
This is the **published version** of the bachelor thesis:

Palacio Aguilera, Claudia. *Defensa activa de xarxes : disseny i implementació d'un sistema autònom de detecció i resposta davant incidències*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334446>

under the terms of the  license.

Defensa Activa de Xarxes: Disseny i Implementació d'un Sistema Autònom de Detecció i Resposta davant Incidències.

Clàudia Palacio Aguilera

30 de juny de 2026

Resum– Escurçar el temps d'exposició durant un incident de seguretat és essencial per a la continuïtat digital. Aquest treball detalla el desplegament d'un entorn virtualitzat amb un sistema de defensa activa basat en la correlació d'esdeveniments. L'arquitectura integra Suricata [1][2] per a la detecció d'amenaçes i Prometheus [3] per al monitoratge del rendiment en temps real. Un orquestrador en Python centralitza les alertes de xarxa i les mètriques del sistema per verificar la legitimitat del trànsit, mostrant tota l'activitat en un panell de control amb Grafana jerarquitzat per nivells de risc. El sistema s'ha validat mitjançant escenaris d'escaneig de ports, força bruta i atacs DoS [4]. En tots els casos, l'orquestrador ha actuat de forma autònoma, aplicant regles *iptables* i reiniciant serveis sense cap intervenció manual. En delegar aquestes tasques a l'orquestrador, s'han reduït significativament els temps de resposta i gestionat les anomalies amb major eficiència que amb una configuració tradicional.

Paraules clau– Automatització, correlació d'esdeveniments, incidents de seguretat, monitoratge en temps real, orquestració, seguretat de xarxes, virtualització.

Abstract– Cutting down exposure time during a security incident is essential for digital continuity. This paper details the setup of a virtualized environment with an active defense system driven by event correlation. The architecture integrates Suricata [1][2] for threat detection and Prometheus [3] for real-time performance monitoring. A Python-based orchestrator centralizes network alerts and system metrics to verify traffic legitimacy, displaying all activity on a Grafana dashboard ranked by risk levels. The system was validated using port scanning, brute-force, and DoS [4] scenarios. In every case, the orchestrator acted autonomously, applying *iptables* rules and restarting services without manual intervention. By shifting these tasks to the orchestrator, response times were significantly reduced, and anomalies were handled with greater efficiency than with a traditional setup.

Keywords– Automation, event correlation, network security, orchestration, real-time monitoring, security incidents, virtualization.



1 INTRODUCCIÓ - CONTEXT DEL TREBALL

AVUI dia, la dependència de les organitzacions vers les seves infraestructures digitals és tan crítica que qualsevol incidència, per petita que sigui, bloqueja

ja l'activitat de forma immediata. Si un servei clau cau, els processos interns s'aturen i la capacitat de reacció és mínima. Un cas clar va ser l'atac de *ransomware* a la Universitat Autònoma de Barcelona (UAB) l'octubre de 2021 [5], on serveis bàsics com el correu electrònic o el campus virtual van quedar inoperatius durant dies, exigint setmanes de recuperació.

A aquesta vulnerabilitat cal sumar-hi el soroll constant de les xarxes corporatives: escanejos de ports, intents de força bruta i atacs de denegació de servei són part del dia a dia. A més, existeixen anomalies no malicioses, com pics de CPU o fallades en nodes VPN, que cal saber distingir d'un atac

• E-mail de contacte: claudia.palacioa@autonoma.cat
• Menció: Enginyeria de Computadors
• Treball tutoritzat per: Eduardo César Cabrera Flores (Departament d'Arquitectura de Computadors i Sistemes Operatius)
• Curs 2025/26

real per evitar falsos positius en la presa de decisions.

Davant d'això, les mesures preventives tradicionals ja no serveixen. Calen sistemes que detectin què passa en temps real i actuïn abans que el problema s'estengui. La defensa activa apareix aquí com una estratègia robusta, combinant monitoratge constant i resposta automatitzada per tallar l'impacte dels incidents de seguretat.

Aquest treball neix d'aquesta premissa i proposa una arquitectura virtual per simular atacs de xarxa reals i validar la resposta automàtica del sistema. L'entorn desplegat unifica la detecció de xarxa amb la correlació de dades, executant contramesures de forma autònoma. Tota aquesta activitat se centralitza en un panell de control que visualitza el risc i l'estat dels serveis, permetent supervisar les mètriques de l'orquestrador en temps real. L'objectiu final és comprovar si aquest model realment escurça els temps de resposta i millora la defensa davant d'amenaces.

2 OBJECTIUS

L'eix d'aquest treball és el disseny i el desenvolupament d'una infraestructura de defensa activa que automatitzi la reacció davant d'incidents de seguretat. Els objectius principals, ordenats per prioritat d'execució, són:

- **Implementació de la Topologia:** Configurar una infraestructura virtualitzada i aïllada (WAN, DMZ) mitjançant VirtualBox per simular el comportament d'un entorn corporatiu.
- **Integració del Sistema IPS:** Incorporar Suricata com a eina d'anàlisi de trànsit per a la detecció i el report de patrons d'atac en temps real.
- **Recollida de Dades del Sistema:** Desplegar Prometheus i Node Exporter per extreure mètriques de rendiment (CPU, memòria, xarxa) i correlacionar-les directament amb les alertes de seguretat.
- **Programació de l'Orquestrador:** Desenvolupar un motor de decisió en Python capaç de processar les alertes de Suricata i executar, de forma autònoma, l'activació de regles en *iptables*.
- **Centralització i Visualització a Grafana:** Desplegar un panell de control a Grafana que integri les dades de Prometheus, incloent les alertes de Suricata processades per l'orquestrador, permetent visualitzar l'estat de la xarxa jerarquitzat per nivells de risc.
- **Validació del Sistema:** Verificar l'eficàcia del sistema davant vectors d'atac reals, incloent-hi paquets ICMP, l'escaneig de ports amb *nmap*, la força bruta mitjançant SSH i els atacs de denegació de servei (DoS) amb *hping3*.

En definitiva, es pretén demostrar la viabilitat d'aquest model de defensa híbrida, on l'automatització resol la mitigació immediata de l'incident i la interfície visual garanteix que el tècnic mantingui la supervisió en tot moment.

Com a resultats s'espera obtenir un sistema capaç de distingir entre trànsit legítim i maliciós correlacionant les alertes de Suricata amb les mètriques de Prometheus, aplicar contramesures de forma autònoma en menys de 10 segons

sense intervenció humana, i centralitzar tota l'activitat en un panell de Grafana organitzat per nivells de risc.

3 ESTAT DE L'ART

L'eix d'aquest treball se situa en la transició de la seguretat reactiva cap a una de proactiva. Fins ara, el model dominant es basava en els IDS (*Intrusion Detection System*), sistemes que es limiten a vigilar i avisar, tal com ja proposava Denning al seu model fundacional de detecció d'intrusions [6], deixant la resposta final en mans d'un humà que sovint no té prou marge de temps. Aquest enfocament resulta insuficient en entorns on els atacs es propaguen en qüestió de segons i on la intervenció manual no pot garantir una resposta adequada. El problema és que, davant d'amenaces com el *ransomware* o atacs DoS massius, el servidor pot caure en qüestió de segons. Per això, el projecte adopta la filosofia d'un IPS (*Intrusion Prevention System*), definit pel *National Institute of Standards and Technology* (NIST) com un sistema capaç no només de detectar sinó també d'actuar de forma activa sobre el trànsit maliciós [7]: aquí l'automatització és la prioritat per tallar l'atac en el mateix instant que es detecta.

