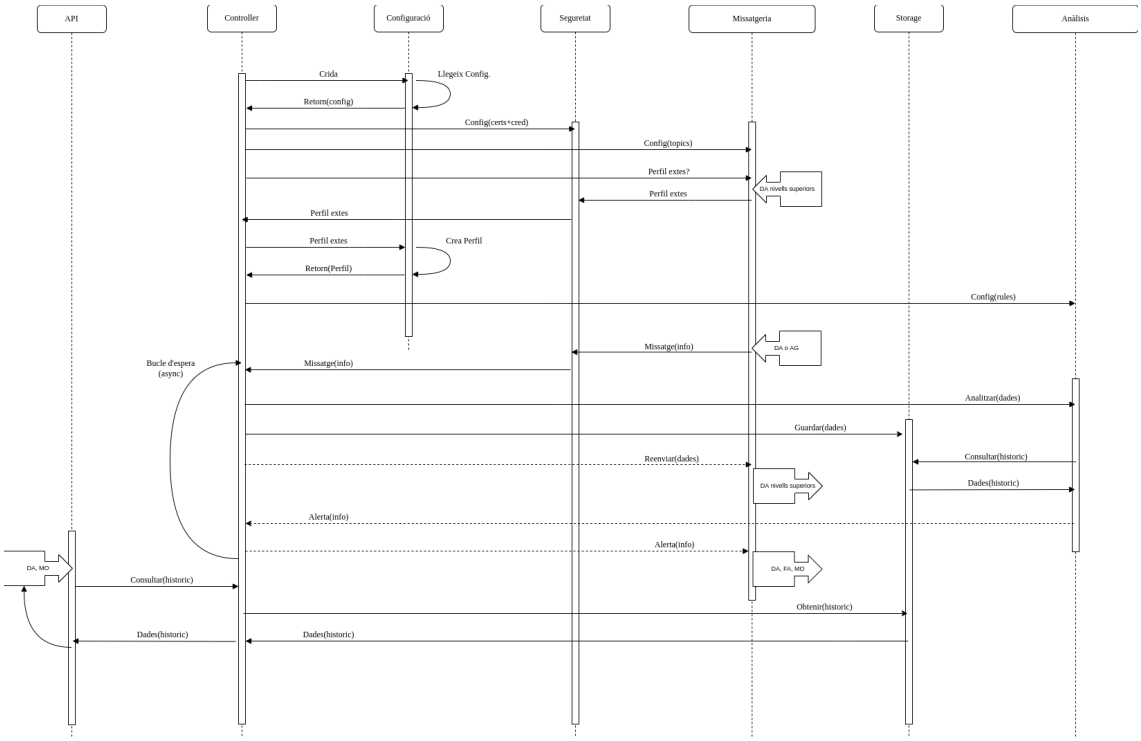


Figura 9: Diagrama de serveis del DA

Com es pot veure al diagrama de la Figura 9, el Controlador crida al servei de Configuració per obtenir la configuració inicial (certificats, credencials, adreces del DA de nivell superior, perfil base, etc.). Aquesta configuració pot ser la de «fabrica» o la guardada l'últim cop que es va reconfigurar l'DA.
Amb aquesta configuració es poden inicialitzar els serveis de Seguretat i Missatgeria, aquests dos serveis permeten sol·licitar (mitjançant el servei de missatgeria) al/s DA de nivell superior versions esteses del perfil, el mòdul de seguretat verificarà la integritat de la informació rebuda mitjançant firma digital amb certificats.

Amb el perfil base i les extensions rebudes del/s DA de nivell superior, el servei de configuració generarà el perfil definitiu que servirà per configurar el servei d'anàlisis amb les regles disponibles per aquest DA. El servei d'anàlisis serà un motor de regles que ens servirà per detectar el comportament anòmal dels dispositius a partir de les dades recollides.

El bucle (d'espera) és asíncron, en rebre noves dades dels AG o notificacions dels DA reaccionarà segons el tipus de missatge. Si rep dades dels AG, executarà tres tasques:

- Enviar les noves dades al servei d'anàlisis, aquest servei pot consultar el servei de storage per tal d'obtenir dades històriques i prendre decisions segons dades guardades, com per exemple l'evolució de la càrrega de la CPU. Les regles configurades en el procés inicial, seran les encarregades de determinar si cal aixecar una alerta. En cas afirmatiu el controller notificarà a tots el topic configurats l'alerta amb la informació necessària perquè els AF puguin prendre contramesures si s'escau. El MO poden usar aquesta informació per mostrar alertes en temps real i el DA per disposar de més dades en fer les anàlisis.

- Guardar les dades mitjançant el servei d'storage, perquè puguin ser consultades pel servei d'anàlisis.

- Reenviar les dades al DA de nivell superior que indiqui el perfil, això podrien ser per exemple DA en el cloud que agregin dades de milions de dispositius.

El servei d'API pot rebre consultes per tal de recuperar dades històriques guardades en el mateix DA. El principal usa ser per part del MO, que usaran aquesta API per obtenir les dades que després mostraran als dashboards d'usuari.

L'AF és un component que llegirà dels topics publicats pels DA la informació sobre els atacs detectats. Depenent del tipus de missatge i la configuració, generarà regles pel firewall local per poder mitigar els possibles atacs.
Des d'un punt de vista de serveis software l'aplicació de l'AG disposaria de 4 serveis, Configuració, Missatgeria, Recol·lector, Seguretat i un bucle principal que anomenaré Controller. En el següent diagrama es poden veure les interaccions entre els diferents serveis:

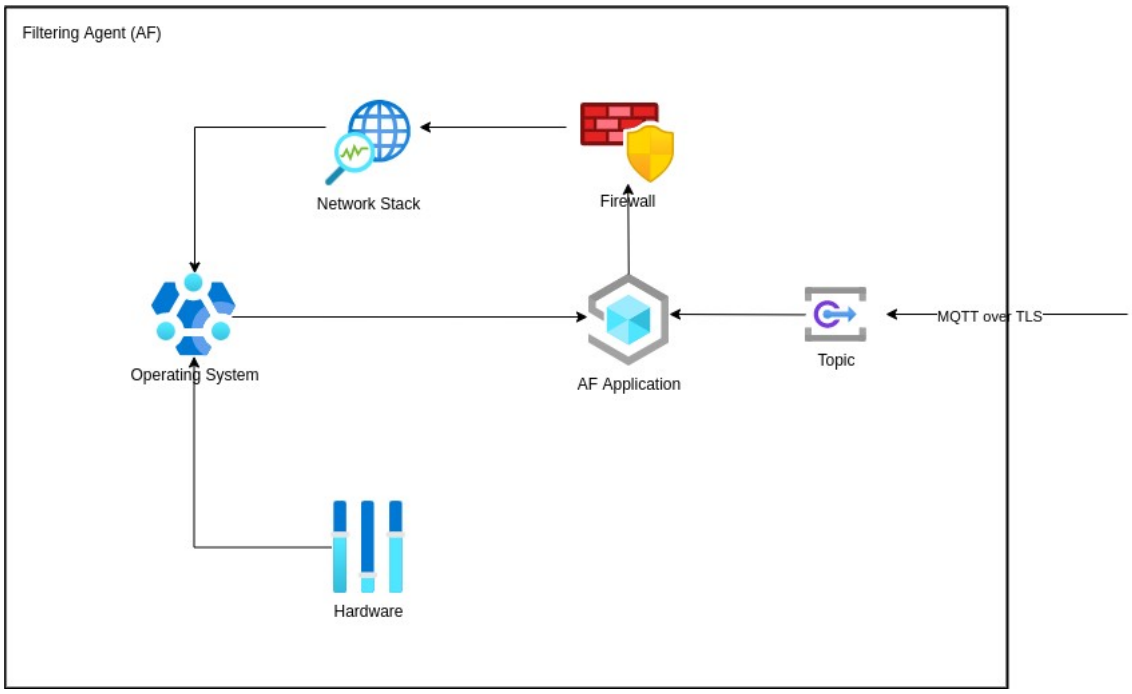


Figura 10: Arquitectura de l'agent de filtrar (AF)

Des d'un punt de vista de serveis software l'aplicació de l'AF disposaria de 4 serveis, Configuració, Missatgeria, Firewall, Seguretat i un bucle principal que anomenaré Controller. En el següent diagrama es poden veure les interaccions entre els diferents serveis:

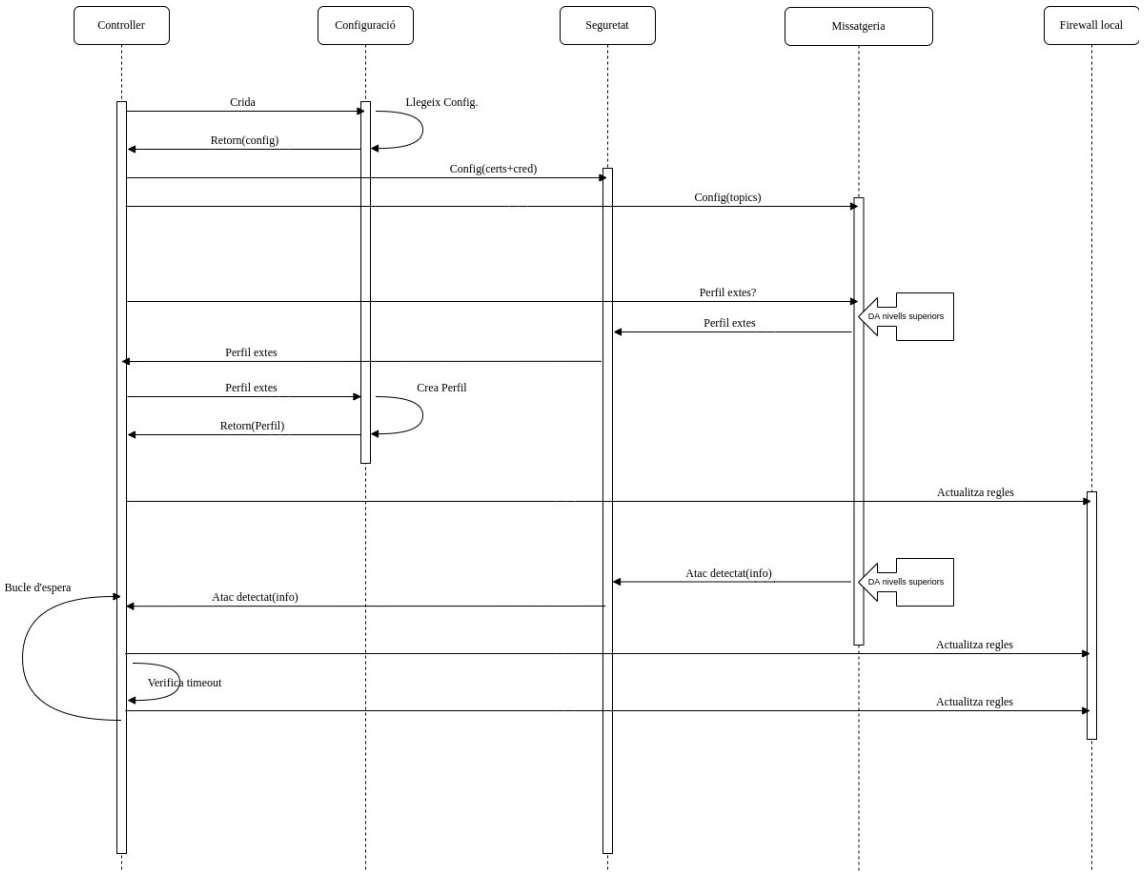


Figura 11: Diagrama de serveis del AF

Com es pot veure al diagrama de la Figura 11, el Controlador crida al servei de Configuració per obtenir la configuració inicial (certificats, credencials, adreces del DA de nivell superior, perfil base, etc.). Aquesta configuració pot ser la de «fabrica» o la guardada l'últim cop que es va reconfigurar l'AF.
Amb aquesta configuració es poden inicialitzar els serveis de Seguretat i Missatgeria, aquests dos serveis permeten sol·licitar (mitjançant el servei de missatgeria) al/s DA de nivell superior versions esteses del perfil, el mòdul de seguretat verificarà la integritat de la informació rebuda mitjançant firma digital amb certificats.

Amb el perfil base i les extensions rebudes del/s DA de nivell superior, el servei de configuració generarà el perfil definitiu que servirà tant per configurar (timeouts de regles, atacs a mitigar, etc.) el bucle d'espera del Controlador, com la configuració inicial del Firewall (IP whitelist, ports prohibits, etc.).

El bucle d'espera es limita a esperar notificacions d'atacs que publicaran els DA als topics que l'AF estigui configurat (descrits al perfil) per escoltar. En base a les dades del perfil i les capacitats del dispositiu el Controlador cridarà al servei de Firewall que generarà noves regles pel firewall local del dispositiu. El bucle d'espera també configurar els timeouts necessaris per fer que les regles de denegació d'alguns atacs no siguin permanents, com es veu al diagrama en saltar el timeout el Controlador crida al servei de Firewall per actualitzar les regles.

El MO és un component que podrà tant llegir els missatges de tots el topics com la informació històrica que emmagatzemin alguns del DA. Mostrarà de forma gràfica tant les alertes publicades pels DA com els últims valors enviats pels AG.

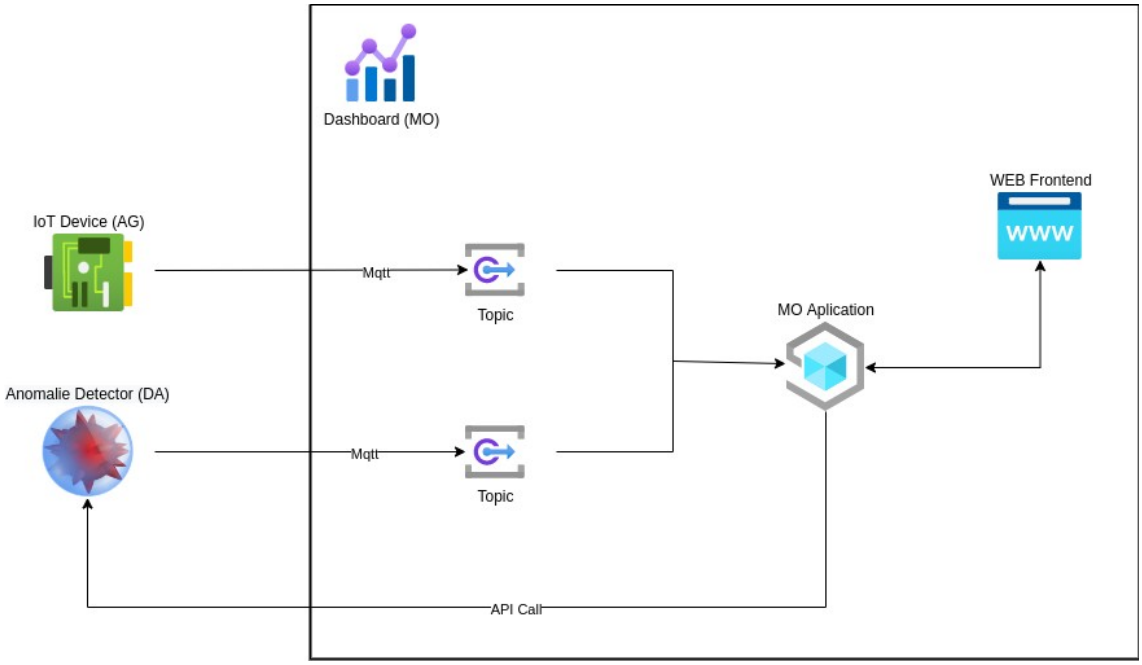


Figura 12: Arquitectura de la monitorització (MO)

Des d'un punt de vista de serveis software l'aplicació de l'MO disposaria de 5 serveis, Configuració, Missatgeria, Frontend WEB, Recol·lector de dades, Seguretat i un bucle principal que anomenaré Controller. En el següent diagrama es poden veure les interaccions entre els diferents serveis:

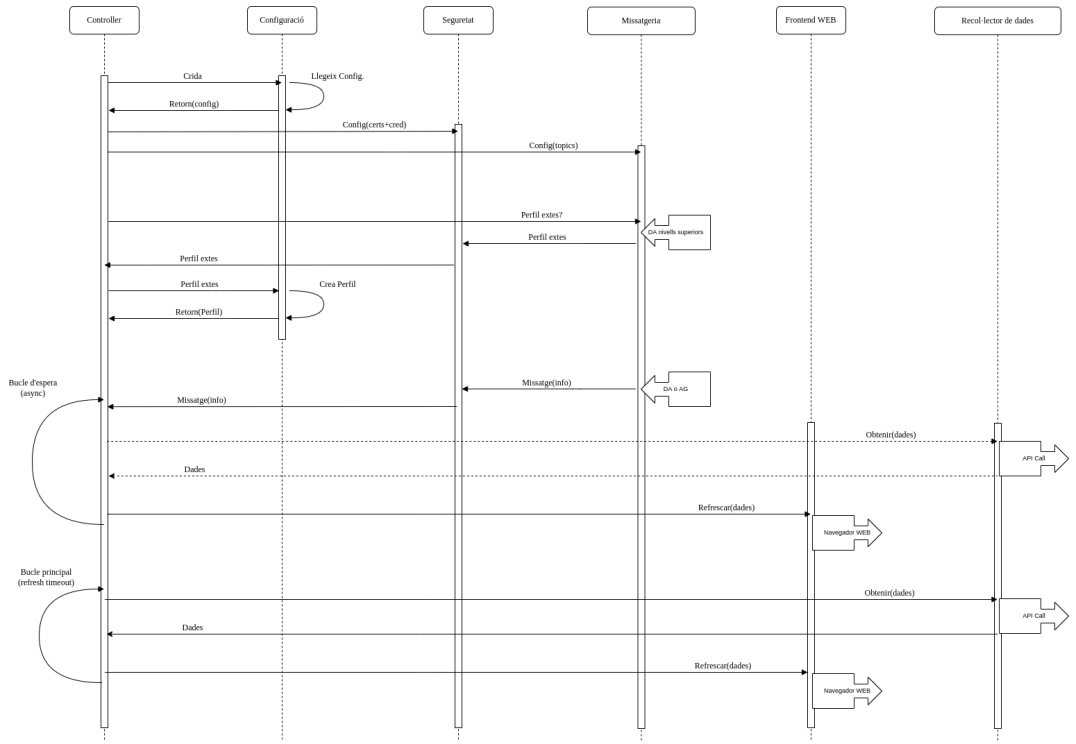


Figura 13: Diagrama de serveis del MO

Com es pot veure al diagrama de la Figura 13, el Controlador crida al servei de Configuració per obtenir la configuració inicial (certificats, credencials, adreces del DA de nivell superior, perfil base, etc.). Aquesta configuració pot ser la de «fabrica» o la guardada l'últim cop que es va reconfigurar l'MO.
Amb aquesta configuració es poden inicialitzar els serveis de Seguretat i Missatgeria, aquests dos serveis permeten sol·licitar (mitjançant el servei de missatgeria) al/s DA de nivell superior versions esteses del perfil, el mòdul de seguretat verificarà d'integritat de la informació rebuda mitjançant firma digital amb certificats.

Amb el perfil base i les extensions rebudes del/s DA de nivell superior, el servei de configuració generarà el perfil definitiu que servirà tant per configurar (topics a escoltar, temps de refresc dels dashboards, etc.) el bucle d'espera del Controlador, com la configuració inicial del Monitor.

El controlador d'aquest mòdul disposa de dos «bucles», el principal s'encarrega de refrescar les dades cada cop que s'arriba al temps configurat entre actualitzacions del dashboard. Per fer això, farà crides al servei de recol·lecció de dades i després al servei que comunica amb el navegador WEB per indicar-li que recarregui les dashboard amb les noves dades. Aquest bucle també s'executarà el primer cop que s'accedeixi al frontal WEB i cada cop que l'usuari sol·liciti una actualitza instantània de les dades.

El segon bucle (d'espera) és asíncron, espera rebre informació publicada pels DA o els AG per actualitzar els dashboard. Opcional-ment pot ser necessari cridar al mòdul de recol·lecció de dades. Els dashboard es poden configurar per mostrar dades en temps real o alertes que es rebran de forma puntual.

5. PoC

5.1 Objectius del PoC

L'objectiu és demostrar que l'arquitectura proposada es pot implementar en sistemes amb pocs recursos (raspberry pi) i que permetria desenvolupar totes les prestacions proposades al punt 2.5 .

Per fer això es demostrarà un cas d'ús: mitjançant una raspberry pi i un laptops, es desplegarà un AG que enviarà dades cada minut a un DA, que les emmagatzemarà i analitzarà. Es configurarà una regla per detectar connexions TCP de sortida cap al port 22 (ssh) per part dels AG, ja que es considera que un dispositiu IoT final mai obrirà connexions d'aquest tipus. Un cop es detecti aquesta connexió anòmala, l'DA emetrà una alarma que serà llegida per un AF que generarà una regla de firewall per tal de bloquejar les connexions al port 22 amb origen al dispositiu IoT que intenta fer connexions ssh cap a altres dispositius.

5.2 Disseny de la implementació

La proposta de característiques feta al punt 2.5 i l'arquitectura descrita al punt 4 permeten desenvolupar un PoC per tal de demostrar que és viable crear aquest software. Com es pot veure al punt 3.2 no existeix cap software ja creat que tingui una arquitectura com la que s'ha descrit, i d'alguna manera tampoc les funcionalitats.

No és possible, amb el temps de què es disposa, crear un software des de zero, així doncs es parteix d'una solució de gestió de dispositius IoT anomenada OpenRemote. Aquest tipus de software s'anomenen IoT device managers, i normalment ofereixen funcions de gestió de dispositius (provisió, monitoratge, reporting) i dasboards per mostrar i actuar sobre els dispositius IoT. Molts d'aquest software també ofereixen altres serveis com per exemple notificacions, motors de regles i data collection. Existeixen uns quants softwares d'aquest tipus, molt opensource però no free, i molts altres sols amb opció SaaS Cloud.

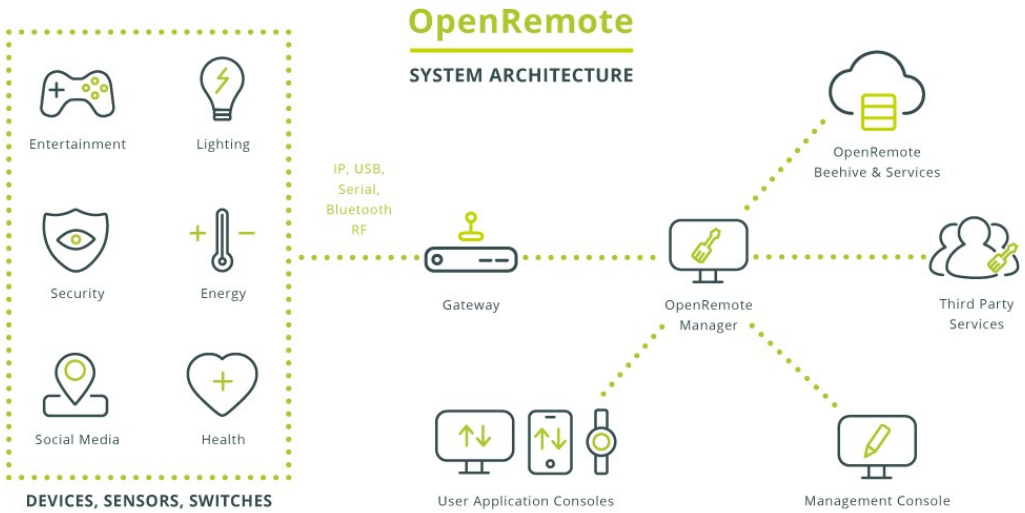


Figura 14: OpenRemote System Architecture

El motiu per triar OpenRemote com a base del PoC, és per què és opensource i free, té opció de desplegar-lo en servidors premises, suporta el protocol MQTT i inclou molts dels elements que calen per desenvolupar tant els DA's com el MO pel PoC.

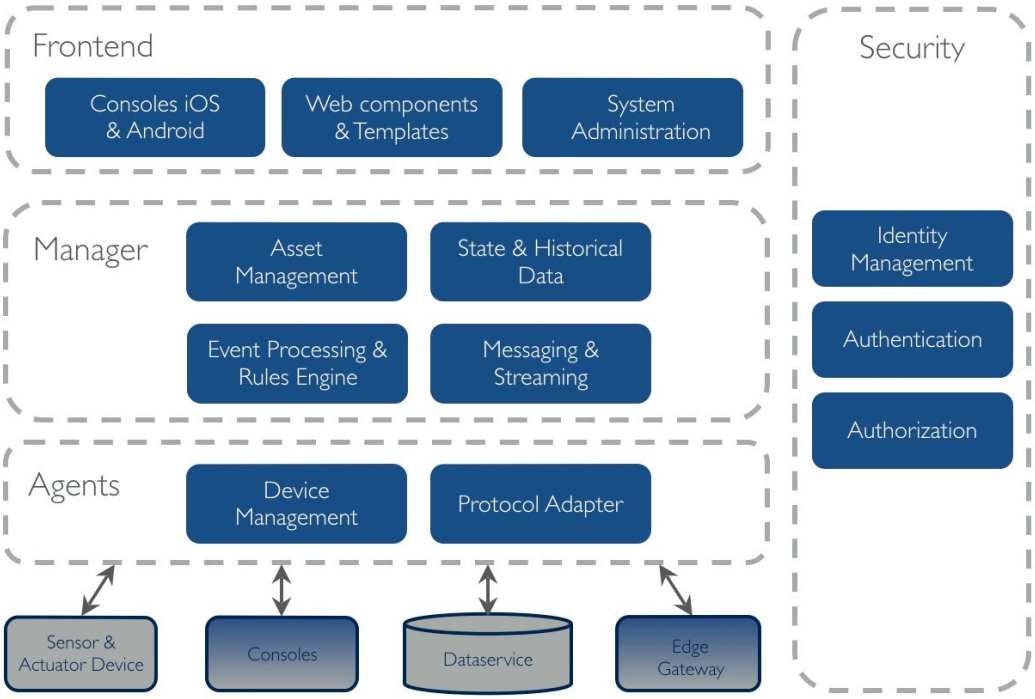


Figura 15: OpenRemote Components

Com es pot veure OpenRemote disposa d'una sèrie de components que es poden usar per a la nostra plataforma.

