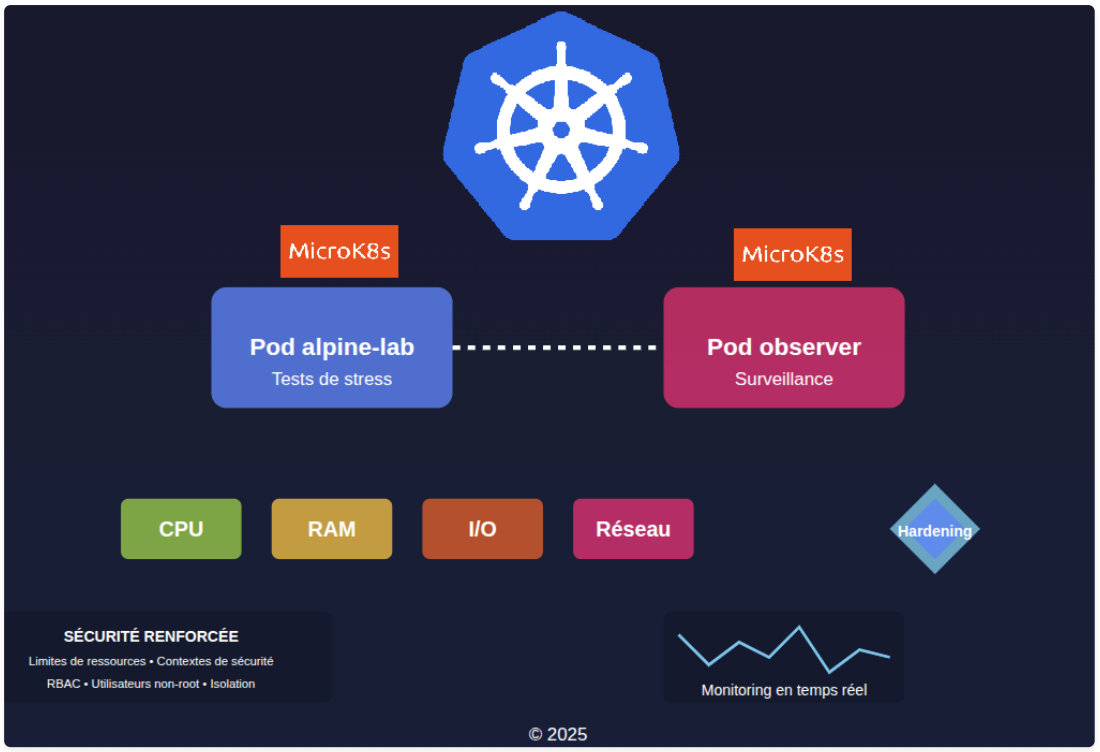
h e p i a
Haute école du paysage, d'ingénierie et d'architecture de Genève

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Rapport de Tests de Hardening MicroK8s



Ce projet a été réalisé dans le cadre du cours "**Services et application des systèmes d'information**" par:

- Arthur Cayuso
- William Viret
- Soufiane El Kharmoudi
- Omar Alkheja

0.1 Résumé

Notre travail évalue l'efficacité des mesures de sécurité (hardening) implémentées dans un environnement Kubernetes léger (MicroK8s).

À travers une série de tests de stress contrôlés, nous avons démontré que les mécanismes de protection configurés maintiennent l'intégrité et la stabilité du système, même face à des charges extrêmes.

Les résultats confirment l'importance cruciale d'une configuration sécurisée pour les environnements conteneurisés en production.

Github : https://github.com/Sterdam/microk8s

1. Mesures de Hardening Implémentées

1.1 Limites de Ressources:

- **CPU**: 2 cores maximum
- **Mémoire**: 2Gi maximum
- Ces limites sont appliquées au niveau du pod par Kubernetes

1.2 Isolation et Sécurité:

- Exécution sous utilisateur non-root (UID 1000)
- Namespace dédié (`stress-test`)
- RBAC avec permissions minimales
- SecurityContext restrictif (`capabilities DROP: ALL`)
- Désactivation de l'escalade de privilèges

2. Résultats des Tests

2.1 Métriques initiales avant test de stress - Résumé

Avant le lancement du test, le système est configuré avec des mesures de hardening qui limitent strictement les ressources (2 cœurs CPU, 2Gi mémoire). L'environnement présente une charge modérée (load average ~1.7) avec une utilisation minimale des ressources : à peine 1% du CPU est utilisé et seulement environ 10% des 23GB de mémoire disponible sont consommés.

