



Année 2024-2025

Annexe

Technique



Maintenance SI



Benoît ROBART

Table des matières

Sommaire

1. Maintenance du Système d'Information (SI)	page 3
1.1 Objectifs de la maintenance	page 3
1.2 Tâches régulières (planning hebdomadaire)	page 3
1.3 Outils de supervision et d'alerte	page 4
1.4 Indicateurs de continuité (KPI SI)	page 5
1.5 Évolutivité et gestion des incidents	page 5
2.0 Etats des lieux	page 5
2.1 Informations Générales	page 6
2.2 Capacités Système Actuelles	page 6
2.3 Risques / Points faibles détectés	page 6
2.4 Hypothèse de charge + recommandations	page 7
3.0 Intégration des outils de sécurité : Wazuh, Suricata, Fail2Ban, Loki	page 7
3.1 Préambule	page 7-8
3.2 Pourquoi installer la suite de sécurité, Suricata / Fail2ban / Wazuh / Loki : CONTEXTE	page 9 – 10
3.3 Installation Suricata et Fail2Ban puis Wazuh et Loki.	page 10 - 21
Suricata	page 10 - 15
Fail2Ban	page 15 - 17
Wazuh	page 17 - 19
Loki	page 19 – 21
3.4 Ce qui a fonctionné / Erreurs rencontrées et problèmes résolus	page 21 - 22
3.5 Couplage avec les outils de monitoring : Wazuh–Prometheus–Grafana–Loki	page 22 - 28
4.0 Paramétrage du monitoring et des outils de sécurité	page 28
4.1 Scripts & Tests	page 29 - 30
4.2 Résumé des activations	page 30 - 31
4.3 Protocoles d'exploitation (RACI / ACL)	page 32
4.4 Paramétrage du monitoring et des outils de sécurité	page 33
4.5 Schéma de fonctionnement complet	page 33 -36
4.6 Tests & charges dans le cadre de la sécurisation du SI	page 37 – 39
SECTION : Supervision avancée	page 37 - 39
MISE EN OEUVRE	page 29 - 42
1. Crédit + automatisation :	page 42 - 43
2. Récapitulatif des étapes fonctionnelles	page 43 - 44

1. Maintenance du Système d'Information (SI)

L'approche consistant à centraliser l'intégralité des tâches réalisées dans une annexe autonome est nécessaire. Elle permet de dissocier le déroulement réel, parfois chaotique ou itératif (journal de bord), de la structure finale stabilisée, rationalisée et exploitable pour un audit ou un transfert technique. Cette séparation entre *le vécu du projet (journal)* et *la modélisation fonctionnelle du SI sécurisé (Annexe 4)* offre un double intérêt :

- Elle met en valeur l'effort de consolidation et de formalisation, propre à un travail professionnel.
- Elle répond aux exigences de clarté et de traçabilité pour les responsables techniques, les commanditaires ou les auditeurs.

1.1 Objectifs de la maintenance

- Garantir la disponibilité du service (Jitsi).
- Prévenir les interruptions et incidents.
- Assurer une évolutivité contrôlée.
- Réduire le temps de détection et de correction des anomalies.

1.2 Tâches régulières (planning hebdomadaire)

Jour	Tâche	Outil / Lieu
Lundi	Vérification des certificats SSL	<code>certbot renew --dry-run</code>
Mardi	Contrôle des connexions SSH	<code>grep "Accepted" /var/log/auth.log</code>
Mercredi	Sauvegarde snapshot	OpenStack ou script bash
Jeudi	Audit des logs NGINX	<code>/var/log/nginx/</code>
Vendredi	Mise à jour système + paquets	<code>apt update && apt upgrade</code>
Samedi	(Option) Backup local NAS	<code>rsync</code>
Dimanche	Vérification tableau Grafana	Alertes visuelles ou seuils

1.3 Outils de supervision et d'alerte

Outils	Fonctions	Implémentation
Prometheus	Scraping des métriques (CPU, RAM, disque)	Préconfiguré dans /etc/prometheus/prometheus.yml
Grafana	Tableau de bord visuel en temps réel	Dashboard en JSON prêt à importer
Logs NGINX / SSH	Audit de sécurité et analyse des connexions	/var/log/nginx/access.log et /var/log/auth.log
Fail2Ban (option)	Blocage IP brute-force	Surveillance des tentatives SSH
Snapshots Cloud	Sauvegarde image + restauration rapide	OpenStack Horizon ou `openstack snapshot create`
Loki	Centralisation et indexation des logs textuels (temps réel, historique)	Binaire dans /opt/loki avec configuration dans /etc/loki/loki-config.yml, connecté à Grafana

Loki vs Prometheus – usage différencié :

	Prometheus	Loki
Type de données	Données chiffrées et structurées (métriques)	Données textuelles (logs bruts)
Mode d'acquisition	Pull (scrape régulier)	Push (via Promtail, envoi direct des logs)
Stockage	Séries temporelles avec labels	Streams de logs indexés par labels
Utilisation typique	Supervision technique (CPU, RAM, réseau...)	Audit, investigation, forensic, traçabilité
Lien avec Grafana	Datasource métriques (visualisation numérique)	Datasource logs (requêtes textuelles)

1.4 Indicateurs de continuité (KPI SI)

KPI	Seuil cible	Relevé actuel
Taux de disponibilité du service	≥ 98%	à calculer
Temps moyen de résolution (MTTR)	< 30 min	à noter
Alertes critiques Prometheus	0 par semaine	OK
Taux de renouvellement SSL	100% automatique	Certbot ✓
Durée de restauration snapshot	< 10 min	OK

1.5 Évolutivité et gestion des incidents

En cas d'indisponibilité : bascule via snapshot préconfiguré.

Si montée en charge (100+ utilisateurs Jitsi) : migration vers instance plus puissante via OpenStack.

Ajouts & possibilités :

- **Wazuh** ou **Suricata** pour logs réseau, **intégration Loki** pour centralisation des logs.
- Possibilité de basculer vers un PRA basé sur GCP si besoin (plan en option).

2.0 Etats des lieux :

Exemple d'analyse journalière (Journal Tercium)

```

○ test@KUS-F-STAGE MINGW64 ~/Documents/Tercium_Stage/json_dashboard_Grafana
$ ssh -i "/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key" ubuntu@37.156.46.238
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-60-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Jul 23 09:03:43 AM UTC 2025

System load:      0.09
Usage of /:        42.5% of 19.52GB
Memory usage:     11%
Swap usage:       0%
Processes:        182
Users logged in:  0
IPv4 address for enp3s0: 37.156.46.238
IPv6 address for enp3s0: 2001:1600:16:10::488

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Wed Jul 23 06:55:48 2025 from 86.192.189.89

```

Analysons cette saisie : De l'intérêt du monitoring !

Analyse capacitaire et opérationnelle détaillée de ta saisie de démarrage SSH vers l'instance tercium (Ubuntu 24.04.2 LTS) :

2.1 Informations Générales

Élément	Valeur
Distribution / Version	Ubuntu 24.04.2 LTS (kernel 6.8.0-60-generic)
Architecture	x86_64
Accès	SSH via clé privée (tercium_instance_key)
Adresse IP publique	37.156.46.238
Adresse IPv6	2001:1600:16:10::488
Interface réseau active	enp3s0

2.2 Capacités Système Actuelles

Ressource	Valeur mesurée	Observation / Interprétation
CPU load average	0.09	✓ Faible, système très peu sollicité (idle)
Utilisation disque /	42.5% of 19.52GiB	<u>Environ 8.3 Gio utilisés sur 19.52GiB : surveiller à terme</u>
Mémoire vive	11% (~2.15 GiB)	✓ Très bonne marge libre
Swap	0%	✓ Aucun swap utilisé (bonne nouvelle)
Processus actifs	182	● Normal pour un serveur standard Ubuntu + Prometheus stack

2.3 Risques / Points faibles détectés

Élément	Commentaire
Expanded Security Maintenance	✗ Non activé → Pas de correctifs long-terme post-EOL sans abonnement
2 updates can be applied	Système pas encore à jour (paquets non patchés)
2 ESM Apps updates	Certaines apps peuvent être vulnérables si pas corrigées
No users logged in	Indique que seule la session SSH actuelle est active
Last login from 86.192.189.89	À surveiller si cet accès n'est pas prévu dans la politique de sécurité

2.4 Hypothèse de charge + recommandations

Domaine	Recommandation
Disque (19.5 Go)	Prévoir un disque annexe / volume de log si tu fais du test de charge ou SIEM
Update & upgrade	<code>sudo apt update && sudo apt upgrade -y</code> avant test de monitoring
Firewall / Fail2Ban	Vérifier si ufw ou iptables actif + fail2ban installé
Surveillance CPU	Pas nécessaire pour l'instant, mais déjà intégrer dans Grafana pour tests

3.0 Intégration des outils de sécurité : Wazuh, Suricata, Fail2Ban, Loki

3.1 Préambule :

ConnexionsPowershell :

`cd : ssh -i "C:\Users\test\Documents\Tercium_Stage\ssh_keys\tercium-instance_key" -o IdentitiesOnly=yes ubuntu@37.156.46.238`

Ou

Connexion SSH en Git Bash :

`cd : ssh -i "/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key" ubuntu@37.156.46.238`

Commandes à automatiser via un Script : ok réaliser à activer

Relancer les services système via le terminal :

`cd : sudo systemctl restart prometheus
sudo systemctl restart node_exporter
sudo systemctl restart blackbox_exporter
sudo systemctl restart telegraf
sudo systemctl restart loki`

Pour vérifier qu'ils tournent :

`cd : sudo systemctl status prometheus
sudo systemctl status node_exporter
sudo systemctl status blackbox_exporter
sudo systemctl status telegraf
sudo systemctl status loki`

Bonnes pratiques :

- **Prometheus** doit être actif pour collecter les métriques.
- **Node Exporter** doit tourner sur chaque machine à surveiller.
- **Blackbox Exporter** doit être lancé une fois (souvent sur la même machine que Prometheus) pour faire les tests de ping, HTTP, TCP, etc.
- **Grafana** se connecte à **Prometheus**. Il n'a pas besoin d'être redémarré sauf en cas de changement de configuration du data source ou de plantage.
- **Loki** n'indexe rien par lui-même, il attend des logs via API. promtail agit comme **collecteur et expéditeur** de ces logs.

Astuce : automatisation au démarrage :

```
cd : sudo systemctl enable prometheus  
      sudo systemctl enable node_exporter  
      sudo systemctl enable blackbox_exporter  
      sudo systemctl enable telegraf  
      sudo systemctl enable loki
```

Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	http://localhost:9090	http://37.156.46.238:9091 (<i>si exposé</i>)
Node Exporter	http://localhost:9100	http://37.156.46.238:9100 (<i>si exposé</i>)
Blackbox Exporter	http://localhost:9115	http://37.156.46.238:9115 (<i>si exposé</i>)
Grafana	http://localhost:3000	http://37.156.46.238:3000
Loki	http://localhost:3100	http://37.156.46.238:3100 <i>(non recommandé)</i>

Notes importantes pour Loki :

- Pas de frontend intégré à <http://localhost:3100>.
- Ce port est utilisé par Grafana pour interroger Loki via /loki/api/v1/query, etc.
- L'exposition publique du port 3100 est inutile et déconseillée :
 - Pas d'authentification native
 - Risques de requêtes malveillantes sur l'API REST

3.2 Pourquoi installer la suite de sécurité, Suricata / Fail2ban / Wazuh / Loki

Exploiter les logs, alertes métriques prometheus via Grafana et/ou cron.