Per executar aquesta estratègia, s'utilitza Suricata [1], que destaca sobre opcions més clàssiques, com Snort, pel seu ús del *multithreading* [2], que permet processar volums de dades molt alts sense que el rendiment del sistema se'n ressentí. Aquesta capacitat és especialment rellevant en entorns amb trànsit intens, on un motor de detecció de fil únic podria convertir-se en un coll d'ampolla i comprometre la detecció en temps real.

Pel que fa a l'evolució cap a sistemes intel·ligents, diversos treballs recents exploren l'ús de tècniques d'aprenentatge automàtic per millorar la detecció d'anomalies de xarxa més enllà dels enfocaments basats en signatures. Al-muhanna i Dardouri [8] revisen com els models híbrids de *deep learning* milloren la precisió de detecció en escenaris dinàmics, especialment davant d'atacs de dia zero que els IDS tradicionals no poden identificar perquè no disposen d'una signatura prèvia. Ahmad et al. [9] presenten una anàlisi exhaustiva dels mètodes de *machine learning* aplicats a IDS, destacant que els principals reptes continuen essent la taxa de falsos positius i la capacitat de generalització a entorns diferents dels de l'entrenament. Aquests reptes justifiquen l'ús de llindars configurables en el present treball, ja que permeten adaptar la resposta al comportament específic de cada entorn.

A més, l'auge de la intel·ligència artificial generativa està transformant el panorama de la ciberseguretat: els atacants l'utilitzen per automatitzar la generació de codi maliciós, mentre que els equips de defensa exploren el seu potencial per millorar la detecció d'amenaces i accelerar la resposta a incidents. Un exemple recent és *Claude Mythos* [10], model desenvolupat per Anthropic amb capacitats avançades en detecció de vulnerabilitats. És capaç d'identificar *zero-days* i generar *exploits* (codis que aprofiten vulnerabilitats per aconseguir accés no autoritzat a un sistema) de forma autònoma, fet que ha accelerat el debat sobre l'ús responsable de la IA en ciberseguretat. Aquest context posa de manifest la necessitat de sistemes de defensa activa capaços d'adaptar-se a un panorama d'amenaces en constant evolució.

En paral·lel a l'evolució dels motors de detecció, l'arquitectura de resposta també ha evolucionat des dels sistemes SIEM (*Security Information and Event Management*) cap a plataformes SOAR (*Security Orchestration, Automation and Response*). Mentre que un SIEM es limita a recollir, normalitzar i correlacionar esdeveniments per generar alertes [11], un SOAR incorpora procediments automatitzats que executen accions de resposta (bloqueig d'IPs, aïllament d'equips, reinici de serveis) sense intervenció humana [12]. Aquesta transició és precisament la que motiva el present treball a escala reduïda: l'orquestrador desenvolupat actua com un SOAR minimalista, ja que substitueix els mecanismes de resposta d'una plataforma comercial per una lògica de correlació pròpia entre alertes de Suricata (equivalent a la component IDS d'un SIEM) i mètriques de Prometheus (equivalent a la component de monitoratge d'infraestructura).

Cal remarcar que les solucions comercials SOAR ofereixen aquesta correlació de manera genèrica i configurable mitjançant interfícies gràfiques, però el seu cost de llicència pot superar els 250.000\$ anuals i requerir un desplegament complex [13], factors que poden limitar-ne l'adopció en organitzacions amb pressupostos reduïts. L'aportació d'aquest treball és demostrar que, per a un cas d'ús acotat (un servidor DMZ amb serveis crítics ben definits), és viable implementar la mateixa filosofia de correlació detecció-mètrica-acció amb eines de codi obert i un orquestrador a mida, amb un cost d'implementació molt inferior i una transparència total sobre la lògica de decisió.

Tanmateix, la capacitat de resposta d'un IPS s'optimitza quan disposa de context sobre l'estat del *hardware*. Per aquest motiu, mentre Suricata vigila el trànsit de xarxa, Prometheus [3] extreu mètriques en temps real com la càrrega de la CPU o la saturació de la memòria. Aquesta correlació de dades, identificada com un dels reptes centrals en sistemes de seguretat moderns [14], és la que permet a l'orquestrador en Python prendre decisions precises: la resposta automatitzada es modula segons si es presenta, o no, una degradació real dels recursos. D'aquesta manera, el sistema és capaç de distingir entre un atac real i un pic de càrrega legítim, reduint significativament la taxa de falsos positius.

Finalment, tota aquesta infraestructura de dades es canalitza cap a Grafana, que crea un panell de control unificat. Mentre l'automatització (mitjançant *iptables*) gestiona gran part de la mitigació, l'operador humà manté la supervisió total a través d'una interfície clara que unifica l'estat de la xarxa i la salut del sistema en una sola pantalla.

4 METODOLOGIES

Per a la gestió del projecte s'ha optat per la metodologia Scrum [15], ja que el desenvolupament d'un sistema de defensa activa requereix un enfocament iteratiu on cada component es pugui provar de forma independent abans de la seva integració global. L'ús de cicles curts facilita la redefinició de tasques sense afectar el lliurament final del sistema. Cada bloc temporal actua com una fita tancada orientada a obtenir un prototip funcional.

4.1 Planificació temporal

El projecte s'organitza en cinc *Sprints* de quinze dies de durada cadascun. Aquest plantejament s'ha dissenyat per validar el funcionament de la infraestructura de manera incremental, fet que permet detectar i corregir errors de configuració abans d'avançar cap als nous objectius de desenvolupament.

A continuació, es detalla el contingut de cada bloc temporal:

- ***Sprint 1 - Anàlisi de Requisits i Disseny de l'Entorn:*** Definició dels objectius i identificació dels vectors d'atac a mitigar. Es determinen els requisits de *hardware* i *software* pel que fa a la virtualització.
- ***Sprint 2 - Disseny i Virtualització:*** Construcció de la topologia a VirtualBox. Es prioritza l'aïllament dels segments (WAN i DMZ) abans d'instal·lar cap servei, assegurant que l'encaminament entre màquines funciona correctament.
- ***Sprint 3 - Monitoratge i Visualització:*** Implementació de Suricata, Prometheus i Grafana. En aquesta etapa, el sistema actua com a IDS, recollint dades i generant *logs*.
- ***Sprint 4 - Desenvolupament de l'Orquestrador (Defensa Activa):*** Programació de l'*script* de Python. Se segueix un desenvolupament basat en proves, verificant primer la lectura de les alertes de Suricata i, posteriorment, la capacitat d'interactuar amb la taula de filtratge de *iptables*.
- ***Sprint 5 - Validació i Avaluació:*** Fase final on es llençen els atacs des d'una màquina atacant Kali Linux per mesurar el temps de resposta del sistema i ajustar els algorismes de bloqueig per evitar falsos positius.