Messagins & Streaming: és l'encarregat de rebre els missatges amb informació dels sensors mitjançant els diferents protocols, en el PoC s'usarà MQTT, però en suporta molts altres, molts d'ells específics del món IoT. Utilitzarem aquest servei per connectar els nostres AG amb el broker MQTT que està inclòs com a part d'OpenRemote.

Asset Management: s'encarrega de mantenir una BBDD amb els atributs dels assets, les jerarquies, i el model de dades per cadi tipus d'Asset. Al PoC els Assets seran els AG, AF i DA's. En aquest servei es catalogaran els dispositius IoT segons les funcionalitats que ofereixin i es mantindran les configuracions.

State and History data: Aquest servei s'encarrega de recollir les dades enviades pels dispositius i enregistra-les a la BBDD. Normalment es poden guardar les dades en format time series i d'aquesta manera es té l'històric dels valors. Aquest servei enregistrarà les dades en cru que es rebin dels AG's i permetrà després als AG que ho requereixin accedir a l'històric de dades per tal de fer les anàlisis corresponents.

Event Processing and Rules engine: És un motor de regles que permet fer operacions amb les dades rebudes. Tot i que en el nostre PoC l'anàlisi de les

dades es farà utilitzant DA externs, algunes de les regles es podrien implementar mitjançant aquest motor de regles.

Web Components and Templates: OpenRemote incorpora una aplicació web que permet gestionar i monotonitzar els dispositius IoT. En el nostre PoC utilitzarem alguns dels components ja dissenyats per mostrar les dades de les alarmes, així com gràfics amb les mesures rellevants que ens envien els AG.

Security: OpenRemote utilitza una versió reduïda de Keycloak per l'autenticació i autorització, així com la gestió d'usuaris i roles. S'utilitzarà el servei per gestionar els comptes, tant els de servei com els d'usuaris web.

Els **Agents** d'OpenRemote (no confondre amb els AG de la nostra proposta) són els encarregats de rebre i enviar els missatges utilitzant els **Protocol Adapters**. En el PoC s'utilitzarà el MQTT Broker que incorpora i així no caldrà desplegar un broker extern.

AF: Al PoC, com que es farà sobre una raspberry pi amb SO complet (Linux) es generaran regles pel firewall iptables. El software es desenvoluparà en python3 i utilitzarà la llibreria paho-mqtt per la comunicació amb el broker MQTT.

AG: Utilitzarem una versió modificada del «AWS IoT Device Defender» per a recollir i enviar els paràmetres que volem, concretament la que està inclosa en el SDK de python AWS IoT Device Defender Agent SDK (Python). Com que es un component extremadament senzill, no es desenvoluparà software específic, s'utilitzarà un client MQTT de consola molt lleuger anomenat Mosquitto.

DA: es desenvoluparà en python3 i utilitzarà la llibreria paho-mqtt per la comunicació amb el broker MQTT. Com que sols implementarem un regla per detectar una anomalia (conexio ssh sortint), s'implementarà com a una funció de python.

La instància d'**OpenRemote (OR)** correra en un docker al mateix laptop que executarà el DA, ja que es considera que en una implementació real, les funcions que ens ofereix les proporcionaria el mateix DA.

L'OR proporcional el model de dades, per al PoC utilitzarem un objecte anomenat «asset», que pot ser de diferents tipus, però al PoC s'utilitzarà el tipus «Thing Asset»

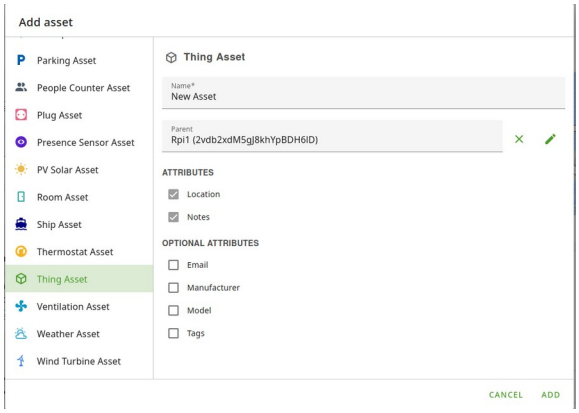


Figura 16: OpenRemote add Asset properties

Com espot veure, aquest tipus d'asset sols incorpora dos atributs obligatoris, Notes i Location. Per al PoC no interessa cap d'aquests atributs, però si crear-ne de nous, cosa que l'OR ens permet, ja que el model és dinàmic:

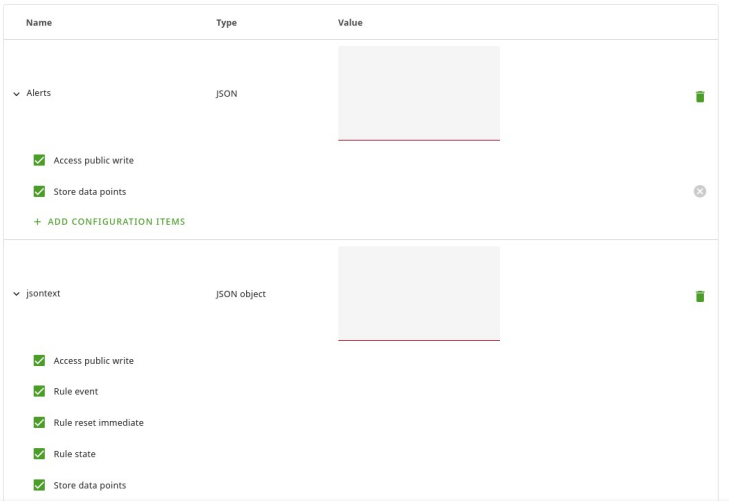


Figura 17: OpenRemote add Attributes to Asset

S'afegeixen dos atributs, jsontext i Alerts. Al primer (jsontext) s'emmagatzemaran les dades en «cru» tal com es rebran dels AG, el segon (Alerts) servirà per guardar les alertes associades a cada dispositiu. S'afegeixen les propietats de «Access public write» i «Store data points», que permeten que escriure l'atribut i que mantingui històric de tots els valors respectivament.

L'OR permet llegir i escriure els atributs dels asset utilitzant missatges MQTT, així doncs es creen topic específics per cada asset amb el següent format:

Lectura: {realm}/{clientId}/attribute/{attributeName}/{assetId}
Escriptura: {realm}/{clientId}/writeattributevalue/{attributeName}/{assetId}

Per al PoC, el realm serà «master» i el clientID un valor arbitrari que serà diferent per cada component software que es connecti al broker MQTT.

Es crea un usuari de servei dins de l'OR que permetrà la subscripció (rebre events) i l'enviament de dades als diferents topics:

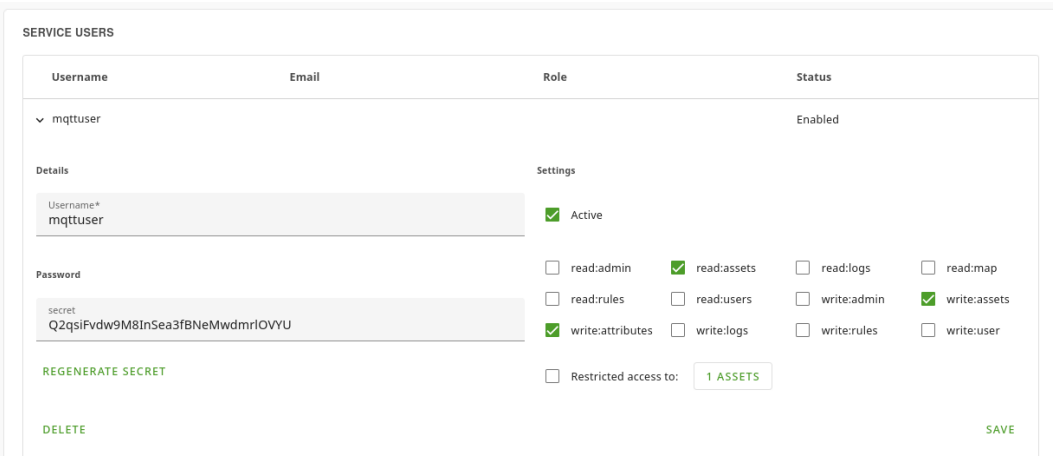


Figura 18: OpenRemote create service user

A la figura anerior s'atorguen els permisos per escriure als atributs, i llegir i escriure a l'asset. Per al PoC tots els components (AG, AF, DA) utilitzaran les mateixes credencials amb els mateixos permisos, en una implementació real, cada component tindria un usuari dedicat i sols els permisos que li calguessin.