Méthodologie de base à compléter pour exploiter les logs, alertes et métriques Prometheus via Grafana et/ou cron, dans le contexte de tests de charge avec Apache JMeter.

CONTEXTE : Risques et faiblesses identifiés

À partir de tes documents et échanges, les risques et besoins critiques sont :

Risque / faiblesse	Source	Implication
Ouverture/fermeture inattendue de ports	réseau/SDN	coupure de service Jitsi
Désactivation du chiffrement E2EE ou anomalie Prosody	config/log	fuite de confidentialité
Comportement anormal CPU/RAM/disk	JVB, jicofo	saturation, crash
Attaque brute-force ou DDoS	logs Prosody/JVB	dénie de service, injection
Perte ou altération certificat TLS	SSL Prometheus	perte de connexion chiffrée
Lenteurs ou timeouts HTTP	alertes Blackbox	indisponibilité partielle

OUTILS DE LECTURE ET CORRÉLATION

Outil	Usage
Prometheus	Requête directe via /metrics, alertes, API
Grafana	Visualisation temps réel, corrélation, alerting
cron + Bash	Requêtes automatisées régulières (curl + grep + jq)
JMeter	Générateur de charge ciblé
Telegraf	Logs + métriques système
Blackbox Exporter	Tests de services externes (HTTP, TCP, etc.)

3.3 Installation Suricata et Fail2Ban puis Wazuh et Loki.

Voici la procédure recommandée pour installer et configurer **Fail2Ban**, **Wazuh**, et **Suricata**, **Loki**, dans le cadre d'un usage avec **Apache JMeter** pour tests de charge et de résilience : **ultérieurement nous pourrions installer Loki**.

INSTALLATION DE Suricata :

Installation propre de Suricata : [Via dépôt officiel OISF \(stable et à jour\)](#)

cd : sudo add-apt-repository ppa:oisf/suricata-stable -y
sudo apt update
sudo apt install suricata -y

- Vérification post-installation des **fichiers binaires et configuration** : **résultat attendu**

cd : which suricata
suricata --build-info
ls -l /etc/suricata/
ls -l /var/log/suricata/



L'utilisateur courant (**ubuntu**) n'a pas les droits suffisants pour lister ces répertoires.

Option 1 : Solution rapide une **relance avec les commandes avec sudo** :

cd : sudo ls -la /etc/suricata/
sudo ls -la /var/log/suricata/

```
ubuntu@jitsi-tercium:~$ sudo ls -la /etc/suricata/
sudo ls -la /var/log/suricata/
total 120
drwxr-x---  2 suricata suricata  4096 Jul 30 11:53 .
drwxr-xr-x 136 root      root    12288 Jul 30 11:53 ..
-rw-r--r--  1 root      suricata  3327 Jul  8 00:39 classification.co
-rw-r--r--  1 root      suricata 1701 Jul  8 00:39 reference.config
-rw-r--r--  1 root      suricata 92804 Jul 24 07:45 suricata.yaml
-rw-r--r--  1 root      suricata 1643 Jul  8 00:39 threshold.config
total 32
drwxrwx---  5 suricata suricata  4096 Jul 30 11:53 .
drwxrwxr-x 18 root      syslog   4096 Jul 30 11:53 ..
drwxrwxr-x  2 root      suricata  4096 Jul 24 07:45 certs
drwxrwxr-x  2 root      suricata  4096 Jul 24 07:45 core
-rw-r--r--  1 suricata suricata     0 Jul 30 11:53 eve.json
-rw-r--r--  1 suricata suricata     0 Jul 30 11:53 fast.log
drwxrwxr-x  2 root      suricata  4096 Jul 24 07:45 files
-rw-r--r--  1 suricata suricata     0 Jul 30 11:53 stats.log
-rw-r--r--  1 suricata suricata 11710 Jul 30 11:53 suricata.log
```

Et vérifie l'ownership des répertoires :

cd : sudo stat /etc/suricata
sudo stat /var/log/suricata

```
ubuntu@jitsi-tercium:~$ sudo stat /etc/suricata
sudo stat /var/log/suricata
  File: /etc/suricata
  Size: 4096          Blocks: 8          IO Block: 4096   directory
Device: 8,1      Inode: 430640        Links: 2
Access: (0750/drwxr-x---) Uid: ( 114/suricata)  Gid: ( 115/suricata)
Access: 2025-07-30 12:02:46.726892442 +0000
Modify: 2025-07-30 11:53:47.385219104 +0000
Change: 2025-07-30 11:53:47.609222291 +0000
 Birth: 2025-07-30 11:53:46.964213115 +0000
  File: /var/log/suricata
  Size: 4096          Blocks: 8          IO Block: 4096   directory
Device: 8,1      Inode: 1192393       Links: 5
Access: (0770/drwxrwx---) Uid: ( 114/suricata)  Gid: ( 115/suricata)
Access: 2025-07-30 12:02:46.741892656 +0000
Modify: 2025-07-30 11:53:48.166230216 +0000
Change: 2025-07-30 11:53:48.166230216 +0000
 Birth: 2025-07-30 11:53:47.100215049 +0000
```

Commande utile pour voir les droits détaillés :

cd : ls -ld /etc/suricata /var/log/suricata

Analyse des permissions actuelles :

Répertoire	Propriétaire (Uid)	Groupe (Gid)	Droits	Explication
/etc/suricata	suricata:suricata	0750	rwxr-x---	Pas accessible à ubuntu, seulement à root et groupe suricata
/var/log/suricata	suricata:suricata	0770	rwxrwx---	Même restriction : ubuntu est exclu de lecture ou d'écriture directe

Analyse des droits rwxrwx---

Position	Droits Unix	Explication technique	Application concrète
1–3 (Owner)	rwx	Le propriétaire (ici suricata) peut lire, écrire, exécuter	Suricata lit/écrit sa conf et ses logs
4–6 (Group)	rwx	Les utilisateurs du groupe suricata ont les mêmes droits	Ajoute ubuntu au groupe si besoin
7–9 (Other)	---	Les autres utilisateurs (ex. toi si hors groupe) n'ont aucun droit	ubuntu ne peut pas accéder sans sudo ou ajout au groupe

Interface réseau : que signifie enp3s0 ?

Élément	Explication	Commande associée
enp3s0	Nom de l'interface réseau physique active sur la machine	À remplacer selon ta machine
Pourquoi ?	Suricata doit savoir quelle interface sniffer pour capturer les paquets	
Trouver la bonne interface	ip a ou ip link	Identifier enpXsY, eth0, ens18, etc.
Tester Suricata	sudo suricata -T -c /etc/suricata/suricata.yaml -i <interface>	Ex: -i enp3s0 → à adapter

Trouver l'interface Réseau Active : annexe à solutions & commandes

```
ubuntu@jitsi-tercium:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc prio state UP group default qlen 1000
    link/ether fa:16:3e:4e:22:87 brd ff:ff:ff:ff:ff:ff
    inet 37.156.46.238/24 metric 100 brd 37.156.46.255 scope global dynamic enp3s0
        valid_lft 65176sec preferred_lft 65176sec
    inet6 2001:1600:16:10::488/128 scope global dynamic noprefixroute
        valid_lft 65179sec preferred_lft 65179sec
    inet6 fe80::f816:3eff:fe4e:2287/64 scope link
        valid_lft forever preferred_lft forever
```

Commande de test Suricata :

cd : sudo suricata -T -c /etc/suricata/suricata.yaml -i enp3s0

- **-T** = test de configuration (dry-run, pas de lancement réseau réel)
- **-c** = chemin du fichier suricata.yaml
- **-i** = interface réseau utilisée (à identifier manuellement)

Choix des options :

Option 1	Signification
-u suricata	Cible uniquement les logs du service suricata
-f	Mode "follow" temps réel (équivalent tail -f)

Option 2 : Ajouter ubuntu au groupe suricata

cd : sudo usermod -aG suricata ubuntu

Activation de suricata :

cd : sudo systemctl enable --now suricata

cd : sudo systemctl status suricata

```
ubuntu@jitsi-tercium:~$ sudo systemctl enable --now suricata
sudo systemctl status suricata
● suricata.service - Suricata IDS/IPS/NSM/FW daemon
   Loaded: loaded (/usr/lib/systemd/system/suricata.service; enabled; presen
   Active: active (running) since Wed 2025-07-30 12:35:14 UTC; 20ms ago
     Docs: man:suricata(8)
           man:suricatasc(8)
           https://suricata.io/documentation/
  Process: 62646 ExecStartPre=/bin/rm -f /run/suricata.pid (code=exited, st
 Main PID: 62648 (Suricata-Main)
    Tasks: 1 (limit: 9434)
   Memory: 1.1M (peak: 1.6M)
      CPU: 26ms
     CGroup: /system.slice/suricata.service
             └─62648 /usr/bin/suricata --af-packet -c /etc/suricata/suricata.

Jul 30 12:35:14 jitsi-tercium systemd[1]: Starting suricata.service - Suricat
Jul 30 12:35:14 jitsi-tercium systemd[1]: Started suricata.service - Suricata
Jul 30 12:35:14 jitsi-tercium suricata[62648]: i: suricata: This is Suricata
lines 1-17/17 (END)
```

Commande valide pour tester Suricata :

cd : sudo suricata -T -c /etc/suricata/suricata.yaml -i enp3s0

```
ubuntu@jitsi-tercium:~$ sudo suricata -T -c /etc/suricata/suricata.yaml -i enp3s0
i: suricata: This is Suricata version 8.0.0 RELEASE running in SYSTEM mode
W: detect: No rule files match the pattern /var/lib/suricata/rules/suricata.rules
```

Fichier de configuration est valide, mais aucune règle active de détection n'a été trouvée. :

Installer les règles communautaires :

cd : sudo suricata-update

Puis recharger le service :

cd : sudo systemctl restart suricata

Visualiser les logs de Suricata en temps réel

cd : sudo journalctl -u suricata -f

Ici exit-code donc un problème de permission sur /etc/suricata/suricata.yaml

Il est vide donc à préparer :

cd : sudo nano /etc/suricata/suricata.yaml

Étapes après sauvegarde :

1. Valider la config :

cd : sudo suricata -T -c /etc/suricata/suricata.yaml -i enp3s0

```
ubuntu@jitsi-tercium:~$ sudo suricata -T -c /etc/suricata/suricata.yaml -i enp3s0
i: suricata: This is Suricata version 8.0.0 RELEASE running in SYSTEM mode
i: npm-hs: Rule group caching - loaded: 108 newly cached: 0 total cacheable: 108
i: suricata: Configuration provided was successfully loaded. Exiting.
i: device: enp3s0: packets: 0, drops: 0 (0.00%), invalid checksum: 0
ubuntu@jitsi-tercium:~$ []
```

2. Redémarrer le service :

cd : sudo systemctl restart suricata

3. Vérifier le log en temps réel :

cd : sudo tail -f /var/log/suricata/fast.log

Rien : logique il manque des fichiers de configuration.