5 ARQUITECTURA DEL SISTEMA

5.1 Entorn de Virtualització

La Figura 1 mostra l'arquitectura genèrica del sistema de defensa activa, organitzada en quatre capes funcionals. La capa base recull les entrades externes mitjançant dos subsistemes independents: la detecció de trànsit de xarxa, a càrrec de Suricata, i el monitoratge de l'estat del sistema, a càrrec de Prometheus i Node Exporter. Ambdues fonts s'integren a la capa de correlació, que determina si una situació constitueix un atac real o un fals positiu. En funció d'aquesta decisió, la capa d'acció aplica les contramesures corresponents de forma autònoma. Finalment, la capa de visualització centralitza tota l'activitat en un panell de control jerarquitzat per nivells de risc.

El flux d'informació segueix una direcció ascendent: les capes inferiors capturen i processen les dades en brut, mentre que les capes superiors prenen decisions cada cop més abstractes fins a arribar a la resposta final i la seva representació visual.

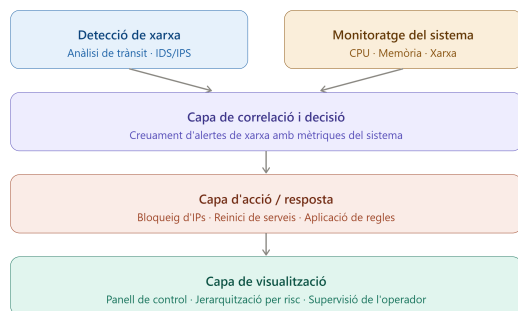


Fig. 1: Arquitectura genèrica del sistema de defensa activa per capes

La Figura 2 mostra la implementació concreta d'aquesta arquitectura en l'entorn virtualitzat desplegat, format per tres màquines virtuals interconnectades de forma aïllada a VirtualBox. Suricata assumeix el rol de detecció de trànsit de xarxa escoltant la interfície *enp0s3* (Segment-WAN), Prometheus i Node Exporter cobreixen el monitoratge del sistema, l'orquestrador en Python implementa la lògica de correlació i decisió aplicant regles *iptables*, i Grafana proporciona la capa de visualització per a l'operador.

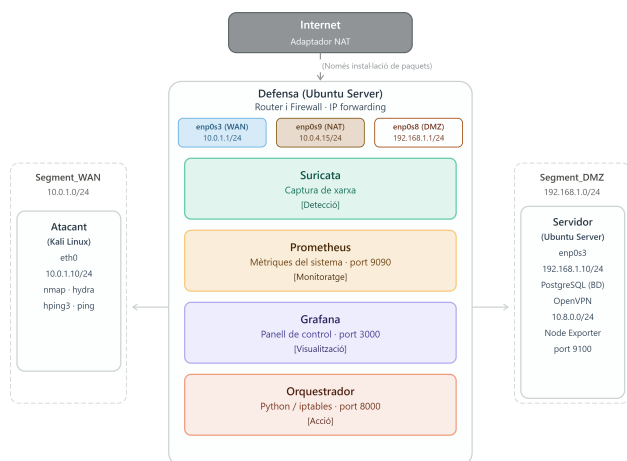


Fig. 2: Esquema Topologia Xarxa

Com s'observa a la Taula 1, la MV Defensa (Ubuntu Server) actua com a *router* i *firewall*, gestionant la connexió entre la WAN i la DMZ. A la MV Defensa, s'ha desactivat *NetworkManager* i s'ha implementat un *script rc.local* per assignar les adreces IP de manera permanent durant l'arrencada; d'aquesta manera se soluciona un conflicte detectat entre *netplan* i *NetworkManager*. Pel que fa a la MV Atacant (Kali Linux), la IP s'ha configurat estàticament mitjançant el fitxer */etc/network/interfaces*. Finalment, la MV Servidor (Ubuntu Server) obté la seva adreça automàticament a través de la xarxa interna de VirtualBox.

TAULA 1: CONFIGURACIÓ DE LA XARXA DE LES MVS

Configuració				
Nom MV	Interfície	IP/Màscara	Xarxa VirtualBox	Rol
Defensa	enp0s3	10.0.1.1/24	X. Int.: Segment-WAN	Firewall (WAN)
Defensa	enp0s8	192.168.1.1/24	X. Int.: Segment-DMZ	Firewall (DMZ)
Defensa	enp0s9	10.0.4.15/24	NAT (Internet)	
Atacant	eth0	10.0.1.10/24	X. Int.: Segment-WAN	Màquina atacant
Servidor	enp0s3	192.168.1.10/24	X. Int.: Segment-DMZ	Servidor DMZ

5.2 Components i versions

Les MVs Defensa i Servidor corren sobre Ubuntu Server 24.04 LTS amb kernel 6.8.0, mentre que la MV Atacant utilitza Kali Linux amb kernel 6.6.9 (amd64). S'ha escollit Ubuntu Server 24.04 LTS perquè, en el moment d'inici del projecte (febrer de 2026), era la darrera versió LTS disponible amb suport actiu fins al maig de 2029. Les versions no LTS, com la 25.10, només reben suport de manteniment de seguretat estàndard durant 9 mesos (fins al juliol de 2026 en el cas de la 25.10), enfront dels 5 anys de les versions LTS [16]. A la MV Defensa s'executen Suricata 7.0.3, Prometheus 2.x, Node Exporter 1.x, Grafana OSS 11.x i l'orquestrador en Python 3.12 amb la llibreria *prometheus_client*. A la MV Servidor s'integren OpenVPN 2.6.19, PostgreSQL 16.x i Node Exporter 1.x. Tot l'entorn corre sobre VirtualBox 7.0.20.

5.3 Vectors d'Atac

La Taula 2 mostra els cinc vectors d'atac i la resposta automatitzada que s'espera que l'orquestrador apliqui en cada cas. El principal repte de disseny en aquesta fase rau en la capacitat de diferenciar de manera unívoca un escaneig de ports d'un atac de denegació de servei per inundació (*SYN flood*), atès que ambdós vectors comparteixen la mateixa naturalesa estructural utilitzant paquets de tipus SYN.

Mentre que el reconeixement de ports busca identificar de manera dispersa quins serveis estan actius en un rang ampli de ports en un interval de temps controlat, la inundació es focalitza a saturar la capacitat de connexions d'un servei web en un port específic. Aquesta lògica de disseny requereix la creació de polítiques de detecció diferenciades a nivell de l'IDS, garantint que l'origen de cada incidència es classifiqui correctament abans que l'orquestrador decideixi quina acció executar.