5.3 Implementació del PoC

Per a simular el dispositiu IoT (AG) s'utilitza una Raspberry pi 3B, amb raspbian linux. S'instal·len els següents components addicionals: pip3, git i mosquitto-clients.

Es clona el repositori python de l'AWS IoT Device Defender Agent i es modifica el fitxer collertor.py per obtenir la IP en comptes del nom de la interface d'origen en les connexions obertes.

Per arrancar la recol·lecció de dades i enviament als DA, sols cal executar la següent comanda a la consola:

```
watch -n 60 'mosquitto_pub -d -q 1 -h "192.168.1.26" -p "8883" --cafile ca.cert --insecure -t "master/Rpi1AG1/writeattributevalue/jsontext/2vdb2xdM5gJ8khYpBDH6lD" -P "Q2qsiFvdw9M8InSea3fBNeMwdmrlOVYU" -u "master:mqttuser" --id "Rpi1AG1" -m "$(python3 collector.py -n 1 -s 1 -cm)"'
```

Aquesta comanda executa cada 60 segons el collector.py d' AWS i utilitza el mosquitto_pub per enviar les dades al broker MQTT. Es pot veure a la consola web de l'OR com s'actualitzen les dades cada minut:

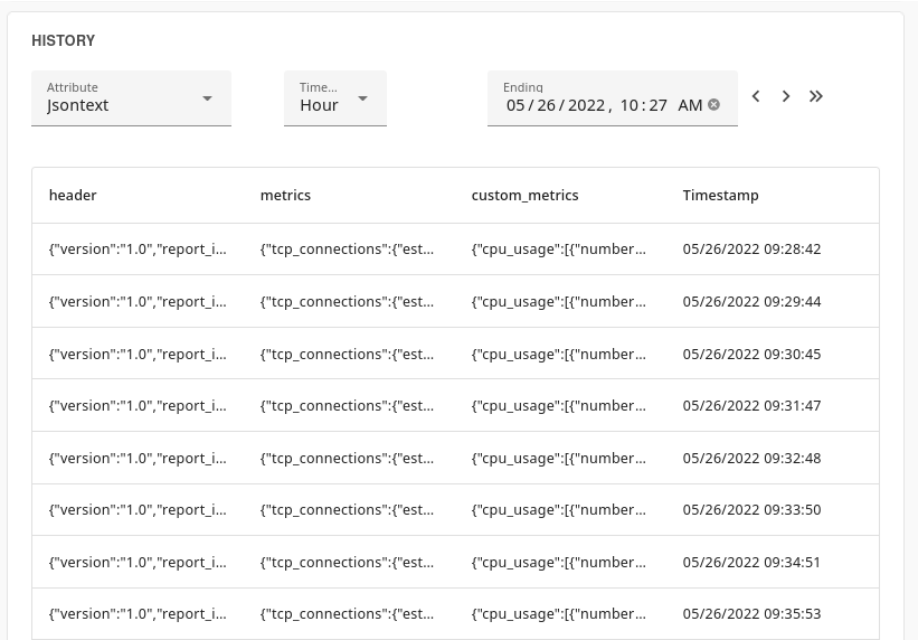


Figura 19: OpenRemote show attribute value history

Un exemple de les dades enviades en format JSON es pot trobar a l'Annex1.

Per desenvolupar el DA s'ha creat un programa molt simple en python3 que escolta tots els events que genera l'OR en modificar un atribut de qualsevol asset, per fer això cal llegir del topic: master/DA1/attribute/jsontext/# . Cada cop que es rep un event s'executa una funció on_message que extrau les dades del missatge rebut i les passa a la funció rule1 (que simula una regla de detecció d'anomalies). Al PoC en aquesta funció s'analitzen les connexions sortints del dispositiu IoT, i si es detecta una connexió al port tcp/22, es considera una anomalia i actualitza l'atribut Alerts de l'asset (Escriu al topic d'Alerts). El codi del DA es pot trobar a l'Annex2 i un exemple d'Alert en format JSON a l'Annex3. S'ha deixat la sortida de les traces per pantalla per poder fer millor seguiment.

Per desenvolupar l'AF s'ha creat un programa molt simple en python3 que escolta tots els events que genera l'OR en modificar un atribut de qualsevol asset, per fer això cal llegir del topic: master/AF1/attribute/Alerts/# . Cada cop que es rep un event s'executa una funció on_message que extrau les dades del missatge rebut i compara la IP destí de l'anomalia amb un llistat de IPs assignades a interfaces locals que s'ha llegit en arrancar el programa. Si la IP destí de l'atac coincideix amb una de les IP del dispositiu, es genera una regla de firewall per tallar el tràfic amb origen el dispositiu que ha generat l'anomalia. El codi de l'AF es pot trobar a l'Annex4 i la sortida que genera a l'Annex5.

6. Conclusions i Treball futur

6.1 Conclusions

La primera conclusió a què s'ha arribat en aquest treball és que existeixen molt sistemes operatius per a dispositius IoT i que és gairebé impossible determinar el marketshare de cada un d'ells. El que si sembla clar és que els dispositius IoT es poden classificar en dos tipus segons el hardware del qual disposen, sistemes amb microcontroladors i sistemes microprocessadors.

S'ha pogut determinar que gran part dels dispositius IoT amb microprocesador utilitzen kernel Linux com a sistema operatiu, i la gran majoria d'atacs a gran escala contra aquest tipus de dispositius estan dissenyats per format xarxes de botnets.

Existeixen 4 grans mètodes de detecció d'intrusos i dins d'aquest els mètodes de detecció d'anomalies són els més adients per detectar tant atacs coneguts com desconeguts. En tot cas, amb el disseny presentat en aquest treball es podrien utilitzar diversos mètodes de detecció, reforçant-los amb tècniques de ML.

S'ha pogut constatar l'existència de força implementacions HIDS en opensource, però tot i que algunes estan pensades per funcionar en entorns distribuïts de grans dimensions, no són adequats per gestionar milers o milions de dispositius IoT i en molts casos els mètodes de recol·lecció de dades són massa exigents per dispositius amb pocs recursos.

S'ha fet una proposta d'un sistema HIDS on la característica principal és que se separa la capa de recol·lecció de dades de la capa d'anàlisis, d'aquesta manera la recollida de dades als dispositius IoT necessita molt pocs recursos, i l'anàlisi es pot fer en altres dispositius que disposin dels recursos necessaris.

Existeixen diverses propostes d'implementacions d'HIDS per a sistemes IoT, però no ha estat possible trobar cap implementació. El que ha estat possible és trobar llibreries i protocols optimitzats per dispositius amb pocs recursos que han permès proposar una arquitectura de software i desenvolupar un PoC per demostrar la viabilitat de la proposta.

L'arquitectura proposada té com a pilar el desacoblament dels diferents components, cosa que permet simplificar el desenvolupament de cada un d'ells i poder reemplaçar-los de forma independent. També permet desplegar tots els components en una configuració multinivell (dispositiu, edge, fog, cloud) , on a cada nivell es poden analitzar les dades agregades dels nivells inferiors, i també prendre accions (filtra tràfic) per protegir els dispositius.

Amb el desenvolupament del PoC es constata que l'arquitectura és viable des del punt de vista tècnic, i que reutilitzant components opensource és possible implementar la funcionalitat mínima amb molt poc codi. Amb les proves realitzades al PoC s'ha pogut comprovar que la recol·lecció de dades i enviament de les mateixes per part del dispositiu IoT consumeix molt pocs

recursos, i per tant seria viable fins i tot utilitzar-la en dispositius amb microcontroladors i sistema operatiu simple.

Amb la implementació realment simple de l'AF s'ha pogut constatar que seria possible evitar la propagació per la xarxa de diferents tipus d'atacs simplement tallant el tràfic quan es detecten.

6.2 Treball Futur

Cal desenvolupar el concepte de perfil: com l'interpreta cada component, com es propaga pels diferents nivells (cloud, edge, fog, etc.), com es fusionen perfils que provenen de capes diferents, quines dades comunes i quines especifiques cal que tinguin, etc.

No s'ha proposat cap metodologia per què els DA puguin ser localitzats per la resta de components i per DA d'altres nivells. Existeixen força tesi per muntar jerarquies de dispositius de forma automàtica, que podrien ser adaptades per a proposta que s'ha fet. També existeixen implementacions ja construïdes de sistemes per la localització de components software.