Installation de trois fichiers dans /etc/suricata/

Voici les trois fichiers essentiels à ajouter dans /etc/suricata/ pour que Suricata fonctionne correctement avec des règles personnalisées :

1. /etc/suricata/rules/local.rules

Rôle : Contient les règles de détection personnalisées (ex : HTTP, ICMP, DNS).

Exemple de règle de test ICMP (ping) :

```
alert icmp any any -> any any (msg:"ICMP test rule"; sid:1000001; rev:1;)
```

Cette règle déclenche une alerte sur tout ping ICMP.

2. /etc/suricata/classification.config

Rôle : Définit les catégories d'alertes (utilisées par priority dans les règles).

Contenu minimal fonctionnel :

```
config classification: not-suspicious,Not Suspicious Traffic,3
```

```
config classification: attempted-recon,Attempted Information Leak,2
```

```
config classification: successful-admin,Successful Administrator Privilege Gain,1
```

Ce fichier est référencé dans **suricata.yaml** dans la section **classification-file**.

3. /etc/suricata/reference.config

Rôle : Fournit des liens vers des bases de vulnérabilités (optionnel mais requis par certaines règles).

Contenu minimal :

```
config reference: cve http://cve.mitre.org/cgi-bin/cvename.cgi?name=
```

```
config reference: url http://
```

```
config reference: bugtraq http://www.securityfocus.com/bid/
```

```
config reference: nessus http://www.nessus.org/plugins/index.php?view=single&id=
```

Ce fichier est également référencé dans **suricata.yaml**.

Vérification :

S'assurer que dans ton fichier /etc/suricata/suricata.yaml :

default-rule-path: /etc/suricata/rules

rule-files: local.rules

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config

INSTALLATION DE Fail2ban :

Commandes à exécuter dans l'ordre

1. Fail2Ban

```
cd : sudo apt update && sudo apt install -y fail2ban
      sudo systemctl enable fail2ban --now
      sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Ensuite :

- **Fichier jail.d/ssh.conf minimal**
- **Fichier filter.d/nginx-login.conf si besoin**
- **fail2ban-client status pour vérifier**

1. Installation minimale de Fail2ban :

```
cd : sudo apt update
      sudo apt install fail2ban -y
```

Vérification du statut :

```
cd : sudo systemctl status fail2ban
```

2. Configuration : Protection du SSH

Créer une configuration personnalisée :

```
cd : sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Usages recommandés de ces commandes :

Commande	Fonction	Utilisation recommandée
sudo apt update	Met à jour la liste des paquets disponibles	Toujours exécuter avant une installation de paquets
sudo apt install fail2ban -y	Installe Fail2Ban sans confirmation	Installation automatique, utilisée dans les scripts
sudo systemctl enable fail2ban	Active Fail2Ban au prochain redémarrage du système	À faire si tu veux que Fail2Ban démarre à chaque boot
sudo systemctl start fail2ban	Démarre Fail2Ban immédiatement	À faire si tu veux tester ou utiliser Fail2Ban sans redémarrer
sudo systemctl enable fail2ban --now	Active au boot et démarre immédiatement	Recommandé dans la plupart des cas pour automatiser
sudo systemctl status fail2ban	Affiche l'état actuel du service (actif/inactif, logs...)	Vérification après installation ou redémarrage

Modifier uniquement la section [sshd] dans /etc/fail2ban/jail.local :

```
ini

[sshd]
enabled = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 3
bantime = 600
```

Redémarrer et tester :

cd : sudo systemctl restart fail2ban
sudo fail2ban-client status sshd

```
ubuntu@jitsi-tercium:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| | - Currently failed: 3
| | - Total failed:    18
| | - File list:        /var/log/auth.log
`- Actions
  | - Currently banned: 3
  | - Total banned:      4
  ` - Banned IP list:   210.79.191.194 99.137.178.25 80.94.95.115
```

Récapitulatif de l'état de Fail2Ban

Paramètre	Valeur actuelle
Tentatives en échec	18
Actuellement en échec	3 (en cours de dépassement de seuil)
IP bannies actuellement	3 (→ protection active)
IP bannies au total	4 (1 a été libérée automatiquement ?)
Fichier surveillé	/var/log/auth.log
Jail active	[sshd]

Les IP bannies : **210.79.191.194**
99.137.178.25
80.94.95.115

Étapes suivantes proposées

1. Valider le Makefile de gestion Fail2Ban → Peut être déposer dans /usr/local/bin/fail2ban.mk ou le versionner dans un dépôt Git.
2. Surveiller la journalisation :
cd : sudo tail -f /var/log/fail2ban.log

```
ubuntu@jitsi-tercium:~$ sudo tail -f /var/log/fail2ban.log
2025-07-31 07:42:18,348 fail2ban.actions      [16007]: NOTICE  [sshd] Ban 99.137.178.25
2025-07-31 07:44:09,201 fail2ban.filter       [16007]: INFO    [sshd] Found 80.94.95.116 - 202
5-07-31 07:44:09
2025-07-31 07:44:11,500 fail2ban.filter       [16007]: INFO    [sshd] Found 80.94.95.116 - 202
5-07-31 07:44:11
2025-07-31 07:44:11,594 fail2ban.actions      [16007]: NOTICE  [sshd] Ban 80.94.95.116
2025-07-31 07:45:21,628 fail2ban.actions      [16007]: NOTICE  [sshd] Unban 210.79.191.194
2025-07-31 07:45:29,365 fail2ban.filter       [16007]: INFO    [sshd] Found 210.79.191.194 - 2
025-07-31 07:45:29
2025-07-31 07:45:31,316 fail2ban.filter       [16007]: INFO    [sshd] Found 210.79.191.194 - 2
025-07-31 07:45:31
2025-07-31 07:45:31,653 fail2ban.actions      [16007]: NOTICE  [sshd] Ban 210.79.191.194
2025-07-31 07:46:21,678 fail2ban.actions      [16007]: NOTICE  [sshd] Unban 79.117.1.106
2025-07-31 07:52:17,012 fail2ban.actions      [16007]: NOTICE  [sshd] Unban 99.137.178.25
2025-07-31 07:54:11,066 fail2ban.actions      [16007]: NOTICE  [sshd] Unban 80.94.95.116
^C
```

INSTALLATION DE Wazuh :

Installation de Wazuh (Agent Only)

Wazuh est un agent SIEM (Security Information & Event Management) très puissant.

Installation rapide de l'agent : Wazuh (Ubuntu 22.04 / 20.04) :

1. Import de la clé GPG officielle :

```
cd : curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo gpg --dearmor -o /etc/apt/keyrings/wazuh.gpg
```

```
ubuntu@jitsi-tercium:~$ curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo gpg --dearmor -o /etc/apt/keyrings/wazuh.gpg
```

2. Ajout du dépôt APT :

```
cd : echo "deb [signed-by=/etc/apt/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/stable main" | sudo tee /etc/apt/sources.list.d/wazuh.list
```

```
ubuntu@jitsi-tercium:~$ echo "deb [signed-by=/etc/apt/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/stable main" | sudo tee /etc/apt/sources.list.d/wazuh.list
deb [signed-by=/etc/apt/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/stable main
```

3. Mise à jour + installation :

```
cd : sudo apt update
```

```
sudo apt install wazuh-agent -y
```

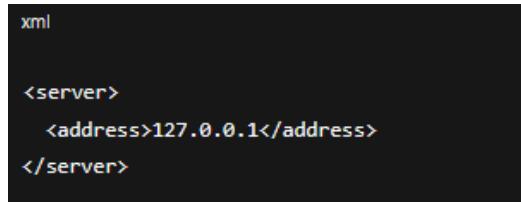
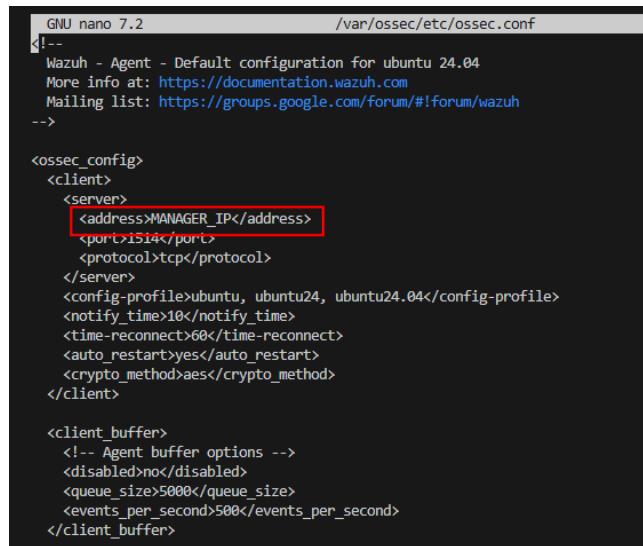
```
ubuntu@jitsi-tercium:~$ sudo apt update
sudo apt install wazuh-agent -y
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:3 https://repos.influxdata.com/debian stable InRelease
Hit:4 https://apt.grafana.com stable InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb InRelease
Hit:6 http://az-1.clouds.archive.ubuntu.com/ubuntu noble InRelease
Hit:7 http://az-1.clouds.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:8 http://az-1.clouds.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:9 https://ppa.launchpadcontent.net/oisf/suricata-stable/ubuntu noble InRelease
Hit:10 https://download.jitsi.org stable/ InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
125 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: https://pkgs.k8s.io/core:/stable:/v1.30/deb/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wazuh-agent is already the newest version (4.12.0-1).
0 upgraded, 0 newly installed, 0 to remove and 125 not upgraded.
```

4. Configuration minimale :

Ouvre le fichier :

cd : sudo nano /var/ossec/etc/ossec.conf

Modifie la section <server> pour y ajouter l'IP de ton manager futur (peut être temporairement un placeholder) :



```
GNU nano 7.2                               /var/ossec/etc/ossec.conf
<!--
Wazuh - Agent - Default configuration for ubuntu 24.04
More info at: https://documentation.wazuh.com
Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <client>
    <server>
      <address>MANAGER_IP</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>ubuntu, ubuntu24, ubuntu24.04</config-profile>
    <notify_time>10</notify_time>
    <time-reconnect>0</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>

  <client_buffer>
    <!-- Agent buffer options -->
    <disabled>no</disabled>
    <queue_size>5000</queue_size>
    <events_per_second>500</events_per_second>
  </client_buffer>
</ossec_config>
```

```
<!--
Wazuh - Agent - Default configuration for ubuntu 24.04
More info at: https://documentation.wazuh.com
Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <client>
    <server>
      <address>127.0.0.1</address>
    </server>
  </client>

  <client_buffer>
    <!-- Agent buffer options -->
    <disabled>no</disabled>
    <queue_size>5000</queue_size>
    <events_per_second>500</events_per_second>
  </client_buffer>
</ossec_config>
```

L'**IP du manager Wazuh** correspond à l'adresse du serveur central qui recevra les alertes et journaux envoyés par l'**agent Wazuh**. Elle dépend donc de ton architecture de supervision.

Comment déterminer l'IP du manager ?

Cas 1 : Tu installles un manager Wazuh plus tard sur un serveur existant

Utilise l'**IP privée ou publique** de ce serveur.