TAULA 2: VECTORS D'ATAC I RESPOSTES ESPERADES

Vector	Eina	Nivell	Resposta
Ping ICMP	ping	Baix	Emmagatzemar a log
Escaneig nmap	nmap -sS	Alt	Bloquejar IP immediatament
Força Bruta SSH	hydra	Mitjà	Bloquejar si ≥ 2 intents
SYN flood	hping3	Alt	Bloquejar si CPU > CPU_MAX
DoS+CPU	hping3	Crític	Bloquejar + Reiniciar SSH

La classificació es correspon amb les regles locals de Suricata que es detallaran a la secció d'Implementació (SID:9000001 a SID:9000005)

6 IMPLEMENTACIÓ

Suricata s'instal·la a la MV Defensa i s'activa en mode promiscu [17] sobre la interfície *enp0s3* (Segment-WAN), punt d'entrada de tot el trànsit extern. S'ha modificat el fitxer de configuració principal *suricata.yaml*, definint la variable *HOME_NET* amb els rangs d'adreces de les xarxes internes a protegir (Segment-WAN 10.0.1.0/24 i Segment-DMZ 192.168.1.0/24) i acotant *EXTERNAL_NET* com a qualsevol trànsit exterior a aquests rangs.

Amb l'objectiu d'analitzar el trànsit, s'ha habilitat el registre de tipus *EVE JSON* a la ruta */var/log/suricata/eve.json*. Aquest fitxer recull de forma estructurada i en temps real les alertes del

sistema i les dades de les connexions (fluxos de xarxa, protocols HTTP i FTP), centralitzant la informació per a l'orquestrador.

A continuació, es mostra un exemple real d'un registre tal com queda registrat al fitxer `eve.json` en disparar-se una alerta per un escaneig de ports detectat per Suricata:

```
{ "timestamp": "2026-06-28T22
  :10:02.124580+0200", "flow_id
  ":192837465503110, "in_iface": "enp0s3", "
  event_type": "alert", "src_ip
  ": "10.0.1.10", "src_port": 42132, "dest_ip
  ": "10.0.1.1", "dest_port": 21, "proto": "TCP
  ", "alert": { "action": "allowed", "gid": 1, "
  signature_id": 9000005, "rev": 4, "signature
  ": "Escaneig nmap detectat", "category
  ": "", "severity": 3 }}
```

Per fer efectiva la detecció dels vectors planificats, s'han dissenyat cinc regles locals dins del fitxer de configuració `/etc/suricata/rules/local.rules`:

- **sid:9000001: Identificació de Trànsit ICMP.**

```
alert icmp any any -> $HOME_NET any (msg:"Paquet
  ICMP detectat"; sid:9000001; rev:1;)
```

Aquesta regla detecta sol·licituds ICMP dirigides a la xarxa protegida.

- **sid:9000002: Control de SYN flood al port 80.**

```
alert tcp any any -> $HOME_NET 80 (msg:"SYN flood
  detectat"; flags:S; flow:stateless; threshold
  : type both, track by_dst_port, count 500,
  seconds 1; sid:9000002; rev:3;)
```

Fixar el llindar en 500 paquets SYN per segon garanteix que cap trànsit legítim el superi, ja que una eina en mode `--flood` assoleix fàcilment milers de paquets per segon. El paràmetre `track by_dst_port` detecta un flux massiu dirigit al mateix port, mentre que `type both` no genera cap alerta fins a superar el llindar i, un cop superat, n'emet únicament una per segon, evitant la saturació del log [18].

- **sid:9000003: Detecció de Força Bruta SSH.**

```
alert tcp any any -> $HOME_NET 22 (msg:"Intent de
  connexió SSH"; flow:to_server,established;
  sid:9000003; rev:1;)
```

Avalua exclusivament connexions establertes (`flow:to_server,established`), és a dir, aquelles que han completat el *three-way handshake* TCP, garantint que cada detecció correspon a un intent d'autenticació real i no a un simple escaneig passiu [19].

- **sid:9000004: Monitoratge de Trànsit TCP elevat.**

```
alert tcp any any -> $HOME_NET any (msg:"Transit
  TCP elevat detectat"; flow:to_server,
  established; sid:9000004; rev:1;)
```

Actua com a monitor general per a connexions TCP establertes en qualsevol servei, permetent a l'orquestrador correlacionar el volum de trànsit amb l'estat del sistema. A diferència de *sid:9000002*, que es focalitza en el port 80, aquesta regla cobreix qualsevol port

TCP, actuant com a monitor general de connexions establertes. L'orquestrador correlaciona el volum d'alertes d'aquesta regla amb les mètriques de CPU de Prometheus per determinar si el trànsit constitueix un atac real o un pic de càrrega legítim.

- **sid:9000005: Anàlisi d'Escaneig nmap.**

```
alert tcp any any -> $HOME_NET 1:21,23:79,81:1024
  (msg:"Escaneig nmap detectat"; flags:S; flow:
  stateless; threshold: type threshold, track
  by_src, count 5, seconds 60; sid:9000005; rev
  :4;)
```

El llindar de 5 paquets SYN en 60 segons s'ha definit de forma conservadora [20]. Tot i que referències especialitzades situen els valors habituals de ràfega entre 7 i 20 paquets SYN per a entorns corporatius amb trànsit mixt [21], en un entorn de laboratori controlat amb activitat legítima reduïda, un llindar de 5 paquets és òptim per registrar qualsevol intent de sondeig sense risc de falsos positius. S'exclouen els ports 22 i 80 per evitar interferències amb altres atacs específics.

Aquestes regles segueixen les directrius de la documentació oficial de Suricata [18, 19, 20].

6.1 Serveis a la DMZ

A la MV Servidor (DMZ) s'han desplegat dos serveis que simulen un entorn corporatiu real amb dades sensibles i accés remot:

- **PostgreSQL (Base de Dades):** S'instal·la PostgreSQL [22] i es crea la taula empleats amb els camps que s'observen a la Figura 3: `id`, `nom`, `email`, `dni` i `salari`. S'insereixen tres registres de prova per simular un repositori de dades sensibles corporatives que l'atacant podria intentar exfiltrar en cas de compromís del servidor, justificant la necessitat del sistema de defensa activa. Atès que el Servidor no té accés directe a Internet, la connexió s'ha encaminat de forma temporal a través de la Defensa activant `ip_forward` i una regla MASQUERADE a `iptables`.

```
claudia@servidor:~$ sudo -u postgres psql -d dades -f /tmp/insert.sql
TRUNCATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
claudia@servidor:~$ sudo -u postgres psql -d dades -c "SELECT * FROM empleats;"
 id |      nom      |      email      |      dni      |  salari
----+-----+-----+-----+-----
 10 | Anna Puig     | anna@nomempresa.cat | 12345679A    |   4000
 11 | Clàudia Rodríguez | claudia@nomempresa.cat | 87654321B    |   3500
 12 | Aina Lopez    | aina@nomempresa.cat | 11122233C    |   3200
(3 rows)
```

Fig. 3: Taula Empleats PostgreSQL

- **OpenVPN (Accés remot):** S'instal·la OpenVPN i es configura la infraestructura de claus (PKI) amb *easy-rsa* [23]: generació de la CA, certificats de servidor i de client, paràmetres *Diffie-Hellman* i claus *TLS Authentication (ta.key)* per a protecció contra atacs de força bruta al port VPN. El fitxer de configuració del servidor i del client es basen en els exemples oficials d'OpenVPN inclosos a `/usr/share/doc/openvpn/examples/`. A la Figura 4 es confirma la connexió de client1 amb IP virtual 10.8.0.2. El servei s'integra al panell de Grafana

via Node Exporter, permetent visualitzar el seu estat operatiu juntament amb la resta de serveis del Servidor.

```
claudia@defensa:~$ ip a show tun0
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast stat
UNKNOWN group default qlen 500
    link/none
    inet 10.8.0.2/24 brd 10.8.0.255 scope global tun0
        valid_lft forever preferred_lft forever
    inet6 fe80::ef05:fa9a:309e:c933/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
claudia@defensa:~$ ping 10.8.0.1 -c 3
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data:
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.584 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.535 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=0.745 ms
--- 10.8.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2043ms
rtt min/avg/max/mdev = 0.535/0.621/0.745/0.089 ms
```