En aquest treball no s'ha entrat en el detall en els diferents mètodes de detecció d'anomalies, existeixen desenes de papers amb propostes teòriques de nous mètodes de detecció, alguns d'ells específics per a protocols molt lligats al IoT com ara: BLE, Zigbee, Z-Wave, etc.

Al PoC s'ha observat que els AG consumeixen molt pocs recursos, el següent pas seria implementar-los en una arquitectura amb microcontroladors i sistema operatiu simple.

Encara que en l'arquitectura s'ha proposat utilitzar un motor de regles per analitzar la informació rebuda dels AG, no s'ha fet la implementació pràctica al PoC per raons de temps. Per acabar de validar que l'arquitectura és correcte, caldria implementar els perfils i usant un motor de regles, analitzar les dades rebudes. Així es verificaria el flux proposat en l'arquitectura.

Pel que fa a les regles, sols s'ha mostrat el funcionament d'una regla molt simple: en detectar una connexió ssh sortint, s'aixeca una alarma. Es podria fer un estudi de les regles específiques del món IoT i veure la viabilitat d'usar-les en un motor de regles, aquest estudi també permetria veure quines dades cal recol·lectar per millorar les ràtios de detecció.

Caldria fer una proposta més completa del funcionament dels AF, ja que cal tenir en compte molts factors abans de restringir el tràfic de xarxa, com per exemple: NAT entre dispositius, atacs de suplantació, etc. També es podrien fer propostes per bloquejar tràfic no IP, i fins i tot proposar altres mètodes per aturar els atacs com parar processos, reinicia el dispositiu, etc.

A la implementació de PoC s'ha utilitzat OpenRemote com a component central que inclou diversos dels serveis necessaris. Caldria estudiar a fons si es podria

basar un producte real en aquest software, ja que incorpora molts altres components que en principi no són d'utilitat per l'arquitectura proposada.

Tot i que s'ha utilitzat la interface WEB GUI de l'OpenRemote, no és la més adequada per mostrar les dades que habitualment mostren els HIDS, així doncs caldria estudiar l'ús d'altres eines per la implementació dels MO.

7. Glossari

Moltes d'aquestes definicions són extretes de la Viquipèdia i Wikipedia.

PoC (Proof of concept): Una prova de concepte és una implementació, sovint resumida o incompleta, d'un mètode o d'una idea, realitzada amb el propòsit de verificar que el concepte o la teoria en qüestió és susceptible de ser explotada d'una manera útil.

IoT (Internet of Things): La Internet de les coses es refereix, en termes d'informàtica, a una xarxa d'objectes físics de la vida quotidiana interconnectats mitjançant sensors, programari i altres tecnologies per tal d'intercanviar dades amb altres dispositius i sistemes a través d'Internet.

DdoS (distributed denial-of-service attack): Un atac de denegació de servei, també conegut com a Atac DoS o atac DDoS és una incursió contra la seguretat informàtica. Consisteix a assaltar el servei d'un dispositiu a través d'un conjunt de moltes màquines i mitjançant trames IP amb bandera falsa per tal que el dispositiu augmenti el seu temps de processador. D'aquesta manera, s'aconsegueix que deixi d'oferir el servei en qüestió.

HIDS (Host Intrusion Detection System): Sistema de detecció d'intrusos en un Host. Cerca detectar anomalies que indiquen un risc potencial, revisant les activitats en el dispositiu (host). Pot prendre mesures protectores. En el context del IoT considerem Host qualsevol dispositiu amb capacitat de comunicar-se a internet.

IPS (Intrusion Prevention System): és un programari que exerceix el control d'accés en una xarxa informàtica per protegir els sistemes computacionals d'atacs i abusos. La tecnologia de prevenció d'intrusos és considerada per alguns com una extensió dels sistemes de detecció d'intrusos (IDS), però en realitat és un altre tipus de control d'accés, més proper a les tecnologies tallafocs.

TFM: Treball final de màster.

Malware: El programari maliciós o programari nociu és el programari o arxiu nociu per a dispositius que està dissenyat per a inserir virus, cucs, troians, programari espia o fins i tot bots, intentant aconseguir algun objectiu, com ara recollir informació sobre l'usuari o sobre l'ordinador en si. La paraula en anglès malware prové d'una agrupació de les paraules en anglès (malicious software).

MMU: Una unitat de gestió de memòria també coneguda per les inicials angleses MMU (Memory Management Unit) és un component de maquinari encarregat de gestionar l'accés a la memòria des de la CPU.

Botnet: és un grup de dispositius (anomenats bots o zombies) connectats a Internet que involuntàriament, un cop han estat infectats amb un virus, un cuc o un troià, poden ser controlats remotament per executar tasques sense l'autorització del propietari i sense que aquest se n'adoni. Les botnets poden arribar a tenir milers o centenars de milers de dispositius sota control i es fan servir per a desfermar, per exemple, atacs massius de denegació de servei, enviar onades de correu brossa (spam) o infectar a altres dispositius poc protegits.

Edge Computing: és un paradigma de computació distribuïda que apropa la computació i l'emmagatzematge de dades a les fonts de dades. S'espera que això millori els temps de resposta i estalviï amplada de banda. Una idea errònia comuna és que Edge i IoT són sinònims. Edge Computing és una forma de computació distribuïda sensible a la topologia i la ubicació, mentre que IoT és un de cas d'ús de l'Edge Computing.

Logs: En informàtica, s'usa el terme registre, log o historial de log per referir-se a l'enregistrament seqüencial en un fitxer o en una base de dades de tots els esdeveniments (esdeveniments o accions) que afecten un procés particular (aplicació, activitat d'una xarxa informàtica, etc.).

LibC: La C standard library o llibreria estàndard de C és un conjunt de seccions de l'estàndard ISO C que descriu una col·lecció de fitxers de capçalera (.h) i rutines de llibreria usades per a implementar operacions comunes en el llenguatge de programació Llenguatge C, com l'entrada/sortida i la gestió de cadenes de text.

Busybox: és una utilitat de programari lliure que combina moltes eines estàndard d'Unix i comandes de Linux en un sol arxiu, estalviant molt d'espai i fent-la gairebé imprescindible en el desenvolupament de sistemes incrustats. Molt sovint se'n fa referència com a la "navalla suïssa" dels sistemes Linux.

C2: La infraestructura de comandament i control, també coneguda com a C2 o C&C, és el conjunt d'eines i tècniques que utilitzen els atacants per mantenir la comunicació amb dispositius compromesos després de l'explotació inicial. Els mecanismes específics varien molt entre els atacs, però C2 generalment consisteix en un o més canals de comunicació encoberts entre dispositius víctimes i una plataforma que controla l'atacant. Aquests canals de comunicació s'utilitzen per enviar instruccions als dispositius compromesos, descarregar càrregues útils malicioses addicionals i enviar dades robades a l'adversari.

PCAP: El pcap és una API per a captura de paquets. La implementació del pcap per a sistemes basats en Unix es coneix com a libpcap; el port per a Windows del libpcap rep el nom de WinPcap. El libpcap i WinPcap poden ser utilitzats per un programa per capturar els paquets que viatgen per tota la xarxa.

API: Una interfície de programació d'aplicacions (en anglès Application Programming Interface, API) és una interfície que especifica com diferents components de programes informàtics haurien d'interaccionar. Dit d'una altra manera, és un conjunt d'indicacions, quant a funcions i procediments, ofert per una biblioteca informàtica per ser utilitzat per un altre programa per interaccionar amb el programa en qüestió.

Dashboard: Un panell de control és una eina de gestió de la informació que monitoritza, analitza i mostra de manera visual els indicadors clau de desenvolupament (KPI), mètriques i dades fonamentals per fer un seguiment de l'estat d'un sistema, empresa, un departament, una campanya o un procés específic.

Podem pensar en el dashboard com una mena de «resum» que recopila dades de diferents fonts en un sol lloc i les presenta de manera digerible perquè el més important salti a la vista.