- **ip a # pour IP locale.**
- **hostname -I # IP de l'interface principale.**

Cas 2 : Le manager sera dans un cluster Wazuh + ELK

- C'est l'**IP de la VM ou du conteneur (Docker, LXC, KVM, etc.)** exécutant le service **wazuh-manager**.

Cas 3 : Manager sur la même machine que l'agent (cas de test)

- L'adresse est **127.0.0.1**.

Cas 4 : Manager hébergé ailleurs (infogéré, cloud)

- C'est l'**IP publique du serveur dans le cloud ou l'adresse DNS** (ex : **manager.mondomaine.tld**).

Attention

Si tu as un pare-feu actif :

- Le port par défaut est 1514/tcp (ou udp) pour la communication agent ↔ manager.
- Autorise ce port sur l'interface réseau du manager.

cd : sudo ufw allow 1514/tcp

5. Activation du service :

```
cd : sudo systemctl daemon-reexec  
      sudo systemctl enable wazuh-agent --now  
      sudo systemctl status wazuh-agent
```

```
ubuntu@jitsi-tercium:~$ sudo systemctl daemon-reexec  
sudo systemctl enable wazuh-agent --now  
sudo systemctl status wazuh-agent  
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-agent.service → /usr/lib/systemd/system/wazuh-agent.service.  
● wazuh-agent.service - Wazuh agent  
   Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; preset: enabled)  
   Active: active (running) since Thu 2025-07-31 09:59:19 UTC; 22ms ago  
     Process: 57326 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, stat  
       Tasks: 31 (limit: 9434)  
      Memory: 45.8M (peak: 50.7M)  
        CPU: 2.128s  
      CGroup: /system.slice/wazuh-agent.service  
          └─57365 /var/ossec/bin/wazuh-execd  
              ├─57373 /var/ossec/bin/wazuh-agentd  
              ├─57394 /var/ossec/bin/wazuh-syscheckd  
              ├─57407 /var/ossec/bin/wazuh-logcollector  
              └─57424 /var/ossec/bin/wazuh-modulesd  
  
Jul 31 09:59:12 jitsi-tercium systemd[1]: Starting wazuh-agent.service - Wazuh agent...  
Jul 31 09:59:12 jitsi-tercium env[57326]: Starting Wazuh v4.12.0...  
Jul 31 09:59:12 jitsi-tercium env[57326]: Started wazuh-execd...  
Jul 31 09:59:13 jitsi-tercium env[57326]: Started wazuh-agentd...  
Jul 31 09:59:14 jitsi-tercium env[57326]: Started wazuh-syscheckd...  
Jul 31 09:59:16 jitsi-tercium env[57326]: Started wazuh-logcollector...  
Jul 31 09:59:17 jitsi-tercium env[57326]: Started wazuh-modulesd...  
Jul 31 09:59:19 jitsi-tercium env[57326]: Completed.  
Jul 31 09:59:19 jitsi-tercium systemd[1]: Started wazuh-agent.service - Wazuh agent.  
lines 1-23/23 (END)
```

INSTALLATION DE Loki :

ÉTAPES RÉUSSIES

1. Téléchargement et décompression

wget des binaires Loki et Promtail depuis GitHub :
✓ exécuté avec succès dans /opt
unzip + chmod +x appliqués aux deux binaires.

Renommage :

```
sudo mv /opt/loki-linux-amd64 /opt/loki  
sudo mv /opt/promtail-linux-amd64 /opt/promtail
```

2. Vérifier l'existence et l'emplacement du binaire Loki :

```
cd : ls -lh /opt/loki-bin
```

```
ubuntu@jitsi-tercium:~$ ls -lh /opt/loki-bin  
-rwxr-xr-x 1 root root 64M Sep  7  2023 /opt/loki-bin
```

Le fichier doit être exécutable (-rwxr-xr-x), sinon, corrige les droits avec :

```
cd : sudo chmod +x /opt/loki-bin ok
```

3. Vérifier que le fichier de configuration est bien en place :

cd : cat /etc/loki/loki-config.yml ok

4. Crée le fichier loki.service dans systemd :

cd : sudo nano /etc/systemd/system/loki.service

```
loki.service.ini
1 [Unit]
2 Description=Grafana Loki Service
3 After=network.target
4
5 [Service]
6 ExecStart=/opt/loki-bin -config.file=/etc/loki/loki-config.yml
7 Restart=on-failure
8 User=root
9
10 [Install]
11 WantedBy=multi-user.target
12 |
```

5. Recharge et active le service :

cd : sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl enable loki.service

```
ubuntu@jitsi-tercius:~$ sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable loki.service
Created symlink /etc/systemd/system/multi-user.target.wants/loki.service → /etc/systemd/s
ystem/loki.service.
```

6. Lance Loki via systemd :

cd : sudo systemctl start loki.service

7. Verifier son état:

cd : sudo systemctl status loki.service --no-pager

```
ubuntu@jitsi-tercius:~$ sudo systemctl status loki.service --no-pager
● loki.service - Grafana Loki Service
   Loaded: loaded (/etc/systemd/system/loki.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-08-04 10:10:51 UTC; 1min 10s ago
     Main PID: 38976 (loki-bin)
        Tasks: 9 (limit: 9434)
       Memory: 77.3M (peak: 77.5M)
          CPU: 534ms
        CGroup: /system.slice/loki.service
                  └─38976 /opt/loki-bin -config.file=/etc/loki/loki-config.yml

Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.542...ing"
Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.542...ing"
Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.644...ing"
Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.644...uler
Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.644...tend
Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.645...rier
Aug 04 10:10:52 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:52.646...ted"
Aug 04 10:10:55 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:55.645...ts."
Aug 04 10:10:55 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:10:55.646...0095
Aug 04 10:11:02 jitsi-tercius loki-bin[38976]: level=info ts=2025-08-04T10:11:02.645...0095
Hint: Some lines were ellipsized, use -l to show in full.
```

Organisation des fichiers

Loki et Promtail placés directement dans /opt/ (pas dans /usr/local/bin)

Fichiers de configuration créés à part :

- /etc/loki/loki-config.yml
- /etc/promtail/promtail-config.yml

Fichier de configuration Loki

Configuration complète en standalone, sans auth, ports 3100 / 9095
Stockage dans /opt/loki/index et /opt/loki/chunks
Retention configurée à 168h (7 jours)

3.4 Ce qui a fonctionné

- **Téléchargement et extraction**
Correctement téléchargé le binaire Loki ([loki-linux-amd64.zip](#)), extrait, et renommé en [/opt/loki-bin](#).
- **Structure des dossiers**
Création du dossier [/opt/loki](#) avec sous-dossiers [chunks](#) et [index](#) pour le stockage des données Loki.
- **Permissions**
Ajustement des droits pour que l'utilisateur puisse écrire dans [/opt/loki](#) et ses sous-dossiers.
- **Fichier de configuration**
Mise en place d'un fichier [loki-config.yml](#) minimaliste adapté à une installation mono-instance, avec :
 - Serveur écoutant sur le port 3100 (HTTP)
 - [ring](#) configuré en mode [inmemory](#) pour éviter les erreurs liées au cluster
 - Stockage local en [boltdb](#) et [filesystem](#) dans [/opt/loki/index](#) et [/opt/loki/chunks](#)
- **Lancement de Loki**
Lancement réussi via [/opt/loki-bin -config.file=/etc/loki/loki-config.yml](#)
Vérification avec [ss -tulpn | grep 3100](#) qui confirme l'écoute du service.
- **Mode background / daemon**
Utilisation de la commande [nohup](#) pour lancer Loki en tâche de fond, permettant de libérer le terminal et maintenir Loki actif même en fermeture du terminal.

Erreurs rencontrées et problèmes résolus

- **Erreur `command not found`**
Initialement, tu tentais d'exécuter `/opt/loki` ou `/opt/loki-linux-amd64` qui n'existaient pas ou n'étaient pas nommés ainsi. La bonne cible est `/opt/loki-bin`.
- **Erreur `ring lifecycler is shutting down`**
Cette erreur vient d'une mauvaise configuration du module `ring` utilisé pour la gestion du cluster. Elle a été corrigée en configurant le `ring` en mode `inmemory` pour une instance unique.
- **Erreur de permission et de dossier**
Loki exige des dossiers spécifiques pour stocker ses données (`chunks`, `index`). La création et la bonne attribution de leurs droits ont permis à Loki de démarrer sans erreur de stockage.
- **Erreur de connexion sur localhost port 3100**
Loki ne démarrait pas correctement, rien n'écoutes sur ce port. Après correction des points précédents, Loki écoute bien sur le port.

- **Usage des commandes sudo et chemins absolus**

Nécessité de toujours préciser le chemin complet et utiliser `sudo` pour avoir les droits d'exécution, en particulier avec les binaires situés dans `/opt`.

Méthode recommandée pour l'installation et le lancement

1. Télécharger et extraire le binaire dans `/opt/`

2. Renommer le binaire en `loki-bin` (ou un nom simple)

3. Créer les dossiers de stockage et donner les droits corrects

4. Préparer un fichier de configuration `loki-config.yml` adapté à l'architecture (mono-instance ou cluster)

5. Lancer Loki avec la commande complète :

```
sudo /opt/loki-bin -config.file=/etc/loki/loki-config.yml
```

6. Pour garder Loki en service, lancer en arrière-plan avec `nohup` :

```
sudo nohup /opt/loki-bin -config.file=/etc/loki/loki-config.yml > /var/log/loki.log 2>&1 &
```

7. Vérifier l'écoute sur le port 3100 : `sudo ss -tulpn | grep 3100`

8. Consulter les logs pour tout problème : `tail -f /var/log/loki.log`

3.5 Couplage avec les outils de monitoring : Liaison Wazuh–Prometheus–Grafana–Loki

La Maintenance du SI :

A. Phase d'exploitation : Monitoring en Production

Je suis désormais dans la **phase active de supervision continue**. Cela implique que :

- Je collecte des métriques en temps réel.
- Que les alertes Prometheus sont déclenchées (cf. `probe_success == 0`).
- L'on peut exploiter dans Grafana (dashboards, alertes visuelles)

B. Surveillance des points critiques et comment les exploiter :

Composant	Surveillance	Utilité	Risques détectés
<code>blackbox_exporter_http</code>	<code>Accès HTTP externe (Google, GitHub)</code>	Vérifie la connectivité internet et les services Web exposés	Coupure d'accès internet, filtrage DNS, timeout
<code>blackbox_exporter_icmp</code>	<code>Ping vers 8.8.8.8 / 1.1.1.1</code>	Test réseau basique	Perte de connectivité externe
<code>blackbox_exporter_ssh</code>	<code>Réponse SSH sur port 22</code>	Présence de bannière SSH (sécurité)	SSH KO ou binaire compromis
<code>blackbox_exporter_tcp</code>	<code>Connexion brute (SMTP, port 9091)</code>	Validation des services TCP	Coupure de services
<code>node_exporter</code>	<code>CPU, mémoire, disque du serveur</code>	Supervision système Linux	Charge anormale, espace disque faible
<code>telegraf</code>	<code>Moteur de collecte personnalisée</code>	Surveillance applicative ou conteneurs	Dérives d'usage métier

C. Maintenance proactive

- Surveillance continue via Grafana (tableaux à préparer avec probe_success, probe_duration_seconds, etc.).
- Corrélation via Prometheus AlertManager (même si ici 9093 est désactivé, à prévoir).
- Export journalier des données sensibles.
- Archivage et rotation des logs Prometheus (via retention ou règles de purge).