Fig. 4: Connexió VPN de client1 a IP virtual

6.2 Monitoratge amb Prometheus i Node Exporter

Es desplega Prometheus a la MV Defensa (port 9090) per recollir mètriques del sistema. La configuració s'ha realitzat mitjançant l'edició del fitxer `/etc/prometheus/prometheus.yml`, on s'han definit com a *targets* els serveis `localhost:9100` (Node Exporter de la Defensa), `192.168.1.10:9100` (Node Exporter del Servidor) i `localhost:8000` (l'orquestrador Python).

S'ha integrat Node Exporter (port 9100) en ambdues màquines virtuals per exportar mètriques de CPU, memòria, trànsit de xarxa per interfície i estat dels serveis. Aquesta visibilitat permet monitorar els pics d'activitat derivats dels vectors d'atac, proporcionant la base de dades necessària per a la visualització posterior de les incidències.

6.3 Grafana: Panell de Control

Grafana [24] s'ha desplegat a la MV Defensa (port 3000) utilitzant Prometheus com a font de dades. El *dashboard* principal conté diversos indicadors ordenats per criticitat: nivell de gravetat actual, IPs bloquejades, estat dels serveis (SSH, PostgreSQL, OpenVPN), alertes per tipus i gravetat, consum de CPU i memòria, trànsit de xarxa per interfície (incloent `tun0` per a la VPN), connexions TCP i ús de disc. Els panells es nodreixen de tres fonts diferenciades: les mètriques de sistema exposades per Node Exporter (CPU, memòria, disc i xarxa), les mètriques de l'orquestrador (alertes per tipus, IPs bloquejades i gravetat actual) i l'estat operatiu dels serveis crítics del Servidor (SSH, PostgreSQL i OpenVPN).

Aquesta eina permet visualitzar i analitzar el comportament del sistema davant d'amenaques, tal com s'il·lustra a la Figura 5. A les gràfiques s'observa una correlació directa entre atacs executats i els increments bruscos del trànsit entrant a la interfície `enp0s3` de la Defensa. El retorn immediat del flux de dades a la normalitat, un cop l'orquestrador aplica les regles, confirma l'efectivitat del sistema per detectar i mitigar múltiples amenaces de forma continuada i autònoma.



Fig. 5: Monitoratge del trànsit de xarxa durant l'execució d'atacs

Com a millora addicional, s'han implementat dos mecanismes de notificació proactiva que permeten a l'operador rebre avisos sense haver de consultar activament el panell de control. Aquesta capacitat és especialment rellevant en escenaris on l'operador no té el *dashboard* obert en tot moment, garantint que cap incident crític passi desapercebut.

En primer lloc, un *script* en *Bash* monitoritza el log de l'orquestrador en temps real i envia avisos visuals (*popup*) a l'escriptori de la MV Defensa, com mostra la Figura 6, mitjançant `notify-send`:

```
tail -f /var/log/orquestrador.log | while read line; do
if [[ "$line" == *"Alerta!"* ]]; then
    notify-send "Alerta d'Orquestrador" "$line"
fi
done
```

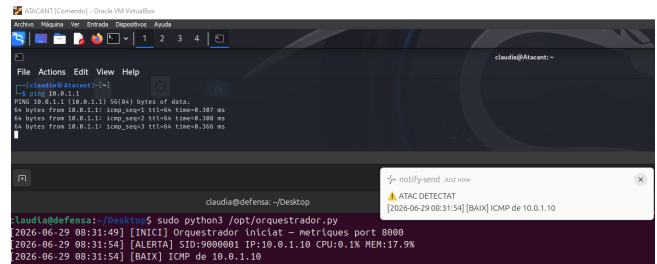


Fig. 6: Enviament de paquets ICMP detectat

En segon lloc, s'ha configurat Grafana per a l'enviament d'alertes via correu electrònic. Aquesta integració requereix la configuració d'un servidor SMTP al fitxer `grafana.ini`, especificant els paràmetres de connexió (host, port, usuari i adreça del destinatari). Gràcies a aquesta configuració, el sistema envia automàticament un missatge d'avís quan el nivell de gravetat de l'amenaça detectada (`orq-gravetat`) supera el valor 0. Un exemple d'aquesta notificació, amb l'assumpte i els detalls de la incidència, es troba a l'Apèndix A.

7 DESENVOLUPAMENT DE L'ORQUESTRA-DOR

L'orquestrador en Python [25]-[29] és el nucli del sistema de defensa activa, encarregat de correlacionar les alertes de *Suricata* amb les mètriques de Prometheus per diagnosticar amenaces i executar contramesures automàtiques. El funcionament detallat del diagrama de flux d'aquest procés es descriu a l'Apèndix B.

7.1 Lògica de Correlació

Tal com es detalla a la Taula 3, el sistema categoritza les incidències en quatre nivells de gravetat, ajustant la resposta

segons la tipologia de l'atac i l'estat dels recursos del servidor.

TAULA 3: NIVELLS DE GRAVETAT I ACCIONS

Nivell	Condicció	Acció
Baix	ICMP o SYN + CPU < 80%	Emmagatzemar al log
Mitjà	SSH $\geq$ SSH-MAX	Bloquejar IP
Alt	SYN + CPU > 80% o nmap	Bloquejar IP
Crític	DoS + CPU > 90%	Bloquejar + reiniciar SSH

Els llindars de configuració (SSH_MAX = 2 intents/5min, SYN_MAX = 50 paquets/5min) permeten un equilibri entre seguretat i disponibilitat, sent totalment ajustables segons l'entorn.

8 DESENVOLUPAMENT DE L'ORQUESTRADOR

L'orquestrador s'ha implementat a `/opt/orquestrador.py` (Python 3.12). Utilitza la llibreria `requests` per consultar l'API de *Prometheus* i `prometheus_client` per exportar mètriques al port 8000. El flux lògic de la funció `processar()` executa la mitigació segons els *Signature IDs* (SID) definits a *Suricata*.

Per garantir el funcionament òptim, l'*script* incorpora tres característiques clau:

- **Filtratge d'IPs locals:** Exclou la pròpia infraestructura (10.0.1.1, 192.168.1.1, localhost) per evitar bloquejos erronis.
- **Persistència:** Els comptadors d'atacs es desen a *stats.json* per mantenir l'estat en cas de reinici.
- **Inicialització de lectura del registre:** S'utilitza `f.seek(0, 2)` per processar únicament les alertes noves del fitxer *eve.json*.