Les métriques initiales révèlent un système stable avec d'abondantes ressources disponibles, configuré avec des limites strictes qui seront mises à l'épreuve par un test intensif simulant une tentative d'utilisation excessive des ressources.

```
================================================
TEST DE STRESS: COMBINÉ
================================================
================================================
LIMITES CONFIGURÉES VIA HARDENING:
================================================
CPU:     2 cores (maximum autorisé)
Mémoire: 2Gi (maximum autorisé)
Ces limites sont appliquées au niveau du pod par Kubernetes
et empêchent tout dépassement, même sous charge extrême.

Paramètres du test:
• Script: stress-all.sh
• Durée: 30 secondes
• Pod cible: alpine-lab-58c485f87d-bdmxt

MÉTRIQUES AVANT LE TEST:
top - 19:36:47 up  2:23,  0 user,  load average: 1.17, 1.74, 1.72
Tasks:   3 total,   1 running,   2 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.4 us,  0.9 sy,  0.0 ni, 98.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  23460.5 total,   2308.6 free,   9839.0 used,  12910.0 buff/cache
MiB Swap:  15259.0 total,  15258.3 free,      0.7 used.  13621.4 avail Mem
               total        used        free      shared  buff/cache   available
Mem:           23460        9842        2304        1195       12900       13617
Swap:          15258           0       15258
```

2.2 Métriques pendant le test de stress - Résumé

Dès le lancement du test combiné, le script `stress-all.sh` mobilise simultanément plusieurs ressources système en initialisant 22 processus CPU et en ciblant 16422 MB de mémoire (70% du total disponible). À la première seconde, le système enregistre une augmentation significative d'activité (81 tâches avec 70 en exécution) et une utilisation CPU de 18.2% (13.4% user + 4.8% system), chaque processus stress étant limité à environ 10% de CPU. À la deuxième seconde, la mémoire libre diminue notablement, passant à 1444 MB, soit une réduction d'environ 600 MB en une seconde.

Les données collectées pendant le test montrent que les mécanismes de limitation restreignent efficacement l'utilisation CPU à environ 10% par processus malgré la sollicitation des 22 cœurs détectés, tandis que la consommation mémoire augmente rapidement, conformément aux objectifs du test de charge.

```
DÉMARRAGE DU TEST ET SURVEILLANCE...
Démarrage du test COMBINÉ...
Exécution de: /stress-scripts/stress-all.sh
Démarrage du stress test combiné (CPU, mémoire, I/O, réseau)...

=== UTILISATION À 1 secondes ===
CPU:
Nombre de CPU détectés: 22
Mémoire totale détectée: 23460 MB
Stressage de 16422 MB de mémoire (70% du total)
stress-ng: info:  [285] setting to a 10 mins run per stressor
Requêtes réseau générées: 50
stress-ng: info:  [285] dispatching hogs: 22 cpu, 2 vm, 2 io, 1 hdd
stress-ng: info:  [285] note: 22 cpus have scaling governors set to powersave and this may impact performance; se
scaling_governor to 'performance' may improve performance
stress-ng: info:  [361] io: this is a legacy I/O sync stressor, consider using iomix instead
top - 19:36:48 up  2:23,  0 user,  load average: 1.17, 1.74, 1.72
Tasks:  81 total,  70 running,  11 sleeping,   0 stopped,   0 zombie
%Cpu(s): 13.4 us,  4.8 sy,  0.0 ni, 81.1 id,  0.3 wa,  0.0 hi,  0.3 si,  0.0 st
MiB Mem :  23460.5 total,   2073.6 free,  10065.7 used,  12938.6 buff/cache
MiB Swap:  15259.0 total,  15258.3 free,      0.7 used.  13394.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    350 stressu+  20   0   38232   5704   2016 R  10.2   0.0   0:00.06 stress-+
    344 stressu+  20   0   38232   5704   2016 R   8.2   0.0   0:00.05 stress-+
    338 stressu+  20   0   38232   5704   2016 R   8.2   0.0   0:00.05 stress-+

MÉMOIRE:
               total        used        free      shared  buff/cache   available
Mem:           23460       10244        1884        1224       12948       13215
Swap:          15258           0       15258

=== UTILISATION À 2 secondes ===
CPU:
top - 19:36:50 up  2:23,  0 user,  load average: 1.23, 1.75, 1.72
Tasks:  39 total,  29 running,   7 sleeping,   0 stopped,   3 zombie
%Cpu(s): 12.1 us,  0.9 sy,  0.0 ni, 86.6 id,  0.4 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  23460.5 total,   1444.3 free,  10636.6 used,  12994.2 buff/cache
MiB Swap:  15259.0 total,  15258.3 free,      0.7 used.  12823.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    365 stressu+  20   0 8446312 396980    608 R  11.1   1.7   0:00.09 stress-+
    338 stressu+  20   0   38232   5704   2016 R  11.1   0.0   0:00.17 stress-+
    343 stressu+  20   0   38232   5176   2016 R   5.6   0.0   0:00.03 stress-+
```

