Detecció d'intrusions i mitigació activa

Índex

Cicle formatiu: CFGS Administració de sistemes informàtics en xarxa (ASIX)

Mòdul: 0378 - Seguretat i alta disponibilitat

Sistema operatiu: Ubuntu Server 26.04 LTS

NOTA

Aquest document cobreix l'**RA2** del mòdul. *"Implanta mecanismes de seguretat activa, seleccionant i executant contramesures davant d'amenaces o atacs al sistema"*. Concretament, treballa els criteris d'avaluació relacionats amb el **monitoratge del trànsit de xarxa**, els **intents de penetració** i els **sistemes de detecció d'intrusions**.

Introducció

En un sistema informàtic connectat a una xarxa, no n'hi ha prou amb un tallafocs. Un tallafocs decideix **què entra i què surt** segons regles estàtiques (ports, IP, protocols), però no analitza el **contingut** del trànsit ni reacciona davant patrons d'atac que no coneix d'entrada.

Per cobrir aquestes mancances, en aquest document treballarem dues eines complementàries:

- **Suricata**: un IDS/IPS (*Intrusion Detection/Prevention System*) que analitza el trànsit de xarxa en temps real i detecta (o bloqueja) activitat maliciosa a partir de signatures.
- **Fail2ban**: una eina de mitigació reactiva que vigila fitxers de log i bloqueja automàticament IP que mostren comportament abusiu (per exemple, intents de força bruta per SSH).

Combinades, formen una capa de **seguretat activa** que complementa el tallafocs (iptables/nftables/ufw) i encaixa directament amb els continguts del RA2: *"Seguretat en la xarxa corporativa: monitoratge del trànsit, riscos potencials dels serveis en xarxa, intents de penetració"*.

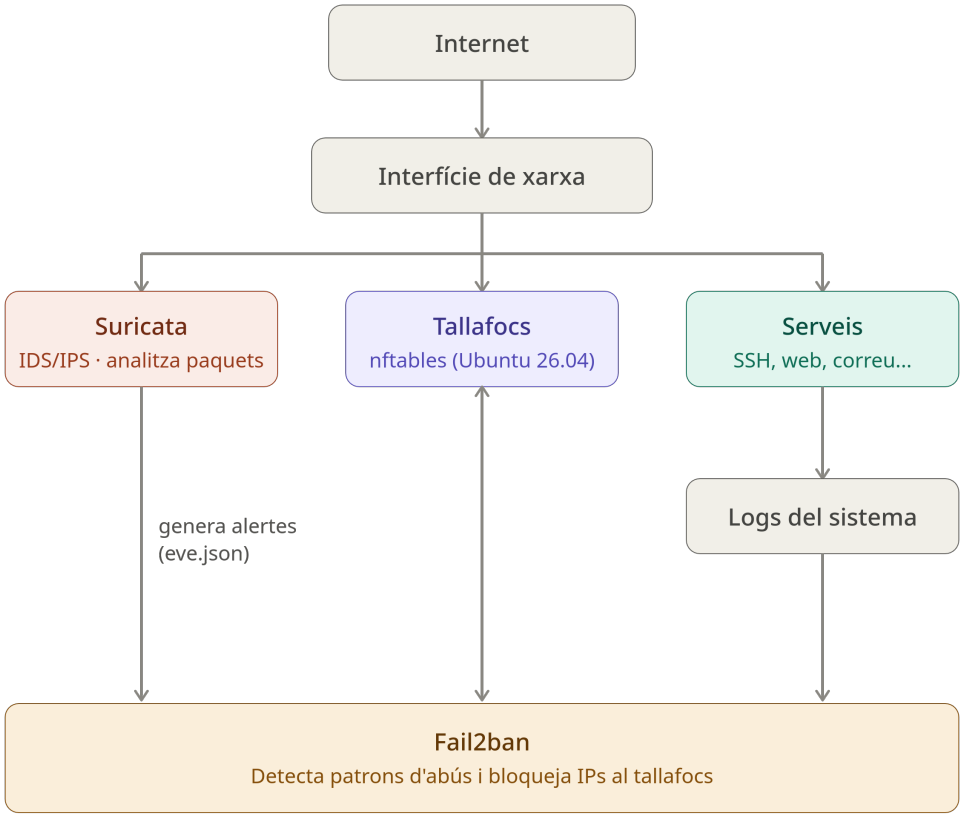

Figura 1: Arquitectura Fail2ban

Part 1: Suricata

Figura 2: Suricata logo

1.1. Què és Suricata?

Suricata és un motor IDS/IPS de codi obert desenvolupat per l'*Open Information Security Foundation* (OISF). Analitza el trànsit de xarxa a escala de paquet i el compara amb un conjunt de **regles** (signatures) per detectar:

