Installation d’une plateforme opensource de détection et gestion des vulnérabilités

Nicolas.Schmitz@ens-lyon.fr - RSSI Ens de Lyon – 17/05/2020 – V0.2

Table des matières

Introduction :

Ce document s'adresse aux RSSI et ingénieurs sécurité qui doivent jongler entre plusieurs « scanners de vulnérabilité », chacun doté de leur propre base d'équipements à scanner, de leur propres reportings, etc.

Une fois les scans définis et effectués dans chaque outil, le RSSI doit encore injecter manuellement les vulnérabilités trouvées dans un circuit permettant la remédiation : plateforme de ticketing, mails, fichiers .xls partagés, etc..

Une fois la vulnérabilité dans le circuit, le RSSI doit s'assurer de la bonne résolution de l'incident, alors même que ces circuits sont souvent inadaptés à de la gestion d'incident SSI.

Difficile dans ses conditions d'être efficace… Mais ça, pour moi, c'était avant :-)

Dans ce document nous allons installer un ensemble d'outils de sécurité qui vont travailler ensemble grâce à un orchestrateur qui va leur apporter une vrai plus-value.

L'orchestrateur : PatrowlManager. C'est un outil opensource français récent et très prometteur permettant de créer des scans sur différents moteurs, puis de centraliser les résultats sur une base avant de pousser les événements pertinents dans un outil de gestion pour traitement par les équipes d'exploitation.

Les scanners : On se concentrera ici sur 3 scanners, pour l'exemple :

- OpenVAS est un excellent scanner opensource généraliste idéal pour scanner des sous-réseaux complets. Ce produit est utilisé comme moteur par de nombreux logiciels commerciaux du marché.

- Nmap va scanner l'infra, et comparer les versions des applications avec la base de vulnérabilité sur vulners.com. Vous saurez rapidement si des applications à l'écoute d'un port sur votre réseau sont vulnérables.

- Arachni sera notre scanner de vulnérabilité web (injections SQL, XSS, file inclusion..)

La réponse à incidents : TheHive sera notre plateforme de gestion d'incident, cet outil est aux adminsys ce que JIRA est aux développeurs. Il nous permettra d'affecter les incidents aux équipes opérationnelles, et offre de nombreux tableaux de bord.

Ce n'est qu'un **aperçu** des possibilités car les interconnexions possibles sont très nombreuses, avec de nombreux outils...

Ce document ne couvre que peu l'usage des produits en eux même mais plutôt l'aspect technique de l'installation. Une vidéo est disponible pour vous montrer très rapidement un cas d'usage, une fois la plateforme installée.

Pré-requis et informations utiles :

- On installera l'ensemble de la solution sur une VM unique en Debian Buster. Pour les besoins de la maquette, la config utilisée est confortable avec 4 CPU, 12Go de Ram, et 64Go d'espace disque SSD.
- Les besoins en production seront bien sûr dépendants de la quantité d'éléments scannés.
- Dans toute la documentation ci-dessous, la VM aura l'ip 192.168.0.154, il faudra bien sûr remplacer par l'ip de votre VM.
- Certaines des manipulations décrites ci-dessous devront être effectuées en root.
- Cette documentation ne couvre pas le changement des différents mots de passe par défaut, le chiffrement ssl, autant de choses qu'il faudra bien sûr faire avant un passage en prod.

Installation de PatrowlManager :

On se base sur la doc dispo ici :
https://github.com/Patrowl/PatrowlDocs/blob/master/installation/installation-guide.md

On va utiliser l'installation Docker avec stockage persistent : le plus simple à maintenir :

```
apt-get install docker-compose
mkdir /sources; cd /sources/
git clone https://github.com/Patrowl/PatrowlManager.git
cd PatrowlManager/
```

On édite à présent le DockerFile pour rendre le stockage persistent.

```
mkdir /data/patrowl_pg_data
vi docker-compose.yml
```

On ajoute la ligne

```
- /data/patrowl_pg_data:/var/lib/postgresql/data/
```

dans la section « volumes » du docker « db »

On peut maintenant lancer la construction et le démarrage des dockers :

```
docker-compose build --force-rm
docker-compose up -d
```

Vous pouvez maintenant vous connecter à http://192.168.0.154:8083 avec admin/Bonjour1!

Par défaut, PatrowlManager est un peu un général sans armée, on va donc lui fournir quelques soldats, appelés ici « engines ».