El següent bloc mostra l'estructura del bucle principal de processament:

```
def processar(alerta, cpu, mem):
    # Filtre per evitar bloquejar la propia maquina
    if ip in ips_bloquejades or ip in IPS_LOCALS: return
    ...
    # Logica basada en el SID de Suricata (Exemple: DoS
    # Critic)
    # S'ha establert un llindar de 10% de CPU per a
    # proves (hauria de ser mes alt en produccio)
    if sid == 9000004:
        if cpu > 10.0:
            bloquejar(ip)
            subprocess.run(["systemctl", "restart", "ssh
            "])
    ...
def main():
    start_http_server(8000)
    with open(EVE_LOG, "r") as f:
        # Cursor al final per ignorar alertes antigues
        f.seek(0, 2)
        while True:
            # Processament de noves entrades del log
            ...
            time.sleep(5)
```

8.1 Persistència de Mètriques

Com s'ha esmentat, el fitxer *stats.json* assegura la persistència de l'estat en cas de reinici. La gestió de dades es divideix en:

- **Comptadors Temporals:** Les mètriques d'atac (SSH, SYN, ICMP) es reinicien cada 5 minuts per descartar activitat antiga i evitar falsos positius.
- **Gestió d'IPs:** Un cop bloquejada una IP, el sistema manté la restricció fins que l'operador realitza un desbloqueig manual (`iptables -F`).

8.2 Exemples de Resultats

Els resultats del sistema s'han validat mitjançant proves d'atac reals. Les figures 7 i 8 il·lustren la resposta de l'orquestrador davant de dos vectors d'atac.

Conceptualment, tant el *SYN flood* com el DoS d'alta intensitat només han d'escalar cap a nivells de criticitat alts o crítics si es constata que la capacitat de còmput del servidor s'està veient compromesa, evitant així bloquejos agressius davant de pics de trànsit legítim. No obstant, en un entorn de màquines virtuals de laboratori, la CPU del Servidor rarament supera el 0.1% durant les proves, ja que els recursos assignats a la MV no reflecteixen la càrrega d'un sistema de producció real. Per tal de validar empíricament aquest comportament i simular l'escenari de saturació, es van ajustar temporalment els llindars de control de la CPU a valors d'activació inferiors (> 9.0% i > 10.0%, respectivament). La generació de càrrega artificial es va realitzar al Servidor mitjançant l'execució en paral·lel de múltiples processos de bucle infinit en Python:

```
python3 -c 'while True: pass' &
python3 -c 'while True: pass' &
...
```

L'ús combinat de 16 processos concurrents va permetre elevar la càrrega del processador fins a un valor estable proper al 14%, superant els paràmetres de prova definits de manera controlada.

La Figura 7 mostra el comportament davant de trànsit ICMP (ping). En classificar-se com a nivell de gravetat *Baix*, l'orquestrador no aplica cap bloqueig, limitant-se a registrar l'activitat al log i a *Grafana*. Això confirma que el sistema detecta i classifica correctament incidències de baixa perillositat sense interrompre la connectivitat.

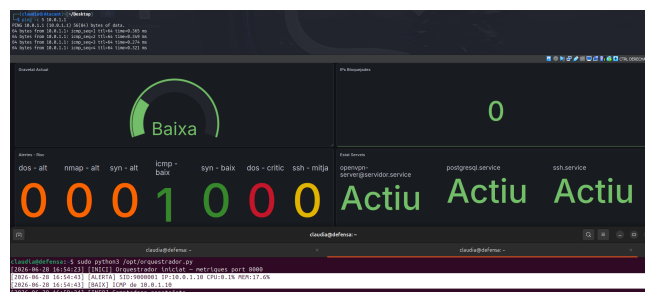


Fig. 7: Prova ICMP: Orquestrador i registre a Grafana

La Figura 8 mostra la resposta davant d'un escaneig de ports amb *nmap*. En detectar-se el patró (*SID:9000005*) i classificar-se com a nivell *Alt*, l'orquestrador bloqueja la IP origen immediatament mitjançant *iptables*. L'alerta i el bloqueig es reflecteixen en temps real al panell de control. Com s'ha comentat anteriorment, aquest bloqueig és persistent fins que l'operador realitza el desbloqueig manual.

9.5 Limitacions

El sistema presenta certes limitacions que haurien de ser abordades en una evolució cap a entorns de producció:

- **Entorn de Proves:** La baixa càrrega de CPU dels entorns virtuals ha requerit l'ús de llindars artificialment reduïts i càrrega sintètica per validar la resposta del sistema. En producció, aquests llindars s'haurien de recalibrar segons la càrrega de treball real.
- **Gestió d'Estats:** El bloqueig d'IPs és actualment permanent. Falta implementar una política de desbloqueig automàtic (per exemple, basat en temps) per mitigar l'impacte de possibles falsos positius.
- **Arquitectura:** El desplegament centralitzat ha estat una decisió tècnica per optimitzar els recursos del laboratori i facilitar la validació de la lògica de correlació. Tot i que crea un *Single Point of Failure* (un punt únic de fallada que pot comprometre tot el servei), el model permet demostrar l'eficàcia del sistema amb total transparència operativa. En un entorn de producció, aquest disseny evolucionaria cap a una arquitectura distribuïda per garantir l'Alta Disponibilitat (HA).

9.6 Comparativa amb Plataformes SOAR

TAULA 5: COMPARATIVA AMB SOLUCIONS EXISTENTS

Característica	IDS tradicional	SOAR comercial	Orquestrador propi
Temps de resposta	Minuts	Segons	~5-10 sec
Correlació detecció + sistema	No	Sí	Sí, a mida
Cost	Baix	Alt	Molt baix
Transparència	N/A	Tancada	Auditable
Escalabilitat	Limitada	Alta	Limitada
Desbloqueig automàtic	No	Sí	No (treball futur)

Aquesta comparativa evidencia que el sistema no pretén substituir una plataforma SOAR comercial, sinó validar que la correlació detecció-acció és viable amb eines de codi obert, oferint una alternativa eficient i de baix cost per a entorns acadèmics o petites organitzacions.

10 CONCLUSIONS

El projecte ha validat l'arquitectura d'un sistema de defensa autònom capaç d'executar contramesures en menys de 10 segons mitjançant la correlació d'alertes de Suricata i dades de rendiment de Prometheus. S'ha demostrat la viabilitat d'un model de SOAR minimalista amb eines *open-source*, garantint visibilitat *end-to-end* i una lògica de decisió auditable a un cost operatiu inferior a les solucions comercials.

Com a línies de treball futur es proposen:

1. **Adaptació Dinàmica:** Utilitzar *Machine Learning* per ajustar els llindars automàticament segons l'historial de trànsit, en lloc de fixar-los manualment [8, 9].
2. **Gestió Temporal:** Implementar un sistema de desbloqueig automàtic d'IPs.
3. **Modularitat:** Evolucionar l'orquestrador amb *playbooks* (procediments automatitzats) que permetin adaptar la defensa a diferents xarxes sense haver de modificar el codi [12].

4. **Validació contínua:** Integrar *Breach and Attack Simulation* (BAS) per comprovar, de forma constant, si les defenses són efectives davant d'atacs reals [30].