- Escanejos de ports
- Explotació de vulnerabilitats conegudes
- Trànsit de programari maliciós i botnets (comunicació amb servidors de comandament i control)
- Anomalies en protocols (HTTP, DNS, TLS…)
- Intents de penetració i moviments laterals

Tipus de funcionament

Mode	Descripció
IDS (Intrusion Detection System)	Mode passiu: analitza còpies del trànsit i genera alertes, però no el bloqueja.
IPS (Intrusion Prevention System)	Mode actiu: s'interposa en el camí del trànsit (`inline`) i pot descartar paquets maliciosos en temps real.

Característiques principals

- **Multithreading natiu**: pot repartir l'anàlisi entre múltiples nuclis de CPU, cosa que li permet treballar a velocitats de xarxa altes.
- **Motor de regles compatible amb Snort**: la majoria de regles escrites per a Snort (VRT, Emerging Threats) funcionen directament a Suricata.
- **Inspecció a nivell d'aplicació**: entén protocols com HTTP, TLS, DNS, SMB, SSH… i pot generar alertes basades en camps específics (per exemple, un `User-Agent` sospitós).
- **Extracció de fitxers**: pot capturar fitxers transmesos per la xarxa (per exemple, un executable descarregat per HTTP) per analitzar-los posteriorment.
- **Sortida estructurada EVE `JSON`**: registra els esdeveniments en format JSON, ideal per integrar amb SIEM, Elasticsearch, Fail2ban o scripts propis.

1.2. Instal·lació a Ubuntu Server 26.04

Ubuntu Server 26.04 inclou Suricata als dipòsits oficials, però és recomanable utilitzar el PPA oficial de l'OISF per tenir sempre la darrera versió estable amb el conjunt de regles actualitzat.

Afegeix el dipòsit oficial de Suricata (recomanat)

```
sudo add-apt-repository ppa:oisf/suricata-stable
```

Actualitza el sistema

```
sudo apt update
```

Instal·la Suricata

```
sudo apt install suricata
```

Comprova la versió instal·lada

```
suricata --build-info | head -n 5
```

CONSELL

Si el PPA no és accessible (per exemple, en un laboratori sense sortida a Internet), es pot instal·lar directament amb `sudo apt install suricata` des dels dipòsits estàndard d'Ubuntu.

Identifica la interfície de xarxa que vols monitorar

```
ip -brief link show
```

Suposarem que la interfície que vols vigilar és `enp0s3`.

1.3. Configuració bàsica

El fitxer principal de configuració és `/etc/suricata/suricata.yaml`. És un fitxer extens (YAML), i els paràmetres clau per començar són:

```
sudo nano /etc/suricata/suricata.yaml
```

Defineix la xarxa local (HOME_NET)

```
vars:
  address-groups:
    HOME_NET: "[192.168.2.0/24]"
    EXTERNAL_NET: "!$HOME_NET"
```

IMPORTANT

Definir correctament `HOME_NET` és essencial: moltes regles distingeixen entre trànsit intern i extern, i una configuració incorrecta genera falsos negatius o un excés d'alertes irrellevants.

Configura la interfície de captura

```
af-packet:
  - interface: enp0s3
    threads: auto
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
```

Ubicació dels logs

```
default-log-dir: /var/log/suricata/

outputs:
  - eve-log:
      enabled: yes
      filetype: regular
      filename: eve.json
      types:
        - alert
        - http
        - dns
        - tls
        - ssh
        - flow
```

El fitxer `eve.json` és el punt central d'integració: hi trobem totes les alertes en format JSON, un registre per línia.