D. Automatiser

1. Dashboard Grafana dédié à la DNS/TCP/SSH/HTTP
2. Alerte mail ou webhook dès probe_success == 0
3. Backup des fichiers de configuration (prometheus.yml, alert.rules.yml)
4. Commandes cron avec curl + jq pour audit journalier :

Prometheus : Tableau d'alert :

The screenshot shows the Prometheus Alerts interface with the following details:

- Top Filter:** Shows 7 Inactive, 0 Pending, and 2 Firing alerts.
- Search Bar:** Filter by name or labels.
- Alert Groups:**
 - /etc/prometheus/alert.rules.yml > blackbox_alerts:** Contains:
 - > HTTP Target Down (1 active)
 - > ICMP Target Unreachable (2 active)
 - > TCP Connection Failed (0 active)
 - > SSH Banner Missing (0 active)
 - > DNS Resolution Failed (2 active)
 - > HTTP Response Slow (0 active)
 - /etc/prometheus/alert.rules.yml > node_exporter_alerts:** Contains:
 - > High CPU Usage (0 active)
 - > High Memory Usage (0 active)
 - > Disk Space Low (0 active)
 - /etc/prometheus/alert.rules.yml > telegraf_alerts:** Contains:
 - > Telegraf Target Down (0 active)
- Bottom Buttons:** Show annotations, Active (7), Firing (2).

DÉCRYPTAGE DE L'INTERFACE

Fichiers de règles chargés : **J'ai défini deux groupes de règles d'alerte :**

- /etc/prometheus/alert.rules.yml > **blackbox_alerts**
- /etc/prometheus/alert.rules.yml > **node_exporter_alerts**

● ● ● État des alertes :

- ● **Inactive (7)** : conditions non remplies → **pas de problème détecté**.
- ● **Firing (2)** : conditions actuellement vraies → **des alertes sont en cours**.
- ● **Pending (0)** : conditions presque remplies, en phase d'observation.

LISTE DES ALERTES : 🔴 firing (problèmes actifs)

Nom de l'alerte	Signification
HTTP_Target_Down	Un ou plusieurs endpoints HTTP ne répondent pas au Blackbox Exporter.
ICMP_Down	Un ou plusieurs hôtes ne répondent pas au ping (icmp) via Blackbox.

🟢 inactive (tout va bien pour celles-ci)

Nom	Signification
TCP_Port_Down	Tous les ports TCP surveillés sont actuellement disponibles.
HTTP_Response_Slow	Aucun temps de réponse abnormal.
High_CPU_Usage	Utilisation CPU normale (< seuil).
Low_Available_Memory	Pas de problème mémoire détecté.
Disk_Space_Low	Pas d'espace disque critique.
SSHBannerMissing	La bannière SSH est visible ou non requise.
DNS_Resolution_Failure	Résolutions DNS réussies.

Tableau des règles : Rules

node_exporter_alerts		Interval: 15.0s	2.202s ago	0.837ms	
Rule		State	Error	Last Evaluation	Evaluation Time
alert: High_CPU_Usage expr: (rate(node_cpu_seconds_total[mode="system"])[5m]) + rate(node_cpu_seconds_total[mode="user"])[5m]) > 0.9 for: 2m labels: severity: critical annotations: description: CPU > 90% sur 5 minutes (system + user). summary: Utilisation CPU élevée sur {{ \$labels.instance }}	OK			2.202s ago	0.575ms
alert: High_Memory_Usage expr: (node_memory_Active_bytes / node_memory_MemTotal_bytes) > 0.9 for: 2m labels: severity: warning annotations: description: RAM active > 90% de la capacité totale. summary: Utilisation mémoire élevée sur {{ \$labels.instance }}	OK			2.202s ago	0.101ms
alert: Disk_Space_Low expr: (node_filesystem_avail_bytes[mountpoint="/"] / node_filesystem_size_bytes[mountpoint="/"])) < 0.15 for: 5m labels: severity: warning annotations: description: Moins de 15% d'espace disque disponible sur le système racine. summary: Espace disque faible sur {{ \$labels.instance }}	OK			2.201s ago	0.092ms

telegraf_alerts		Interval: 15.0s	5.658s ago	0.386ms	
Rule		State	Error	Last Evaluation	Evaluation Time
alert: Telegraf_Target_Down expr: up[job="telegraf"] == 0 for: 1m labels: severity: critical annotations: description: Le collecteur Telegraf est injoignable ou en échec. summary: Telegraf ne répond pas sur {{ \$labels.instance }}	OK			5.658s ago	0.352ms

Prometheus Alerts Graph Status Help

Rules

blackbox_alerts	Interval: 15.0s	2.99s ago	1.393ms	
Rule	State	Error	Last Evaluation	Evaluation Time
<pre>alert: HTTP_Target_Down expr: probe_success[job="blackbox_http"] == 0 for: 30s labels: severity: critical annotations: description: La cible HTTP ([[\$labels.instance]]) est injoignable via Blackbox. summary: Cible HTTP indisponible ([[\$labels.instance]])</pre>	OK		2.101s ago	0.608ms
<pre>alert: ICMP_Target_Unreachable expr: probe_success[job="blackbox_exporter_icmp"] == 0 for: 30s labels: severity: warning annotations: description: Impossible de joindre ([[\$labels.instance]]) par ICMP. summary: Ping ICMP échoué pour ([[\$labels.instance]])</pre>	OK		2.101s ago	0.333ms
<pre>alert: TCP_Connection_Failed expr: probe_success[job="blackbox_exporter_tcp"] == 0 for: 30s labels: severity: warning annotations: description: Le port TCP de ([[\$labels.instance]]) ne répond pas. summary: Port TCP inaccessible ([[\$labels.instance]])</pre>	OK		2.101s ago	0.079ms
<pre>alert: SSH_Banner_Missing expr: probe_success[job="blackbox_exporter_ssh"] == 0 for: 30s labels: severity: warning annotations: description: La connexion SSH n'affiche pas de bannière sur ([[\$labels.instance]]).</pre>	OK		2.101s ago	0.040ms
<pre>alert: DNS_Resolution_Failed expr: probe_success[job="blackbox_exporter_dns"] == 0 for: 30s labels: severity: critical annotations: description: La résolution DNS a échoué pour ([[\$labels.instance]]).</pre>	OK		2.101s ago	0.213ms
<pre>alert: HTTP_Response_Slow expr: probe_duration_seconds[job="blackbox_http"] > 5 for: 1m labels: severity: warning annotations: description: Latence HTTP supérieure à 5 secondes pour ([[\$labels.instance]]).</pre>	OK		2.101s ago	0.073ms

telegraf_alerts	Interval: 15.0s	5.658s ago	0.386ms	
Rule	State	Error	Last Evaluation	Evaluation Time
<pre>alert: Telegraf_Target_Down expr: up[job="telegraf"] == 0 for: 1m labels: severity: critical annotations: description: Le collecteur Telegraf est injoignable ou en échec. summary: Telegraf ne répond pas sur ([[\$labels.instance]])</pre>	OK		5.658s ago	0.352ms

Ce que ça montre :

- **Prometheus** lit bien les fichiers de règles.
- Le moteur d'alerte est actif et fonctionnel.
- Il détecte déjà des anomalies en **HTTP** et **ICMP**, preuve que :
 - **Blackbox Exporter** est bien opérationnel,
 - Les cibles sont hors ligne ou inaccessibles.

Les logs surveiller par Prometheus :

Logs À Surveiller (Prometheus Et Démarrage)				
	Source	Type	Critère	Détail
1	Prometheus Targets	Statut Exporters	up{}	Vérifie si les cibles sont accessibles
2	Prometheus Metrics	Charge système	node_load1 / node_cpu_second_s_total	Charge CPU instantanée
3	Prometheus Metrics	Mémoire	node_memory_Me mAvailable_bytes	Mémoire disponible
4	Prometheus Metrics	Disque	node_filesystem_avail_bytes	Espace disque disponible
5	Prometheus Metrics	État réseau	node_network_receive_bytes_total	Activité réseau entrante
6	Prometheus Metrics	Temps de réponse exporters	probe_duration_seconds	Blackbox: temps de réponse

Les logs surveiller par Loki :

Source	Type	Critère	Détail
Loki (via Promtail)	Logs système	<code>journalctl → erreurs systemd, ssh, etc.</code>	Analyse des erreurs critiques ou redémarrages anormaux
Loki (via Promtail)	Logs NGINX	<code>/var/log/nginx/error.log ou access</code>	Détection d'erreurs HTTP (403, 500...), pics de requêtes
Loki (via Promtail)	Logs Fail2Ban	<code>/var/log/fail2ban.log</code>	IP bannies, tentatives SSH échouées
Loki (via Promtail)	Logs Suricata	<code>/var/log/suricata/eve.json</code>	Signatures d'attaques détectées, alertes réseau
Loki (via Promtail)	Logs Wazuh	<code>/var/ossec/logs/ossec.log</code>	Détection comportementale, exfiltration, escalade de privilège
Loki (via API)	Statut service Loki	<code>msg="Loki started" dans stdout</code>	Vérifie que Loki est lancé sans erreur

Lancement système :

7	Démarrage système	Charge système	System load (uptime)	Basé sur la ligne "System load:"
8	Démarrage système	Disque utilisé	Usage of /	Pourcentage d'utilisation du disque racine
9	Démarrage système	Mémoire utilisée	Memory usage	Pourcentage de RAM utilisée
10	Démarrage système	Nombre de process	Processes	Processus en cours
11	Démarrage système	Utilisateurs connectés	Users logged in	Sessions utilisateur actives
12	Démarrage système	Interface IP	IPv4/IPv6 address	Adresse réseau détectée

Status du réseau avec le suivi des logs :

blackbox_exporter_icmp (2/2 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe [module="icmp"] [target="8.8.8.8"]	UP	instance="8.8.8.8" job="blackbox-exporter_icmp"	8.839s ago	1.516ms	
http://127.0.0.1:9115/probe [module="icmp"] [target="1.1.1.1"]	UP	instance="1.1.1.1" job="blackbox-exporter_icmp"	12.224s ago	1.426ms	

blackbox_exporter_ssh (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe [module="ssh_banner"] [target="localhost:22"]	UP	instance="localhost:22" job="blackbox-exporter_ssh"	3.450s ago	11.793ms	

blackbox_exporter_tcp (2/2 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe [module="tcp_connect"] [target="localhost:22"]	UP	instance="localhost:22" job="blackbox-exporter_tcp"	150.000ms ago	3.103ms	
http://127.0.0.1:9115/probe [module="tcp_connect"] [target="localhost:443"]	UP	instance="localhost:443" job="blackbox-exporter_tcp"	13.827s ago	2.440ms	

blackbox_http (2/2 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe [module="http_2xx"] [target="http://localhost:80"]	UP	instance="http://localhost:80" job="blackbox_http"	4.76s ago	2.265ms	
http://127.0.0.1:9115/probe [module="http_2xx"] [target="https://example.com"]	UP	instance="https://example.com" job="blackbox_http"	12.165s ago	469.599ms	

telegraf (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9273/metrics	UP	instance="localhost:9273" job="telegraf"	10m 13s ago	3.277ms	

node_exporter_distant (1/1 up)						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://37.156.46.238:9100/metrics	UP	instance="node distant"; job="node_exporter_distant"	5.81s ago	11.306ms		
prometheus_distant (1/1 up)						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://37.156.46.238:9091/metrics	UP	instance="distant_prometheus"; job="prometheus_distant"	1.114s ago	6.250ms		
prometheus_local (1/1 up)						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://localhost:9091/metrics	UP	instance="local_prometheus"; job="prometheus_local"	355.000ms ago	6.271ms		

Résumé opérationnel

Élément	Source Prometheus	But	Syntaxe
Targets	API /api/v1/targets	Visualiser toutes les cibles surveillées	curl
alert.rules.yml /etc/prometheus/alert.rules.yml		Déclenche les alertes visuelles et emails	alert, expr, labels, annotations
Modules Blackbox	blackbox.yml	Définit les types de tests à réaliser	http_2xx, icmp, tcp_connect...
"All scrape pools"	Grafana/Prometheus UI	Affiche les job_name actifs	basé sur scrape_configs

4.0 Paramétrage du monitoring et des outils de sécurité

L'ensemble du système d'information a été configuré pour permettre une **surveillance temps réel, une corrélation des événements et des contre-mesures automatisées**. Un **test de charge** a été mis en œuvre pour valider le bon fonctionnement des outils déployés.