ÚS DE IA GENERATIVA

Durant el desenvolupament d'aquest treball s'han utilitzat eines d'IA generativa com a suport en les tasques següents:

- **Millora de l'Estil del Text:** S'ha utilitzat per millorar la cohesió lingüística i la claredat de la redacció.
- **Depuració de LaTeX:** L'eina ha servit d'ajut per identificar i corregir errors de compilació i de format en el document.
- **Resolució de Dubtes Tècnics:** S'ha emprat de forma puntual per entendre errors de configuració durant el desplegament de l'entorn virtual (Suricata, Prometheus, Grafana i l'orquestrador Python).
- **Cerca de Bibliografia:** S'ha utilitzat per identificar referències rellevants, que posteriorment han estat verificades i validades per l'autora a les fonts originals.

En tots els casos, l'ús de la IA ha estat supervisat per l'autora.

AGRAÏMENTS

L'autora vol agrair al seu tutor, Eduardo César Cabrera Flores, per la seva orientació, disponibilitat i *feedback* constructiu al llarg de tot el desenvolupament d'aquest treball. També vol expressar el seu agraïment als professors del Grau en Enginyeria Informàtica de la UAB pels coneixements transmesos durant aquests anys de formació. Finalment, un agraïment especial a la seva família, parella i amics pel suport incondicional rebut durant tot el procés.

REFERÈNCIES

[1] Suricata, *Suricata Features and Capabilities*, 2026. Disponible a: <https://suricata.io/features/>. [Accedit: febrer i març 2026].

[2] D. S. Ghazi, H. S. Hamid, M. J. Zaiter, and A. S. G. Behadili, "Snort Versus Suricata in Intrusion Detection," *Iraqi Journal of Information and Communication Technology*, vol. 7, no. 2, pp. 73–88, 2024. doi: 10.31987/ijict.7.2.290

[3] Prometheus, *Overview and Introduction to Prometheus*, 2026. Disponible a: <https://prometheus.io/docs/introduction/overview/>. [Accedit: febrer i març 2026].

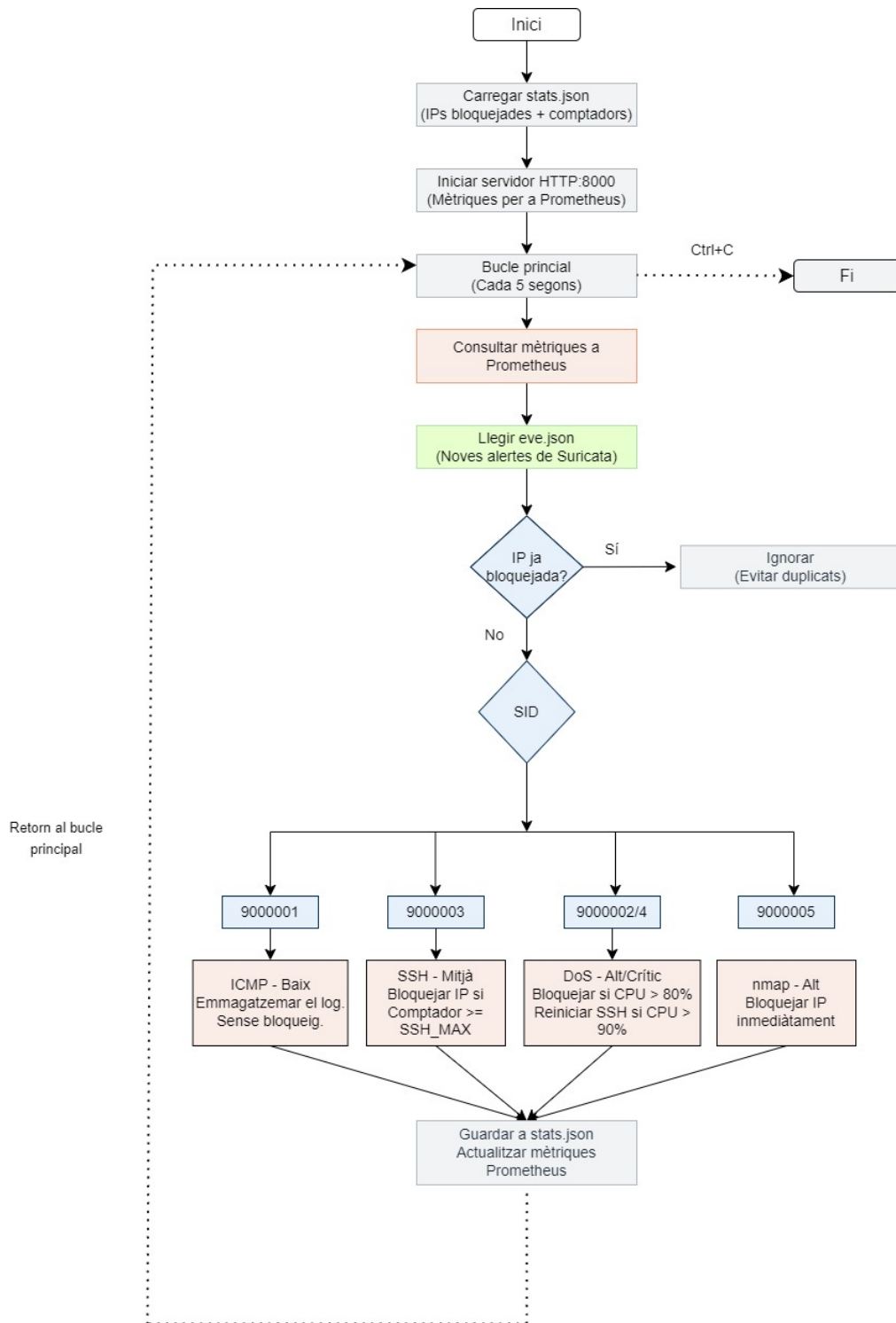
[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, Jul. 2017. doi: 10.1109/MC.2017.201

[5] Hispasec, *Un ciberataque deja sin servicio a la UAB, 2021*. Disponible a: <https://unnaaldia.hispasec.com/un-ciberataque-deja-sin-servicio-a-la-uab/>. [Accedit: febrer 2026].