1.4. Gestió de regles (signatures)

Suricata utilitza `suricata-update` per descarregar i gestionar conjunts de regles (Emerging Threats Open, ET Pro, regles pròpies…).

Instal·la suricata-update (normalment ja inclòs)

```
sudo apt install suricata-update
```

Descarrega el conjunt de regles per defecte (Emerging Threats Open)

```
sudo suricata-update
```

Llista les fonts de regles disponibles

```
sudo suricata-update list-sources
```

Activa una font addicional, per exemple

```
sudo suricata-update enable-source et/open
sudo suricata-update
```

Les regles descarregades es desen a `/var/lib/suricata/rules/suricata.rules`, i cal referenciar-les a `suricata.yaml`:

```
sudo nano /etc/suricata/suricata.yaml
```

```
default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
```

Escriu una regla pròpia

Una regla de Suricata (compatible amb Snort) segueix aquesta estructura:

```
acció protocol IP_origen port_origen -> IP_desti port_desti (opcions)
```

Exemple: alertar davant intents de connexió SSH des de fora de la xarxa local:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"Intent de connexio
 ↪  SSH des de xarxa externa"; flow:to_server; sid:1000001; rev:1;)
```

Es pot desar a `/etc/suricata/rules/local.rules` i referenciar-lo:

```
sudo nano /etc/suricata/suricata.yaml
```

```
rule-files:
  - suricata.rules
  - local.rules
```

1.5. Mode IDS vs. mode IPS (inline)

Mode IDS (per defecte, af-packet)

En mode IDS, Suricata rep una còpia del trànsit i només genera alertes; no interfereix amb els paquets.

```
sudo systemctl enable suricata
sudo systemctl start suricata
sudo systemctl status suricata
```

Mode IPS amb NFQUEUE

Per bloquejar trànsit en temps real, cal desviar els paquets cap a Suricata mitjançant `iptables`/`nftables` i el mode `NFQUEUE`:

Desvia trànsit cap a la cua NFQUEUE 0

```
sudo iptables -I FORWARD -j NFQUEUE --queue-num 0
sudo iptables -I INPUT -j NFQUEUE --queue-num 0
```

I executa Suricata en mode NFQ:

```
sudo suricata -c /etc/suricata/suricata.yaml -q 0
```

Cal, a més, canviar l'acció de les regles d'`alert` a `drop` perquè el mode IPS descarti realment els paquets:

```
drop tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"Bloqueig SSH extern";
 ↪  sid:1000002; rev:1;)
```

AVÍS

El mode IPS introdueix Suricata **al camí del trànsit**. Una configuració incorrecta pot deixar el sistema sense connectivitat. Cal provar sempre primer en mode IDS (`alert`) abans de passar regles a `drop` en producció.

1.6. Comprova el funcionament

Valida la configuració sense arrencar el servei

```
sudo suricata -T -c /etc/suricata/suricata.yaml -v
```

Segueix les alertes en temps real

```
sudo tail -f /var/log/suricata/eve.json | jq
 ↪  'select(.event_type=="alert")'
```

Genera trànsit de prova (des d'una altra màquina)

```
nmap -sS 192.168.2.10
```

Si tot funciona correctament, hauries de veure alertes corresponents a l'escaneig de ports al fitxer `eve.json`.

Part 2: Fail2ban

Figura 3: Fail2ban logo

2.1. Què és Fail2ban?

Fail2ban **no és un IDS de xarxa**: no analitza paquets ni entén protocols. És una eina de **mitigació reactiva basada en logs**. El seu funcionament es resumeix en tres passos:

1. **Vigila** fitxers de log (SSH, Apache, Postfix, o el propi `eve.json` de Suricata).
2. **Detecta** patrons repetits mitjançant expressions regulars (*filters*) --- per exemple, múltiples intents d'inicis de sessió SSH fallits en poc temps.
3. **Actua** afegint una regla temporal al tallafocs (`iptables`/`nftables`/`firewalld`) que bloqueja la IP origen durant un període configurable.

Components de Fail2ban

Component	Funció
Jail (presó)	Defineix quin servei es vigila, quin filtre s'aplica i quina acció s'executa.
Filter	Expressió regular que identifica una línia de log com a “intent fallit”.
Action	Comanda que s'executa quan se supera el llindar (normalment, bloquejar la IP).
Ban time / find time / max retry	Paràmetres temporals: durant quant de temps es bloqueja, en quina finestra es compten els intents, i quants intents es toleren.

2.2. Instal·lació a Ubuntu Server 26.04

Actualitza la llista de paquets