8. Bibliografia

Bibliografía

[1]:Kevin Y Huang,2020,https://www.trendmicro.com/en_us/research/20/h/probing-attempts-on-home-routers-increase-in-1H-2020.html, Data visita:20/2/2022
[2]:Ryan Daws,2020,https://iottechnews.com/news/2020/sep/18/ibm-xforce-mozi-botnet-iot-traffic/, Data visita:20/2/2022
[3]:wikipedia,2022,https://en.wikipedia.org/wiki/DDoS_attack_on_Dyn, Data visita:20/2/2022
[4]:Radware,2021,https://www.radware.com/security/threat-advisories-and-attack-reports/dark-iot-botnet/, Data visita:20/2/2022
[5]:AWS,2022,https://docs.aws.amazon.com/freertos/latest/userguide/afr-device-defender-library.html, Data visita:20/2/2022
[6]:Kevin Ashton,2009,https://www.rfidjournal.com/that-internet-of-things-thing, Data visita:20/4/2022
[7]:Help Net Security,2022,https://www.helpnetsecurity.com/2020/07/24/analysis-of-5-million-unmanaged-iot-and-iomt-devices/, Data visita:25/2/2022
[8]:Tripwire,2021,https://www.tripwire.com/misc/iot-and-iiot-cybersecurity-report, Data visita:25/2/2022
[9]:Sjoerd Langkemper,2021,https://www.eurofins-cybersecurity.com/news/security-problems-iot-devices/, Data visita:24/02/2022
[10]:John Paul Farmer,2021,https://www1.nyc.gov/assets/cto/#/project/iot-strategy, Data visita:23/2/2022
[11]:Christina Skouloudi, Apostolos Malatras, Rossen Naydenov, Georgia Dede,2020,https://www.enisa.europa.eu/publications/guidelines-for-securing-the-internet-of-things, Data visita:23/2/2022
[12]:ENISA,2017,https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot, Data visita:24/02/2022
[13]:Wikipedia,2021,https://en.wikipedia.org/wiki/Host-based_intrusion_detection_system_comparison, Data visita:26/02/2022
[14]:Eclipse Fundation,2021,https://outreach.eclipse.foundation/iot-edge-developer-2021, Data visita:10/3/2022
[15]:Robet Hackett,2016,http://fortune.com/2016/10/03/botnet-code-ddos-hacker/, Data visita:14/3/2022
[16]:Lily Hay Newman,2016,https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/, Data visita:14/3/2022
[17]:Dorka Palotay,2021,https://cujo.com/iot-botnet-report-2021-malware-and-vulnerabilities-targeted/, Data visita:14/3/2022
[18]:Lindsey O'Donnell,2021,https://threatpost.com/d-link-iot-tor-gafgyt-variant/164529/, Data visita:16/3/2022
[19]: Charlie Osborne,2021,https://www.zdnet.com/article/this-is-why-the-mozi-botnet-will-linger-on/, Data visita:17/3/2022

[20]:Callum Cyrus,November 2021,https://www.iotworldtoday.com/2021/11/16/botenago-malware-targets-millions-of-iot-devices/, Data visita:2/5/2022
[21]:John Leyden,2022,https://portswigger.net/daily-swig/zero-day-vulnerabilities-in-nooie-baby-monitors-could-allow-video-feed-hijack, Data visita:16/3/2022
[22]:Bitdefender,2022,https://www.bitdefender.com/blog/labs/vulnerabilities-identified-in-nooie-baby-monitor/, Data visita:16/3/2022
[23]:Mitre,2020,https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-15744, Data visita:16/3/2022
[24]:Ionut Arghire ,2020,https://www.securityweek.com/vulnerability-linear-emerge-access-controllers-exploited-wild, Data visita:16/3/2022
[25]:Gjoko Krstic,2019,https://applied-risk.com/resources/ar-2019-005, Data visita:16/3/2022
[26]:Eduard Kovacs,2019,https://www.securityweek.com/over-100-flaws-expose-buildings-hacker-attacks, Data visita:16/3/2022
[27]:Edmund Brumaghin,2018,https://blog.talosintelligence.com/2018/07/samsung-smartthings-vulns.html?m=1, Data visita:16/3/2022
[28]:Jean-Luc Aufranc,2022,https://www.cnx-software.com/2022/03/19/gas-pumps-insecure-typical-router/, Data visita:20/3/2022
[29] :Pietro Spadaccino and Francesca Cuomo, Intrusion Detection Systems for IoT: opportunities, 2020
[30] :Ansam Khraisat*, Iqbal Gondal, Peter Vamplew and Joarder Kamruzzaman, Survey of intrusion detection systems:, 2019
[31]:Apoorv Dixit,2022,https://www.codingninjas.com/codestudio/library/ids-detection-methods, Data visita:13/3/2022
[32]:Garrett Gross,2022,https://cybersecurity.att.com/blogs/security-essentials/intrusion-detection-techniques-methods-best-practices, Data visita:13/3/2022
[33]:Jason Hoffman,2022,https://wisdomplexus.com/blogs/different-types-of-intrusion-detection-systems-ids/#Methods, Data visita:13/3/2022
[34] :N. U. Sheikh, H. Rahman, S. Vikram, and H. AlQahtani,, A lightweight signature-based ids for iot environment, 2018
[35] :Bharat Atul Desai, Dinil Mon Divakaran, et al, A feature-ranking framework for IoT device, 2018
[36] :Muhammad Aminu Lawal, Riaz Ahmed Shaikh, Syed Raheel, A DDoS Attack Mitigation Framework for IoT Networks using Fog Computing, 2021
[37]:Shahid Raza, Linus Wallgren, Thiemo Voigt,2013,https://www.researchgate.net/publication/256935547_SVELTE_Real-time_intrusion_detection_in_the_Internet_of_Things, Data visita:20/04/2022
[38]:Adrien Cosson, Amit Kumar Sikder, Leonardo Babun, Z. Berkay Celik, Patrick McDaniel , A. Selcuk Uluagac,2021,https://www.researchgate.net/publication/351679100_Sentinel_A_Robust_Intrusion_Detection_System_for_IoT_Networks_Using_Kernel-Level_System_Information, Data visita:20/04/22
[39]:Freertos,2022,https://www.freertos.org/, Data visita:9/3/2022
[40]:Microsoft,2022,https://developer.microsoft.com/en-us/windows/iot/, Data visita:9/3/2022
[41]:Zephyr® Project, a Linux Foundation Project,2022,https://www.zephyrproject.org/, Data visita:10/3/2022

[42]:Wikipedia,2019,https://es.wikipedia.org/wiki/GNU_toolchain, Data visita:12/2/2022
[43]:Wikipedia,2022,https://en.wikipedia.org/wiki/MClinux, Data visita:12/2/2022
[44]:Erik Andersen,2008,https://busybox.net/, Data visita:12/3/2022

Annex1


```
{
  "header": {
    "version": "1.0",
    "report_id": 1653551661
  },
  "metrics": {
    "tcp_connections": {
      "established_connections": {
        "total": 1,
        "connections": [
          {
            "local_port": 22,
            "remote_addr": "192.168.1.26:35592",
            "local_interface": "192.168.1.34"
          }
        ]
      }
    },
    "listening_tcp_ports": {
      "ports": [
        {
          "port": 22,
          "interface": "0.0.0.0"
        },
        {
          "port": 22,
          "interface": "::"
        }
      ],
      "total": 2
    },
    "listening_udp_ports": {
      "ports": [
        {
          "port": 68,
          "interface": "0.0.0.0"
        },
        {
          "port": 56978,
          "interface": "::"
        },
        {
          "port": 5353,
          "interface": "0.0.0.0"
        },
        {
          "port": 5353,
          "interface": "::"
        },
        {
          "port": 41357,
          "interface": "0.0.0.0"
        }
      ],
      "total": 5
    }
  },
  "custom_metrics": {
    "cpu_usage": [
      {
        "number": 25.8
      }
    ]
  }
}
```

Annex2

```
import time
import paho.mqtt.client as mqtt
import json
import psutil as ps

broker_address = "localhost"
broker_port = 8883
topic = "master/DA1/attribute/jsontext/#"
client = ""

def rule1(tcp_conns, asset):
    for i in tcp_conns['connections']:
      ra = i['remote_addr'].split(':')
      if ra[1] == '22' :
          print("!!!!!! Outgoing SSH detected on device: ",asset," !!!!!!")
          alarms="master/DA1/writeattributevalue/Alerts/"+asset[0]
            alm = { 'source_assetID': asset[0], 'description': 'Outgoing SSH detected',
'connection_info': i }
          client.publish(alarms, payload=json.dumps(alm), qos=0, retain=False)


def on_log(client, userdata, level, buf):
    print("log: ",buf)

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

def on_message(client, userdata, message):
    json1 = json.loads(str(message.payload.decode("utf-8")))
    assetID = json1['path']
    values = json1['attributeState']['value']
    cpu_usage = values['custom_metrics']['cpu_usage']
              print("################################    asset    ID:    ",assetID,    "
################################")
    print("CPU Usage: ", cpu_usage)
    listening_tcp_ports = values['metrics']['listening_tcp_ports']
    print("Listening TCP ports: ", listening_tcp_ports)
    listening_udp_ports = values['metrics']['listening_udp_ports']
    print("Listening UDP ports: ", listening_udp_ports)
                tcp_established_connections    =    values['metrics']['tcp_connections']
['established_connections']
    print("TCP established connections: ", tcp_established_connections)
    rule1(tcp_established_connections, assetID)

client = mqtt.Client("DA1")
client.on_message = on_message
client.on_connect = on_connect
client.on_log=on_log
client.username_pw_set("master:mqttuser", "Q2qsiFvdw9M8InSea3fBNeMwdmrlOVYU")
client.tls_set("./ca.cert")
client.tls_insecure_set(True)
client.connect(broker_address, broker_port, 60)
client.subscribe(topic)
client.loop_forever()
```