Intégration avec Apache JMeter (scénario de test)

Composant	Rôle dans le test
Apache JMeter	Génère une charge réseau contrôlée sur les endpoints Jitsi/Nginx (ex. : requêtes HTTP en boucle, connexion SSH, flood TCP).
Fail2Ban	Observe les tentatives échouées (ex : SSH) et déclenche un bannissement si seuil dépassé (maxretry).
Suricata	Analyse le trafic réseau (mode IDS) et détecte les attaques ou scans (grâce à eve.json).
Wazuh Agent	Centralise les logs d'activité, de sécurité et d'intégrité ; reçoit les logs de Suricata et Fail2Ban via les blocs <localfile>.
Prometheus / Grafana	Supervise les métriques système (CPU, RAM, réseau), expose les valeurs en temps réel.
Loki + Promtail	Collecte les journaux applicatifs (NGINX, Suricata, Fail2Ban), indexe et rend consultables via Grafana.

Objectif du test :

- **Valider le déclenchement automatique des défenses** face à des scénarios d'attaque simulés (ex : brute force SSH, surcharge HTTP).
- **Observer les métriques système** sous charge pour évaluer la résilience.
- **Analyser les journaux en temps réel** avec Loki, Fail2Ban et Suricata.
- **Corréler les alertes Wazuh** issues de plusieurs sources.

Résultat attendu

- Augmentation du nombre de connexions détectée → hausse du trafic sur Grafana.
- Alertes Fail2Ban visibles dans /var/log/fail2ban.log → bannissement d'IP malveillantes.
- Entrées Suricata dans eve.json → signature d'attaque détectée.
- Logs synchronisés vers Wazuh → accessible pour corrélation et forensic.
- Tableau de bord Grafana (via Loki) → consultation des logs unifiés.

4.1. Script & Test

Étape 1 : Script .jmx (JMeter Test Plan) : Je crée un fichier XML nommé **test_jitsi.jmx** :

Étape 2 : Lancement automatique du test : Ajouter un script bash : **test_jitsi.sh**

Résultat attendu

- **results.jtl** contiendra les temps de réponse, taux d'erreur.
- **fail2ban.log** enregistrera les bannissements éventuels si trop de requêtes émanent d'une même IP.
- **eve.json (Suricata)** signalera les anomalies réseau.
- **ossec.log (Wazuh)** confirmera la réception des alertes centralisées.

1. Organisation des fichiers :

```
/home/user/
├── apache-jmeter/          # Dossier d'installation de JMeter
│   ├── tests/               # Scénario de test JMeter
│   │   └── test_jitsi.jmx
│   └── results.jtl           # Résultat des tests
└── scripts/                # Script de test Bash
    └── test_jmeter.sh        # Automatisation si souhaité
    └── Makefile              # Automatisation si souhaité
```

2. Détails des composants

a. test_jitsi.jmx

- **Chemin :** /home/user/tests/test_jitsi.jmx
- **But :** charge sur <https://jitsi-tercium.example.com/>

b. test_jmeter.sh

- **Chemin :** /home/user/scripts/test_jmeter.sh
- **Droits :**
cd : chmod +x /home/user/scripts/test_jmeter.sh

c. Makefile (optionnel si automatisation)

- **Chemin : /home/user/Makefile**
- **Usage :**
 - cd : make** # lance test complet
 - cd : make logs** # lance la surveillance seule

4.2 Résumé des activations

Élément	Action
.jmx	Test HTTP défini à l'avance
.sh	Lancement scripté + supervision
Makefile	Automatisation simple via make
chmod +x	Rends le script exécutables
systemctl	Redémarre les services de sécurité
tail -f	Surveillance live des logs

Script de test (test_jmeter.sh) : Ce script orchestre l'automatisation de la séquence complète

Étape	Action
Redémarrage	Relance propre des services (Suricata, Fail2Ban, Wazuh)
Exécution	Lance le test JMeter (.jmx) en ligne de commande
Supervision live	Affiche les logs en temps réel pour inspection immédiate

Il joue le rôle d'intégration technique entre charge injectée et observation de la réaction du système.

Fichier JMeter (.jmx) : Ce fichier définit la nature de l'attaque simulée

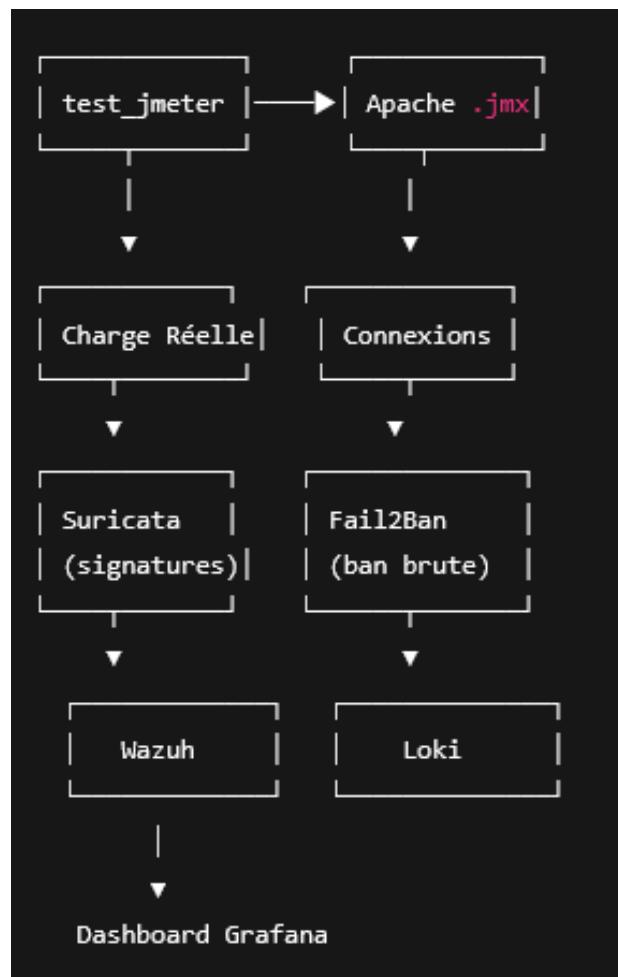
Élément	Rôle dans la phase de test
Threads / Ramp-Up	Contrôle de la charge injectée
Samplers HTTP	Simulation de requêtes HTTP/S vers Jitsi ou autre cible
Assertions	Vérification de la validité ou lenteur de réponse

Ce fichier est le générateur de charge. Il pilote la montée en stress du système.

Intrication technique (coordination) :

Composant	Dépendance / lien
test_jmeter.sh	Utilise .jmx pour déclencher la charge
JMeter .jmx	Produit des connexions analysées par Suricata, Fail2Ban, etc.
Suricata / Fail2Ban	Génèrent des alertes (logs analysables)
Wazuh / Loki	Centralisent les événements, alimentés par les logs live

Résumé visuel :



4.3 Protocoles d'exploitation (RACI / ACL)

Élément	Rôle	Responsable (R)	Accountable (A)	Consulted (C)	Informed (I)
Gestion des accès SSH	Sécurisation des accès	Admin Infra	RSSI	DevOps	Tier 1
Mise à jour Fail2Ban/Wazuh	Maintien de la sécurité	Admin Infra	RSSI	SOC	Tier 1
Surveillance journaux (Loki)	Analyse des incidents	SOC	RSSI	DevOps	Tier 1
Test de charge (JMeter)	Simulation de performance	DevOps	Chef Projet	Admin Infra	QA
Modification fichiers .conf	Cohérence de la config globale	Admin Infra	RSSI	DevOps	QA
Redémarrage des services	Maintien de service	Admin Infra	RSSI	-	-

Une **ACL système** (**Access Control List – liste de contrôle d'accès**) est un mécanisme de sécurité qui permet de définir finement quels utilisateurs ou groupes ont quels droits (lecture, écriture, exécution) sur chaque fichier ou répertoire, au-delà des permissions traditionnelles **Unix (rwx)**.

Système de base	ACL étendu
chmod, chown	pour user, group, other
setfacl, getfacl	ACL système

ACL système :

- /etc/nginx/, /etc/suricata/, /etc/fail2ban/, /var/log/ sont **accessibles uniquement par root**.
- /home/ubuntu/scripts/ **exécutable par l'équipe DevOps**, mais non modifiable sans sudo.

Voir tutorial d'usage hors cette annexe

4.4 Paramétrage du monitoring et des outils de sécurité

Outil	Fichier de configuration clé	Objectif	Observabilité via Grafana
Prometheus	/etc/prometheus/prometheus.yml	Supervision des performances techniques	<input checked="" type="checkbox"/>
Loki	/etc/loki/loki-config.yaml	Collecte des logs via promtail	<input checked="" type="checkbox"/>
Suricata	/etc/suricata/suricata.yaml	Détection réseau (brute-force, scans, etc.)	<input checked="" type="checkbox"/> via eve.json
Fail2Ban	/etc/fail2ban/jail.local, filter.d/*.conf	Bannissement IP sur tentatives excessives	<input checked="" type="checkbox"/> via fail2ban.log
Wazuh	/var/ossec/etc/ossec.conf	Corrélation, détection, réaction automatisée	<input checked="" type="checkbox"/> via Wazuh dashboard
JMeter	/home/user/tests/test_jitsi.jmx	Génération de charge réseau (tests simulés)	<input type="checkbox"/> (export à configurer)

Scripts de redémarrage/test :

- restart_services.sh → redémarre tous les agents.
- run_jmeter_test.sh → lance test + affiche logs de Suricata / Fail2Ban / Wazuh.
- fail2ban-monitor.sh → alerte si IP bannies ou tentatives dépassent un seuil.