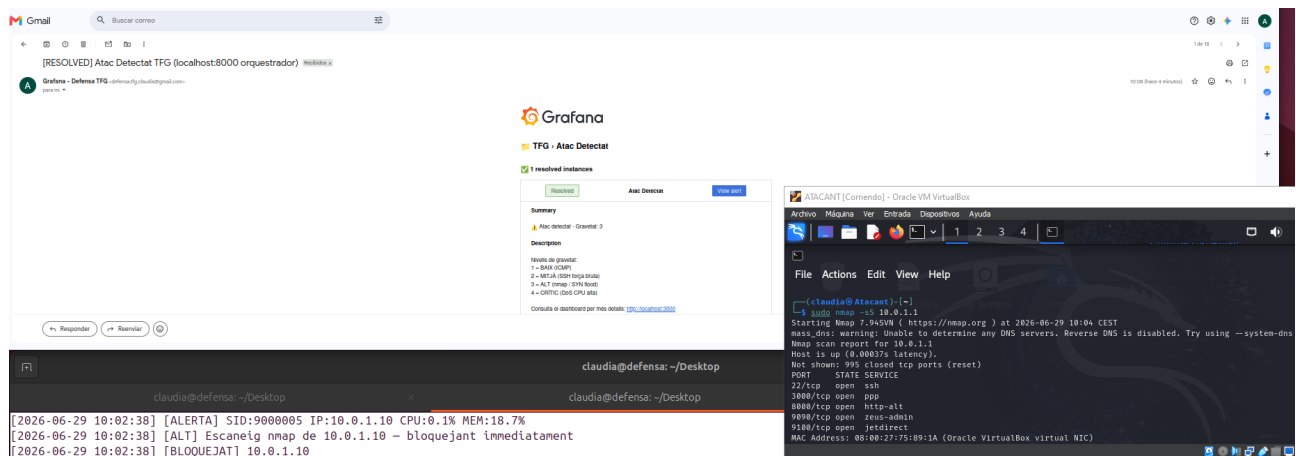
- [6] D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987. doi: 10.1109/TSE.1987.232894
- [7] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication 800-94, Feb. 2007. doi: 10.6028/NIST.SP.800-94
- [8] R. Almuhanha and S. Dardouri, "A deep learning/machine learning approach for anomaly based network intrusion detection," *Frontiers in Artificial Intelligence*, vol. 8, 2025. doi: 10.3389/frai.2025.1625891
- [9] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, Jan. 2021. doi: 10.1002/ett.4150
- [10] WebReactiva, *Claude Mythos: el model d'IA que preocupa el món de la ciberseguretat*, 2026. Disponible a: <https://www.webreactiva.com/blog/claude-mythos>. [Accedit: abril 2026].
- [11] M. Vielberth, "Security Information and Event Management (SIEM)," in *Encyclopedia of Cryptography, Security and Privacy*, S. Jajodia, P. Samarati, and M. Yung, Eds. Springer, Berlin, Heidelberg, 2021, pp. 1–3. doi: 10.1007/978-3-642-27739-9_1681-1
- [12] Palo Alto Networks, *SOAR vs. SIEM: What Is the Difference?*, 2026. Disponible a: <https://www.paloaltonetworks.com/cyberpedia/what-is-soar-vs-siem>. [Accedit: març 2026].
- [13] UnderDefense, *Palo Alto Networks Pricing 2026: Ultimate Guide for Security Products*, 2026. Disponible a: <https://underdefense.com/industry-pricings/palo-alto-networks-pricing-ultimate-guide-for-security-products/>. [Accedit: juny 2026].
- [14] I. Kotenko, D. Gaifulina, and I. Zelichenok, "Systematic Literature Review of Security Event Correlation Methods," *IEEE Access*, vol. 10, pp. 43387–43420, Apr. 2022. doi: 10.1109/ACCESS.2022.3168976
- [15] Scrum.org, *What is Scrum?*, 2026. Disponible a: <https://www.scrum.org/resources/what-scrum-module>. [Accedit: març 2026].
- [16] Canonical, *Ubuntu Release Cycle*, 2026. Disponible a: <https://ubuntu.com/about/release-cycle>. [Accedit: març 2026].
- [17] OPNsense, *Intrusion Prevention System*, 2026. Disponible a: <https://docs.opnsense.org/manual/ips.html>. [Accedit: març 2026].
- [18] Suricata, *Rule Thresholding*, 2026. Disponible a: <https://docs.suricata.io/en/latest/rules/thresholding.html>. [Accedit: març 2026].
- [19] Suricata, *Flow Keywords*, 2026. Disponible a: <https://docs.suricata.io/en/latest/rules/flow-keywords.html>. [Accedit: març 2026].
- [20] Suricata, *Rules Introduction*, 2026. Disponible a: <https://docs.suricata.io/en/latest/rules/intro.html>. [Accedit: març 2026].
- [21] F. Vlaicu, *Suricata Nmap Rules*, 2025. Disponible a: <https://vlaicu.io/posts/suricata-nmap-rules/>. [Accedit: maig 2026].
- [22] PostgreSQL, *SQL Create Table*, 2026. Disponible a: <https://www.postgresql.org/docs/current/sql-createtable.html>. [Accedit: març 2026].
- [23] Easy-RSA, *Easy-RSA Documentation*, 2026. Disponible a: <https://easy-rsa.readthedocs.io/en/latest/>. [Accedit: març 2026].
- [24] Grafana, *Install on Debian or Ubuntu*, 2026. Disponible a: <https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/>. [Accedit: abril 2026].
- [25] Python Software Foundation, *subprocess - Subprocess Management*, 2026. Disponible a: <https://docs.python.org/3/library/subprocess.html>. [Accedit: maig 2026].
- [26] Prometheus, *Python Client Library for Prometheus*, 2026. Disponible a: <https://github.com/prometheus/client-python>. [Accedit: maig 2026].
- [27] Suricata, *EVE JSON Output Format*, 2026. Disponible a: <https://docs.suricata.io/en/latest/output/eve/eve-json-format.html>. [Accedit: maig 2026].
- [28] Prometheus, *Prometheus HTTP API*, 2026. Disponible a: <https://prometheus.io/docs/prometheus/latest/querying/api/>. [Accedit: maig 2026].
- [29] Python Software Foundation, *time — Time Access and Conversions*, 2026. Disponible a: <https://docs.python.org/3/library/time.html>. [Accedit: maig 2026].
- [30] V. Lopez, S2GRUPO, *Breach and Attack Simulation: metodologia i eines clau*, 2025. Disponible a: <https://s2grupo.es/breach-and-attack-simulation/>. [Accedit: juny 2026].

APÈNDIX

A Diagrama de Flux de l'Orquestrador



B Notificació Automàtica



Gmail

[RESOLVED] Atac Detectat TFG (localhost8000 orquestrador)

Grafana - Defensa TFG (defensa.tfg@uab.cat)

TFG - Atac Detectat

1 resolved instances

Summary

Atac detectat - Gravitat 3

Descripció

Nivells de gravetat:

- 1 - BLOC CORT
- 2 - MITJA (orden forja trau)
- 3 - ALT (nmap) (alta)
- 4 - CRÍTIC (DMS CPU alta)

Consulta el dashboard per més detalls: [https://localhost:3000](#)

claudia@defensa: ~/Desktop

```
[2026-06-29 10:02:38] [ALERTA] SID:9000005 IP:10.0.1.10 CPU:0.1% MEM:18.7%
[2026-06-29 10:02:38] [ALT] Escaneig nmap de 10.0.1.10 - bloquejant immediatament
[2026-06-29 10:02:38] [BLOQUEJAT] 10.0.1.10
```

claudia@defensa: ~/Desktop

```
claudia@defensa:~/Desktop
$ sudo nmap -sS 10.0.1.1
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-06-29 10:04 CEST
nmap_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns
Nmap scan report for 10.0.1.1
Host is up (0.00037s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
3000/tcp   open  php
8000/tcp   open  http-alt
9090/tcp   open  zeus-admin
2100/tcp   open  jetdirect
MAC Address: 08:00:27:75:89:1A (Oracle VirtualBox virtual NIC)
```