Annex3

```
{
  "description": "Outgoing SSH detected",
  "source_assetID": "2vdb2xdM5gJ8khYpBDH6lD",
  "connection_info": {
    "local_port": 44502,
    "remote_addr": "192.168.1.26:22",
    "local_interface": "192.168.1.34"
  }
}
```

Annex4

```
import time
import paho.mqtt.client as mqtt
import json
import psutil as ps

broker_address = "localhost"
broker_port = 8883
topic = "master/FA1/attribute/Alerts/#"
lIPs = []
client = ""

def localIPs():
    for nic, addrs in ps.net_if_addrs().items():
        if nic == 'lo': continue;
        for addr in addrs:
            if str(addr.family) == 'AddressFamily.AF_INET':
              lIPs.append(addr.address)

def on_log(client, userdata, level, buf):
    print("log: ",buf)

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

def on_message(client, userdata, message):
    json1 = json.loads(str(message.payload.decode("utf-8")))
    vals = json1['attributeState']['value']
    print(json.dumps(vals))
    ra = vals['connection_info']['remote_addr'].split(':')
    la = vals['connection_info']['local_interface']
    if ra[0] in lIPs:
       print("!!!!! BAD CONNECTION !!!!")
       print("iptables -A INPUT -p tcp -s",la,"--dport 22 -j DROP")

localIPs()
client = mqtt.Client("FA1")
client.on_message = on_message
client.on_connect = on_connect
#client.on_log=on_log
client.username_pw_set("master:mqttuser", "Q2qsiFvdw9M8InSea3fBNeMwdmrlOVYU")
client.tls_set("./ca.cert")
client.tls_insecure_set(True)
client.connect(broker_address, broker_port, 60)
client.subscribe(topic)
client.loop_forever()
```

Annex5

```
{"source_assetID": "2vdb2xdM5gJ8khYpBDH6lD", "description": "Outgoing SSH detected",
"connection_info": {"local_interface": "192.168.1.34", "local_port": 40400,
"remote_addr": "192.168.1.26:22"}}

!!!!! BAD CONNECTION !!!!
iptables -A INPUT -p tcp -s 192.168.1.34 --dport 22 -j DROP
```

Annex6

Es fa una petita anàlisi dels següents sistemes operatius:

- Linux: Azure Sphere, Ubuntu core, Raspbian, etc.
- FreeRTOS [39]
- Windows 10 IoT [40]
- Zephyr [41]

Linux és un kernel que normalment s'envolta de les eines GNU[42] per tal de conformar el que s'anomena distribucions de linux, que són un sistema operatiu complet. És un kernel molt versàtil, que tant es pot usar en grans computadors, clusters, telèfons (android), així com en sistemes incrustats amb relativament pocs recursos. Tot i que existia una versió (uclinux[43]) que podia funcionar sense MMU (bàsicament en microcontroladors), la versió que utilitzen el 99,9% dels dispositius requereix una CPU de 32 bits amb suport de l'esmentada MMU. Així doncs, un sistema IoT amb Linux, en realitat és una distribució preparada i configurada per a dispositius amb pocs recursos. Aquestes distribucions inclouran tant el kernel de linux com la llibreria libc, altres llibreries (openssl, ld-linux, etc.), i en la majora de casos també inclouen les eines de sistema, sigui individualment o com a grup utilitzant Busybox[44]. En molts casos requereix l'ús de sistema de fitxers, que normalment es mantenen sobre dispositius no volàtils com són memòries Flash (NOR/NAND/SLC/MLC), microSD card, etc. El kernel del linux inclou tots els controladors dels perifèrics (wifi, targetes de so, GPIO, etc.) i les eines de sistema proporcionen mecanismes (dhcp, fitxers, etc.) per configurar-los mitjançant scripts en el procés d'arrancada del dispositiu. Els protocols de xarxa com TCP/IP, WPA, PPP, etc. són implementats al kernel de linux també. El programes d'usuari poden executar amb usuaris no privilegiats i utilitzar les llibreries que proporciona el sistema operatiu per fer les seves operacions, com per exemple xifra les connexions utilitzant openssl. En tenir sistema de fitxers, es poden actualitzar els diferents components de forma independent. S'utilitza àmpliament en càmeres de videovigilància, assistents i robots domèstics, i en general dispositius que requereixen força processament o amplada de banda. Cal esmentar que Android, que és molt usat en dispositius multimèdia domèstics (TV, VDR, etc.), utilitza el kernel de linux, el que augmenta encara mes l'adopció d'aquest kernel en el mercat de dispositius IoT.

FreeRTOS és un kernel pensat per sistemes en temps real (RTOS vol dir Real Time Operating System), i microcontroladors que utilitza molt pocs recursos (processador de 10Mhz i 16K de RAM), opcional-ment es poden utilitzar llibreries que faciliten el desenvolupament. FreeRTOS normalment no utilitza sistema de fitxers, es distribueix el kernel, les llibreries i l'aplicatiu en un sol binari que és executat pel microcontrolador directament des de la ROM, fent impossible actualitzar sols una part del sistema. Com que no té sistema de fitxers ni carrega dinàmica de llibreries, totes les funcions que hagi d'usar l'aplicatiu s'han d'incloure en el kernel, això inclou llibreries per la gestió del SSL, per fer crides HTTP o per gestionar serveis MQTT, entre d'altres. Cal esmentar que els protocols de xarxa també s'implementen com a llibreries, i en

alguns casos és possible trobar diverses implementacions del mateix protocol. Tots els controladors dels perifèrics també estan implementats dins del kernel i existeix una gestió de privilegis molt simple, ja que la majoria de microcontroladors no tenen MMU i no poden protegir zones de memòria. Es força utilitzats en dispositius de baix consum i que no requereixen molta potència de còmput, com bombetes, alarmes, termòstats, electrodomèstics, etc.

Windows 10 IoT core: és una versió de Windows 10 que està optimitzada per a dispositius més petits amb pantalla o sense, que s'executa tant en dispositius ARM com x86/x64. Igual que el Linux, per funcionar requereix un processador de 32bits com a mínim amb suport de MMU. Aquesta versió de windows tot i estar retallada, continua tenim moltes de les funcions de la versió de windows 10 estàndard d'escriptori. Així doncs te kernel, llibreries del sistema, drivers, llibreries auxiliars i sistema de fitxers, tot i que es poden instal·lar diverses aplicacions, sols una es pot arrancar com aplicació per defecte. Les aplicacions en segon pla no tenen interfície d'usuari i s'executen contínuament. Es llancen a l'inici en l'arrancada del dispositiu i ho continuaran fent indefinidament, es tornaran a obrir si es bloquegen. La gestió remota dels dispositius amb aquesta versió de windows es fa per powershell, sobretot dels que no disposen de pantalla.
Cal esmentar que implementa força mecanisme de seguretat de forma predeterminada cosa que en els sistemes amb linux és opcional. En tenir sistema de fitxers es poden fer actualitzacions de parts del sistema operatiu, de fet el windows update és del tot funcional. Windows IoT core es pot trobar sobretot en equips mèdics i industrials.

Zephyr: és un sistema operatiu de temps real per a dispositius amb pocs recursos, sobretot microcontroladors. Cal esmentar que és un sistema molt flexible que permet treballar amb diferents sistemes de gestió de memòria, planificadors de recursos i fins i tot amb sistema de fitxers o sense. Així i tot no deixa de ser un sistema operatiu molt semblant a FreeRTOS, amb un sol binari que inclou el kernel, llibreries i l'aplicació. La distribució base inclou tot un set de protocols de xarxa com ara IPv4 and IPv6, Constrained Application Protocol (CoAP), LwM2M, MQTT, 802.15.4, Thread, Bluetooth Low Energy i CAN, sense tenir que recorre a llibreries externes.
Des del punt de vista de la seguretat, implementa protecció de desbordament de pila específica de cada arquitectura, seguiment de permisos d'objectes del nucli i controlador de dispositiu i aïllament de fils amb protecció de memòria a escala de fils en arquitectures x86, ARC i ARM, espai d'usuari i dominis de memòria.

Per a plataformes sense MMU/MPU i dispositius amb memòria limitada, admet la combinació de codi específic de l'aplicació amb un nucli personalitzat per crear una imatge monolítica que es carrega i s'executa al maquinari d'un sistema. Tant el codi de l'aplicació com el codi del nucli s'executen en un únic espai d'adreces compartides.