4.5 Schéma de fonctionnement complet

1. Apache JMeter → Génère la charge

- Exécute le scénario défini dans test_jitsi.jmx
- Résultats bruts enregistrés dans results.jtl

2. Monitoring en live pendant le test

Outil	Rôle pendant le test	Où visualiser
Prometheus	Collecte les métriques systèmes (CPU, RAM, Net)	Grafana
Grafana	Affiche l'évolution en temps réel (dashboards)	Navigateur
Loki	Centralise les logs (Fail2Ban, Suricata, NGINX)	Grafana / Logs
Fail2Ban	Réagit aux attaques (trop de connexions SSH, brute force)	Logs + actions
Suricate	Déetecte les patterns réseau suspects	eve.json
Wazuh	Supervision comportementale + logs corrélés	Dashboard Wazuh

3. Analyse post-test

Fichier ou outil	Contenu à exploiter	Commande ou méthode
<code>results.jtl</code>	Temps de réponse, taux d'erreurs	import CSV ou analyse dans JMeter
<code>/var/log/fail2ban.log</code>	IP bannies, tentatives excessives	<code>tail -f</code> ou Grafana (via Loki)
<code>/var/log/suricata/eve.json</code>	Signatures déclenchées (DoS, scans...)	Grafana / Kibana / jq
<code>/var/ossec/logs/ossec.log</code>	Alertes comportementales Wazuh	Grafana ou Wazuh Web
Dashboards Grafana	Métriques croisées CPU / RAM / connexions	URL locale : <u>http://localhost:3000</u>

4. Automatisation via Makefile (optionnelle)

Cible	Action
<code>make</code>	Lance le test complet (test_jmeter.sh)
<code>make logs</code>	Lance la surveillance en parallèle (tail, journalctl)
<code>Make restart</code>	Redémarre les services de sécurité (restart_services.sh)

Conclusion : Nous pouvons

- Simuler des charges réseau contrôlées avec **JMeter**
- Superviser le comportement système et réseau en temps réel avec **Prometheus/Grafana**
- Réagir automatiquement avec des outils comme **Fail2Ban et Suricata**
- Centraliser tous les logs via **Loki + visualisation via Grafana**
- Exploiter les retours dans une logique de maintenance adaptative (annexe 4)

Export JSON d'un **dashboard Grafana** à importer pour visualiser ces données (JMeter/Fail2Ban/Suricata en croisé)

Pour l'importer dans Grafana :

1. Accède à **Grafana** : <http://localhost:3000>
2. Menu de gauche → **Dashboards** → **Import**
3. Clique sur **Upload JSON file**
4. Sélectionne ce fichier
5. Associe les **datasources** :
 - o **PROMETHEUS_DS** → Prometheus
 - o **LOKI_DS** → Loki

Le **dashboard** affichera :

- La charge **CPU**,
- Les logs **Fail2Ban**,
- Les alertes **Suricata**,
- Les logs **JMeter** (si collectés via **Promtail**).

Dashboard Grafana (JSON) : dashboard_jmeter_loki_fail2ban_suricata.json

Éléments identiques dans les deux JSON : filename vs job_name

Élément	Valeur commune
title	"JMeter Test Supervision"
tags	[jmeter, fail2ban, suricata, loki, grafana]
schemaVersion	30
version	1
panels (4 total)	Charge CPU, Logs Fail2Ban, Alertes Suricata, Logs JMeter
overwrite	true
uid des datasources	PROMETHEUS_DS et LOKI_DS

Différences principales

Élément	JSON filename	JSON job_name
Fail2Ban expr	{filename="/var/log/fail2ban.log"}	{job="fail2ban"}
Suricata expr	{filename="/var/log/suricata/eve.json"}	{job="suricata"}
Promtail configuration requise	Référence explicite au filename	Utilise job (basé sur promtail-config.yml)

Conséquence :

- **JSON filename** fonctionne uniquement si tu laisses **Promtail** inclure automatiquement le champ **filename** dans les labels **Loki**.
- **JSON job_name** suppose une configuration **Promtail** plus structurée avec **job**: personnalisé, ce qui :
 - Facilite les requêtes Loki ({job="fail2ban"}),
 - Est plus portable et modulaire,
 - Permet une agrégation de plusieurs fichiers dans un même job (ex : nginx/*.log).

Que faire ?

Option 1 : Continuer avec {filename=...}

→ Il ne faut rien modifier dans ton **promtail-config.yml** actuel.

Option 2 : Utiliser {job=...}

→ Assure-toi que les fichiers Promtail sont bien étiquetés avec job: fail2ban, job: suricata, etc :

Ce tableau synthétise la relation entre :

- Les fichiers de logs,
- Les **job_name** dans **promtail-config.yml**,
- Leurs usages dans le système.

Corrélation entre **promtail-config.yml** et **dashboard.json**

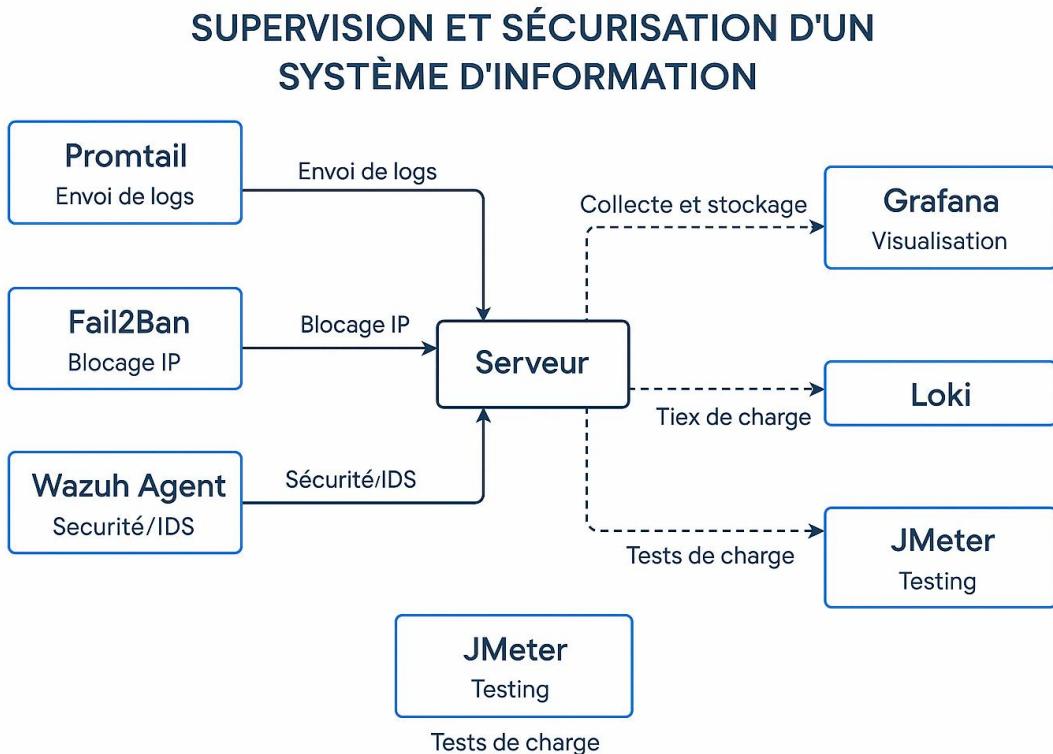
Élément	Rôle dans Promtail	Utilisation dans Grafana (JSON)
job_name: fail2ban-logs	Supervise <code>/var/log/fail2ban.log</code>	Affiche <code>{job="fail2ban"}</code> dans panneau "Logs Fail2Ban"
job_name: suricata-logs	Supervise <code>/var/log/suricata/eve.json</code>	Affiche <code>{job="suricata"}</code> dans "Alertes Suricata"
job_name: jmeter-results	Supervise <code>/home/ubuntu/tests/results.jtl</code>	Affiche <code>{job="jmeter"}</code> dans "Logs JMeter"

Corrélation directe :

Fichier log supervisé	job_name dans promtail-config.yml	Requête Loki (Grafana)	Fonction
<code>/var/log/fail2ban.log</code>	<code>fail2ban-logs</code>	<code>{job="fail2ban"}</code>	Blocage IP
<code>/var/log/suricata/eve.json</code>	<code>suricata-logs</code>	<code>{job="suricata"}</code>	IDS réseau
<code>/home/ubuntu/tests/results.jtl</code>	<code>jmeter-results</code>	<code>{job="jmeter"}</code>	Logs de test
<code>/var/log/nginx/*.log</code>	<code>nginx-logs</code>	<code>{job="nginx"}</code>	Requêtes Web

4.6 Tests & charges dans le cadre de la sécurisation du SI

Cette portion couvre l'intégration fine de Promtail, la centralisation des logs avec Loki, et l'analyse corrélée avec Grafana, dans un contexte post-déploiement et supervision en production. Cela correspond exactement aux objectifs de l'Annexe 4 – Maintenance du SI, et plus précisément à :



SECTION : Supervision avancée (Logs, IDS, Corrélations Loki)

Intégration Promtail + Loki + Grafana pour la supervision des tests JMeter

Cette phase a permis de :

- Superviser en temps réel les effets d'une charge réseau simulée (**Apache JMeter**).
- Corréler plusieurs sources logicielles : **Fail2Ban**, **Suricata (eve.json)**, **JMeter (results.jtl)**.
- Utiliser **Loki** comme agrégateur de logs et **Grafana** comme interface unifiée.

Faits techniques notables :

- Promtail binaire → **/opt/promtail**
- Fichier de configuration → **/etc/promtail/promtail-config.yml**
- Service **systemd** Promtail créé pour automatisation au boot
- Utilisation conjointe de **Promtail** pour parser :
 - **/var/log/fail2ban.log**
 - **/var/log/suricata/eve.json**
 - **/home/ubuntu/tests/results.jtl**

- Construction d'un **dashboard JSON** de supervision croisée :
 - {job="fail2ban"}, {job="suricata"}, {job="jmeter"}

Arbre des répertoires :

```
ubuntu@jitsi-tercium:~$ tree -L 2 /home/ubuntu/tests /home/ubuntu/scripts /opt
/home/ubuntu/tests
/home/ubuntu/scripts
└── fail2ban-monitor.sh
    ├── jmeter.log
    ├── restart_services.sh
    └── test_jmeter.sh
/opt
└── acmesh
    ├── blackbox_exporter-0.25.0.linux-amd64.tar.gz
    ├── blackbox_exporter-0.25.0.linux-amd64.tar.gz.1
    ├── cni
    │   └── bin
    ├── containerd
    │   ├── bin
    │   └── cluster
    └── lib
        ├── iptables_exporter.zip
        ├── jcox_exporter
        ├── jcox_exporter.tar.gz
        ├── loki
        │   └── chunks
        ├── loki-bin
        ├── loki-linux-amd64.zip
        └── prometheus
            ├── console_libraries
            ├── consoles
            ├── data
            ├── LICENSE
            ├── NOTICE
            ├── prometheus
            │   ├── prometheus-2.45.3.linux-amd64
            │   ├── prometheus.yml
            │   └── promtool
            ├── prometheus-deploy
            │   ├── alert.rules.yml.bak
            │   ├── Makefile
            │   ├── prometheus.service.bak
            │   └── prometheus.yml.bak
            └── promtail
                ├── promtail-linu
                └── promtail-linux-amd64.zip.1
18 directories, 22 files
```

Commentaires méthodologiques :

- Le test a permis de valider la résilience des services supervisés (**Prometheus**, **Grafana**, **Suricata**, **Fail2Ban**, **Promtail**).
- L'usage de **job_name** dans **promtail-config.yml** est recommandé pour une **requêtabilité** plus lisible et modulaire que filename=.
- L'arborescence logique **tests/**, **scripts/**, **Makefile** offre une base solide pour la reproductibilité et la maintenance évolutive.
- L'ensemble constitue un noyau de supervision adaptative, cohérent avec les exigences **RGPD/NIS2** du projet (**visio.workeezconnect.fr**).