```
sudo apt update
```

Instal·la Fail2ban

```
sudo apt install fail2ban
```

Habilita perquè s'inicïi automàticament en cada arrencada del sistema:

```
sudo systemctl enable --now fail2ban
```

Verifica l'estat del servei

```
sudo fail2ban-client status
```

IMPORTANT

Mai s’ha d’editar `/etc/fail2ban/jail.conf` directament: aquest fitxer se sobreescriu en actualitzacions. Cal crear `/etc/fail2ban/jail.local` (configuració global) o altres fitxers dins `/etc/fail2ban/jail.d/` (configuració per servei), que tenen prioritat.

Particularitat d'Ubuntu 26.04: nftables com a acció de bloqueig per defecte

A diferència de versions anteriors (que bloquejaven amb `iptables-multiport`), el paquet de Fail2ban a Ubuntu 26.04 ja porta configurat per defecte l'ús de **nftables** com a mecanisme de bloqueig, definit al fitxer `/etc/fail2ban/jail.d/defaults-debian.conf`:

```
sudo cat /etc/fail2ban/jail.d/defaults-debian.conf
```

```
[DEFAULT]
banaction = nftables
banaction_allports = nftables[type=allports]
```

```
[sshd]
backend = systemd
journalmatch = _SYSTEMD_UNIT=ssh.service + _COMM=sshd
enabled = true
bantime = 600
findtime = 3m
maxretry = 5
action = %(action_mw)s
```

Aquest fitxer ja arriba amb el jail `sshd` **activat per defecte**, llegint directament del `journal` de systemd (`backend = systemd`) en lloc d'un fitxer de log pla. És per això que a Ubuntu 26.04 no cal indicar `logpath` per SSH: n'hi ha prou amb el `journalmatch`.

CONSELL

Seguint la convenció del paquet Debian/Ubuntu, en lloc de posar tota la configuració a un únic `jail.local`, és més net crear **un fitxer per servei** dins `/etc/fail2ban/jail.d/` (per exemple `sshd.local`, `apache-auth.conf`, `suricata.conf`...). Fail2ban els llegeix tots per ordre alfabètic i els combina.

2.3. Configuració bàsica: jail.local

```
sudo nano /etc/fail2ban/jail.local
```

```
[DEFAULT]
# IPs que mai es bloquegen (loopback i xarxa local de gestió)
ignoreip = 127.0.0.1/8 ::1 192.168.2.0/24
# Temps de bloqueig
bantime  = 1d
# Finestra de temps per comptar intents
findtime = 5m
# Nombre màxim d'intents abans de bloquejar
maxretry = 5
# Adreça de destí i remitent per a notificacions per correu
destemail = root@localhost
sender    = root@resolute.thos.local
```

Aquests valors del `[DEFAULT]` a `jail.local` sobreescriuen els que ja porta `/etc/fail2ban/jail.conf`, però **no** el `banaction`/`backend` propis de `defaults-debian.conf`, que continuen aplicant-se llevat que els sobreescriviu explícitament al vostre propi fitxer.

Per ajustar el jail `sshd` amb valors propis (per exemple, més restrictius que el que porta per defecte Ubuntu), és millor crear un fitxer dedicat:

```
sudo nano /etc/fail2ban/jail.d/sshd.local
```

```
[sshd]
enabled  = true
backend  = systemd
maxretry = 3
bantime  = 30m
# %(action_mw)s = bloqueig + correu amb informació whois
# %(action_mwl)s = bloqueig + correu amb whois i les línies de log
 ↪  implicades
action   = %(action_mwl)s
```

Reinicia (o recarrega) el servei per aplicar canvis:

```
sudo systemctl restart fail2ban
# o, sense tallar les sessions de bans actius:
sudo systemctl reload fail2ban
```

2.4. Gestió i monitoratge

Estat general (jails actius)