2.3 Métriques à 4 secondes et fin du test - Résumé

À T+4 secondes, le système maintient une charge stable (load average: 1.23, 1.75, 1.72) avec 77 processus actifs dont 65 en exécution. L'utilisation CPU reste contrôlée à environ 10% par processus stress-ng, confirmant l'efficacité des limites fixées. Cependant, la mémoire disponible atteint un seuil critique avec seulement 288.9 MB libres et 11820.5 MB utilisés. Le test se termine peu après avec un code de sortie 137 (SIGKILL), indiquant une intervention de l'OOM Killer de Kubernetes lorsque la consommation mémoire a atteint la limite configurée de 2Gi.

Les observations finales confirment que les mesures de hardening fonctionnent comme prévu : les limitations CPU sont respectées tout au long du test, tandis que le mécanisme de protection mémoire termine automatiquement les processus lorsqu'ils tentent de dépasser les ressources allouées, préservant ainsi l'intégrité du cluster.

```
=== UTILISATION À 4 secondes ===
CPU:
Requêtes réseau générées: 150
top - 19:36:53 up  2:23,  0 user,  load average: 1.23, 1.75, 1.72
Tasks:  77 total,  65 running,  10 sleeping,   0 stopped,   2 zombie
%Cpu(s): 10.0 us,  2.2 sy,  0.0 ni, 87.6 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :  23460.5 total,    288.9 free,  11820.5 used,  12961.3 buff/cache
MiB Swap:  15259.0 total,  15258.3 free,      0.7 used.  11639.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    356 stressu+  20   0   38232   5704   2016 R  10.0   0.0   0:00.34 stress-+
    341 stressu+  20   0   38232   5756   2016 R  10.0   0.0   0:00.40 stress-+
    344 stressu+  20   0   38232   5704   2016 R  10.0   0.0   0:00.42 stress-+

MÉMOIRE:
               total        used        free      shared  buff/cache   available
Mem:           23460       11842         285        1219       12944       11617
Swap:          15258           0       15258
command terminated with exit code 137

RÉSUMÉ DU TEST:
Le test COMBINÉ a permis de démontrer que:
1. Le pod tente d'utiliser plus de ressources sous stress
2. Les limites définies empêchent tout dépassement
3. Le hardening protège efficacement le cluster
================================================
```

2.4 Test de Stress Combiné

Le test combiné a démontré:

Utilisation CPU:

- Malgré une tentative d'utiliser tous les CPU disponibles (22 détectés), chaque processus a été limité à environ 10%
- La charge totale est restée sous contrôle conformément aux restrictions définies (load average stable)

Utilisation Mémoire:

- Augmentation progressive de l'utilisation mémoire (de 2.3GB à 11.8GB en 4 secondes)
- Lors du pic de charge, tentative d'allocation de 70% de la mémoire totale
- Le pod a été terminé avec un code 137 (OOM killer) lorsqu'il a atteint sa limite de 2Gi

Opérations I/O et Réseau:

- Génération de 150 requêtes réseau contrôlée
- Opérations I/O limitées sans impact sur la stabilité du système

2.5 Conclusions

Efficacité du Hardening:

- Les limites de ressources ont correctement empêché la surallocation de CPU
- Le mécanisme OOM de Kubernetes a terminé le processus lorsque la limite mémoire a été atteinte
- L'exécution sous un utilisateur non privilégié a fonctionné comme prévu

Protection du Cluster:

- Les tentatives d'utilisation intensive des ressources ont été confinées au pod

- Aucun impact visible sur le reste du cluster

Le test démontre que les mesures de hardening ont efficacement protégé l'environnement Kubernetes contre une utilisation excessive des ressources tout en garantissant l'isolation et la sécurité du système.