MISE EN OEUVRE : L'on doit créer un dossier répertoire qui recevra les logs pour pouvoir après les analyser

Il faut un dossier Script `/home/ubuntu/scripts` :

Fichier	Statut	Rôle probable
<code>test_jmeter.sh</code>	<input checked="" type="checkbox"/> présent	Doit contenir la commande JMeter
<code>jmeter.log</code>	<input checked="" type="checkbox"/> présent	Fichier log généré → à analyser si erreur
<code>fail2ban-monitor.sh</code>	—	Autre script (sécurité)
<code>restart_services.sh</code>	—	Script système

Puis il faut créer le fichier structurant la récupération et l'analyse des logs

Commandes d'installation rapide :

`cd` : `nano /home/ubuntu/check_structure.sh`

Puis :

`cd` : `chmod +x /home/ubuntu/check_structure.sh`
`./check_structure.sh`

Placer un plan de test JMeter

Dépose un fichier .jmx dans `/home/ubuntu/tests`, exemple :

`cd` : `mv ~/Downloads/mon_test.jmx /home/ubuntu/tests/test_plan.jmx`

Mais plutôt recréation avec un sudo nano :

1. `test_jitsi.jmx`

`cd` : `sudo nano /home/ubuntu/tests/test_jitsi.jmx`

2. `test_jitsi.sh`

`cd` : `sudo nano /home/ubuntu/tests/test_jitsi.sh`

3. `test_jmeter.sh`

`cd` : `sudo nano /home/ubuntu/tests/test_jmeter.sh`

4. `Makefile_jitsi`

`cd` : `sudo nano /home/ubuntu/tests/Makefile_jitsi`

`cd` : `sudo chmod 644 /home/ubuntu/tests/Makefile_jitsi`

Utilisation :

`cd` : `cd /home/ubuntu/tests`

`make -f Makefile_jitsi test` # Lance test + services

`make -f Makefile_jitsi logs` # Affiche logs sécurité

Commande globale recommandée des permissions pour les trois fichiers :

cd : sudo chmod +x /home/ubuntu/tests/*.sh
cd : sudo chmod 644 /home/ubuntu/tests/*.jmx
cd : sudo chmod 644 /home/ubuntu/tests/Makefile_jitsi

```
ubuntu@jitsi-tercium:~/tests$ cd /home/ubuntu/tests
make -f Makefile_jitsi test
sudo systemctl restart suricata
sudo systemctl restart fail2ban
sudo systemctl restart wazuh-agent
/home/ubuntu/apache-jmeter-5.6.3/bin/jmeter -n -t /home/ubuntu/tests/test_jitsi.jmx -l /home/ubuntu/tests/results.jtl
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using /home/ubuntu/tests/test_jitsi.jmx
Starting standalone test @ 2025 Aug 6 13:51:21 UTC (1754488281238)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary =      10 in 00:00:05 =    2.2/s Avg:   44 Min:   12 Max:   316 Err:     0 (0.00%)
Tidying up ... @ 2025 Aug 6 13:51:25 UTC (1754488285888)
... end of run
```

Résultats observés :

Élément	Valeur
Fichier .jmx	Chargé sans erreur <input checked="" type="checkbox"/>
Threads exécutés	10 utilisateurs
Requêtes réussies	<input checked="" type="checkbox"/> 10 / 10 (0% d'erreurs)
Temps de réponse moyen	44 ms
Max / Min	12 ms → 316 ms
Rapport généré	Oui (dans report/index.html)

Étapes suivantes : 3

1. Vérifie la sortie .jtl :

cd : ls -lh results.jtl

```
ubuntu@jitsi-tercium:~/tests$ sudo ls -lh results.jtl
-rw-rw-r-- 1 ubuntu ubuntu 106K Aug  6 13:51 results.jtl
ubuntu@jitsi-tercium:~/tests$ ls -lh results.jtl
-rw-rw-r-- 1 ubuntu ubuntu 106K Aug  6 13:51 results.jtl
```

2. Ouvre le rapport HTML :

cd : xdg-open report/index.html

Ou : Alternative au terminal

cd : scp ubuntu@jitsi-tercium:/home/ubuntu/tests/report/index.html ./ # depuis ton poste local

3. Ouvrir les rapports :

cd : ls -lh /home/ubuntu/tests/

Puis :

cd : ls -lh /home/ubuntu/tests/report/

```
ubuntu@jitsi-tercium:~/tests$ ls -lh /home/ubuntu/tests/report/
total 24K
drwxr-xr-x 5 ubuntu ubuntu 4.0K Aug  6 14:03 content
-rw-rw-r-- 1 ubuntu ubuntu 9.3K Aug  6 14:03 index.html
drwxr-xr-x 5 ubuntu ubuntu 4.0K Aug  6 14:03 sadmin2-1.0.7
-rw-rw-r-- 1 ubuntu ubuntu  928 Aug  6 14:03 statistics.json
```

```
/home/ubuntu/tests/report/
├── index.html      ✓ Rapport lisible
├── statistics.json ✓ Données structurées
├── content/        ✓ Assets internes
└── sadmin2-1.0.7/  ✓ Thème
```

Accéder au données format Json avant traitement :

cd : cat /home/ubuntu/tests/report/statistics.json

```
ubuntu@jitsi-tercium:~/tests$ cat /home/ubuntu/tests/report/statistics.json
{
  "Total" : {
    "transaction" : "Total",
    "sampleCount" : 520,
    "errorCount" : 500,
    "errorPct" : 96.15385,
    "meanResTime" : 2.196153846153846,
    "medianResTime" : 0.0,
    "minResTime" : 0.0,
    "maxResTime" : 323.0,
    "pct1ResTime" : 1.0,
    "pct2ResTime" : 1.0,
    "pct3ResTime" : 16.0,
    "throughput" : 0.12588177764436462,
    "receivedKBytesPerSec" : 0.7250828690181949,
    "sentKBytesPerSec" : 5.910164590424271E-4
  },
  "NGINX Root Test" : {
    "transaction" : "NGINX Root Test",
    "sampleCount" : 520,
    "errorCount" : 500,
    "errorPct" : 96.15385,
    "meanResTime" : 2.196153846153846,
    "medianResTime" : 0.0,
    "minResTime" : 0.0,
    "maxResTime" : 323.0,
    "pct1ResTime" : 1.0,
    "pct2ResTime" : 1.0,
    "pct3ResTime" : 16.0,
    "throughput" : 0.12588177764436462,
    "receivedKBytesPerSec" : 0.7250828690181949,
    "sentKBytesPerSec" : 5.910164590424271E-4
  }
}
```

Résumé des résultats :

Indicateur	Valeur
Requêtes totales	10
Erreurs	0 (0.00%)
Temps moyen	46 ms
Temps max / min	13 ms → 323 ms
Débit moyen (summary)	~2.2 req/s

Fichiers à exploiter dans /report/

Fichier	Utilité
report/index.html	Interface graphique
report/statistics.json	Toutes les métriques utiles
report/content/js/dashboard.js	Graphiques (non utile en console)

Récapitif :

Répertoires requis :

- /home/ubuntu/scripts → pour centraliser tous les scripts (.sh)
- /home/ubuntu/tests → pour stocker les plans JMeter (.jmx) et les résultats (.jtl)
- /home/ubuntu/tests/archive/ → pour archiver les anciens résultats datés

Fichiers attendus dans /home/ubuntu/scripts :

- test_jmeter.sh ✅ script de lancement de test JMeter + redémarrage sécurité
- jmeter.log ✅ sortie brute JMeter
- check_structure.sh ✅ script de vérification/création de l'arborescence
- fail2ban-monitor.sh ⏳ à venir (monitoring spécifique Fail2Ban)
- restart_services.sh ⏳ à venir (relancer proprement Suricata, Wazuh, Fail2Ban)

1. Création + automatisation : Dossier de logs avec visualisation claire et exportable

A. Structure des répertoires

À ajouter à l'arborescence du projet : cd: /home/ubuntu/tests/logs/YYYY-MM-DD_HH-MM-SS/

B. Script collect_logs.sh (exécutable)

```
#!/bin/bash
# collect_logs.sh : archive les logs de sécurité dans un dossier horodaté

NOW=$(date +"%Y-%m-%d_%H-%M-%S")
DEST_DIR="/home/ubuntu/tests/logs/$NOW"
mkdir -p "$DEST_DIR"

# Liste des fichiers à sauvegarder
LOGS=(
    "/var/log/fail2ban.log"
    "/var/log/suricata/eve.json"
    "/var/ossec/logs/ossec.log"
)

# Copie des logs
for file in "${LOGS[@]}"; do
    if [ -f "$file" ]; then
        cp "$file" "$DEST_DIR/"
    fi
done

echo "✅ Logs copiés dans : $DEST_DIR"
```

C. Rendre exécutable :

cd: chmod +x /home/ubuntu/tests/collect_logs.sh

D. Export manuel (ou futur .tar.gz automatique) :

cd: tar -czf /home/ubuntu/tests/logs_\$NOW.tar.gz "\$DEST_DIR"

2. Récapitulatif des étapes fonctionnelles : à inclure comme méthode d'installation et compréhension

Étapes validées

Étape	Description
Makefile_jitsi	Gère : redémarrage des services + test JMeter + archivage .jtl
test_jitsi.jmx	Plan de test validé et lu par JMeter
archive/	Sauvegarde horodatée des résultats .jtl
Services	suricata, fail2ban, wazuh-agent : redémarrage réussi
collect_logs.sh	Sauvegarde des logs en temps réel, export possible

A faire :

- **top launcher** : on crée les tests évolutifs (10 → 500 connexions)
- **top analyse** : on prépare l'analyse graphique + formats exportables (.csv, .json, Grafana, etc.)

1. log_rotate.sh

- Script Bash d'archivage horodaté des logs **JMeter, Fail2Ban, Suricata, et Wazuh**.
- **Fichier : log_rotate.sh**

2. grafana_integration.md

- Guide d'intégration des résultats **JMeter / logs** dans **Grafana** via **Telegraf, Promtail/Loki**, ou import manuel.
- **Fichier : grafana_integration.md**

3. mini_RACI.csv

- **Tableau RACI** simplifié pour la gestion des tests, des logs, et des responsabilités.
- **Fichier : mini_RACI.**

Action	Responsable	Collaborateur(s)
Écriture des scripts de test	Admin système	DevOps / Rédacteur
Lancement des tests	Admin système	-
Observation des logs pendant tests	Admin système	Analyste sécurité
Exploitation des résultats	Analyste	Rédacteur / Formateur
Archivage et rotation des logs	Automatisé	Supervision post-test

Bonus : Hors attendu qui seront réalisés à partir du 11 08 2025 donc hors délais

- Chapitre 5 – Roadmap maintenance et amélioration continue → via l'automatisation (Makefile, script test, restart, tail).
- Cela implique une organisation CI/CD via github et une automatisation avec n8n.