```
sudo fail2ban-client status
```

Estat detallat d'un jail concret

```
sudo fail2ban-client status sshd
```

Desbloqueja una IP manualment

```
sudo fail2ban-client set sshd unbanip 203.0.113.45
```

Bloqueja manualment una IP (o un rang)

```
sudo fail2ban-client set sshd banip 203.0.113.45
sudo fail2ban-client set sshd banip 203.0.113.0/28
```

Desbloqueja totes les IP de tots els jails

```
sudo fail2ban-client unban --all
```

Consulta els logs propis de Fail2ban

```
sudo tail -f /var/log/fail2ban.log
```

Exemple de sortida de `fail2ban-client status sshd` un cop hi ha hagut un bloqueig:

```
Status for the jail: sshd
|- Filter
```

```
|  |- Currently failed: 0
|  |- Total failed:     5
|  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   203.0.113.45
```

Verifica el bloqueig a baix nivell (nftables)

Com que a Ubuntu 26.04 l'acció per defecte és `nftables` (i no `iptables`), el bloqueig real es pot comprovar amb:

```
sudo nft list ruleset
```

```
table inet f2b-table {
    set addr-set-sshd {
        type ipv4_addr
        elements = { 203.0.113.45 }
    }

    chain f2b-chain {
        type filter hook input priority filter - 1; policy accept;
        tcp dport 22 ip saddr @addr-set-sshd reject with icmp
 ↪  port-unreachable
    }
}
```

Fail2ban crea una taula `inet f2b-table` amb un `set` d'adreces per jail i una cadena (`chain`) que hi enganxa la política de rebuig. Aquest és exactament el mecanisme intern que substitueix les antigues regles `iptables -I INPUT ... -j DROP` d'altres distribucions o versions.

2.5. Jails habituals més enllà de SSH

`/etc/fail2ban/jail.conf` inclou desenes de definicions de jails predefinits per a serveis habituals. Per veure'n la llista completa:

```
grep '^\[' /etc/fail2ban/jail.conf | tail -n +3
```

```
[sshd]
[dropbear]
[selinux-ssh]
[apache-auth]
[apache-badbots]
[apache-noscript]
[apache-overflows]
```

```
[apache-nohome]
[apache-botsearch]
[apache-fakegooglebot]
[apache-modsecurity]
[apache-shellshock]
[openhab-auth]
...
```

Cadascun ja porta el `filter` i el `logpath`/`journalmatch` correctes definits al mateix `jail.conf`; per activar-los només cal un fitxer breu a `jail.d/` amb `enabled = true`:

Apache amb autenticació bàsica

```
sudo nano /etc/fail2ban/jail.d/apache-auth.conf
```

```
[apache-auth]
enabled  = true
bantime  = 600
findtime = 3m
maxretry = 5
action   = %(action_mw)s
```

Vsftpd

```
sudo nano /etc/fail2ban/jail.d/vsftpd.conf
```

```
[vsftpd]
enabled = true
action  = %(action_mw)s
```

Postfix amb autenticació SASL

```
sudo nano /etc/fail2ban/jail.d/postfix-sasl.conf
```

```
[postfix-sasl]
enabled = true
action  = %(action_mw)s
```

Rellegeix la nova configuració

```
sudo systemctl reload fail2ban
```

Verifica l'estat del servei

```
sudo fail2ban-client status
```

```
Status
|- Number of jail: 4
`- Jail list: apache-auth, postfix-sasl, sshd, vsftpd
```

Crea un filtre personalitzat

Els filtres es defineixen a `/etc/fail2ban/filter.d/`. Exemple d'un filtre bàsic:

```
# /etc/fail2ban/filter.d/exemple.conf
[Definition]
failregex = ^.*Intent fallit des de <HOST>.*$
ignoreregex =
```

I s'associa a un jail nou a `jail.local`:

```
[exemple]
enabled  = true
filter   = exemple
logpath  = /var/log/exemple.log
maxretry = 4
bantime  = 1h
```

Part 3: Integració Suricata + Fail2ban

Aquesta és la part central per al RA2 del mòdul: combinar un IDS de xarxa (Suricata) amb un sistema de bloqueig automàtic (Fail2ban) perquè les alertes de Suricata acabin traduint-se en bloquejos reals al tallafocs.

3.1. Per què integrar-los?

Suricata **detecta**, però, en mode IDS (el més habitual i segur en un entorn de pràctiques), no **bloqueja**. Fail2ban pot llegir el fitxer `eve.json` de Suricata i bloquejar automàticament les IP que generen alertes, sense necessitat de passar Suricata a mode IPS/inline (que és més complex i arriscat).

3.2. Filtre de Fail2ban per a eve.json

Creem un filtre que interpreti les alertes de Suricata:

```
sudo nano /etc/fail2ban/filter.d/suricata.conf
```

```
# /etc/fail2ban/filter.d/suricata.conf
[Definition]
failregex = ^.*"event_type":"alert".*"src_ip":"<HOST>".*$
ignoreregex =
```

NOTA

Aquest filtre és una versió simplificada amb finalitat didàctica. En un entorn real, convé filtrar per `signature` o `severity` per evitar bloquejar per alertes de baixa gravetat (per exemple, només actuar davant `"severity":1`).

3.3. Jail per a Suricata

```
sudo nano /etc/fail2ban/jail.local
```

Afegeix:

```
[suricata]
enabled  = true
filter   = suricata
logpath  = /var/log/suricata/eve.json
maxretry = 1
findtime = 10m
bantime  = 1h
# A Ubuntu 26.04 l'acció per defecte ja bloqueja via nftables
 ↪  (banaction = nftables,
# heretat de /etc/fail2ban/jail.d/defaults-debian.conf); només cal
 ↪  triar si es vol
# notificació per correu:
action   = %(action_mwl)s
```

Es recomana desar-ho com a fitxer independent `/etc/fail2ban/jail.d/suricata.conf`, seguint la mateixa convenció “un fitxer per servei” que ja fa servir Ubuntu per a `sshd`.

Amb `maxretry = 1`, una sola alerta de Suricata és suficient per bloquejar la IP: té sentit perquè Suricata ja fa la feina d'anàlisi profunda; Fail2ban només n'executa la conseqüència.

Reinicia

```
sudo systemctl restart fail2ban
```

Verifica l'estat

```
sudo fail2ban-client status suricata
```

3.4. Prova d'integració extrem a extrem

1. Des d'una màquina externa a `HOME_NET`, llança un escaneig:

   ```
   nmap -sS 192.168.2.10
   ```

2. Comprova que Suricata genera l'alerta:

   ```
   sudo tail -n 5 /var/log/suricata/eve.json | jq
    ↪  'select(.event_type=="alert")'
   ```

3. Comprova que Fail2ban ha detectat la línia i ha bloquejat la IP:

   ```
   sudo fail2ban-client status suricata
   sudo nft list ruleset | grep -A 5 "addr-set-suricata"
   ```

Si els tres passos es compleixen, l'esquema de detecció (Suricata) → mitigació (Fail2ban) funciona correctament.

3.5. Comparativa de responsabilitats

Funció	Suricata	Fail2ban
Analitza contingut de paquets	Sí	No
Entén protocols d'aplicació (HTTP, DNS, TLS)	Sí	No
Vigila fitxers de log	No (genera logs)	Sí
Bloqueja les IP al tallafocs	Només en mode IPS/inline	Sí (sempre)
Detecta escanejos i signatures d'atac conegudes	Sí	No
Detecta força bruta (SSH, web, correu)	Parcialment (segons regles)	Sí (nadiu)
Complexitat d'implantació	Alta	Baixa
Risc de tallar trànsit legítim	Alt (mode IPS)	Baix (bans temporals)

Referències

- Documentació oficial de Suricata: https://docs.suricata.io/
- Emerging Threats (regles): https://rules.emergingthreats.net/
- Documentació oficial de Fail2ban: https://github.com/fail2ban/fail2ban/wiki
- Dipòsit PPA de Suricata: https://launchpad.net/~oisf/+archive/ubuntu/suricata-stable
- Server World --- Fail2Ban a Ubuntu 26.04 (backend systemd/journald i acció nftables): https://www.server-world.info/en/note?os=Ubuntu_26.04&p=fail2ban

Versions d'aquest document

- HTML - 0378RA2.html
- PDF - 0378RA2.pdf
- ODT - 0378RA2.odt
- MD - 0378RA2.md