Installation de PatrowlEngine :

En théorie, il faut utiliser le dépôt https://github.com/Patrowl/PatrowlEngines.git, cependant j'ai rencontré quelques bugs qui avaient déjà été corrigés par Harduino de Siemonster. Grand merci à lui! Bien sûr, ce n'est probablement qu'une question de jours avant que les PR soient acceptées dans le dépôt principal.

```
cd /sources/ ;
git clone -b develop https://github.com/harduino/PatrowlEngines.git
```

Chaque engine va être déposé dans /sources/PatrowlEngines/engines.

Installation d'OpenVAS et de son engine :

Le moteur :
Malheureusement PatrowlEngine n'est pas encore capable de s'interfacer avec la dernière version d'OpenVAS (gvm11), on va donc partir sur un OpenVAS 9 fourni en docker qui est amplement suffisant.

On lance l'image mikesplain/openvas avec les options qui vont bien pour pouvoir s'y connecter et avoir de la persistance de donnée :

```
mkdir -p /data/openvas/
```

Sur une ligne :

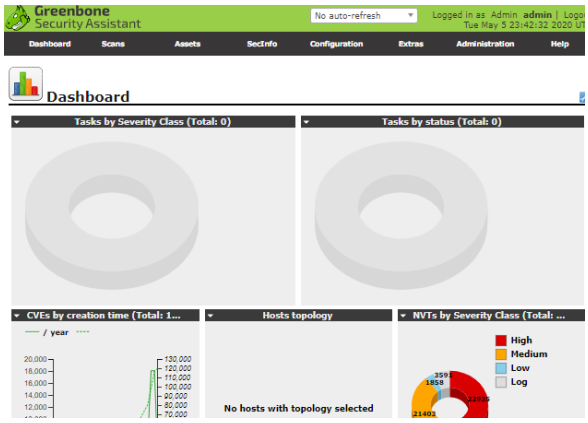
```
docker run --rm -d -p 4443:443 -p 9390:9390 -e PUBLIC_HOSTNAME=192.168.0.154 -v /data/openvas/:/var/lib/openvas/mgr/ --name openvas mikesplain/openvas:9
```

Attention : openvas met plusieurs minutes à s'initialiser (la récupération des signatures est longue). N'hésitez pas à aller prendre un café ! Vous pourrez vérifier l'avancement avec un

```
docker exec -it openvas top
```

Hint : tant que le CPU est à 100 % il faut attendre :-)

Vous pouvez maintenant vous connecter sur https://192.168.0.154:4443 avec admin/admin et accéder à une console vierge openvas :



C'est tout avec le moteur
OpenVAS, vous n'y définirez aucun réseau ou scan : tout se fera à partir de Patrowl via l'engine dédié :-)

L'engine :

```
cd /sources/PatrowlEngines/engines/openvas/
cp openvas.json.sample openvas.json
vi openvas.json
```

On modifie dans le fichier openvas.json :

```
gmp_host : 192.168.0.154
gmp_password : admin
```

Note : le projet python-gvm ne compilait pas pour cause d'absence du fichier setup.py. J'ai donc dû lui en fournir un manuellement, et modifier le DockerFile pour le gérer :

```
cd /sources/PatrowlEngines/engines/openvas/
wget https://raw.githubusercontent.com/greenbone/python-gvm/v1.3.0/setup.py
vi DockerFile
#Ajout ligne COPY setup.py python-gvm/ avant ligne « RUN pip3 #install -e python-gvm »
```

On peut maintenant construire et lancer le docker :

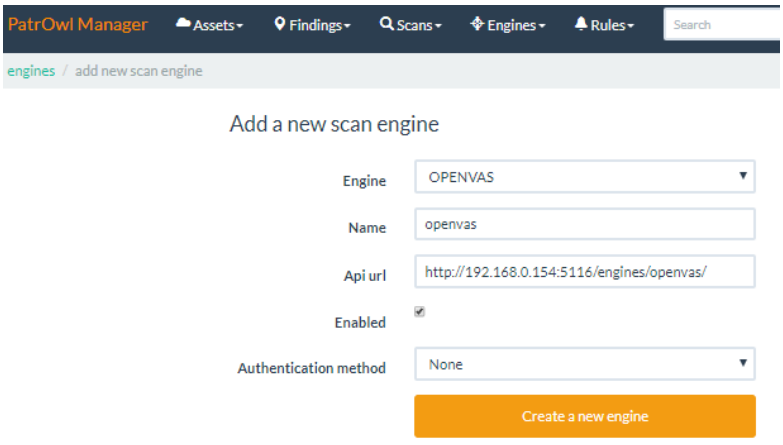
```
cd /sources/PatrowlEngines/engines/openvas/
docker build --tag "patrowl-openvas" .
```

En une seule ligne :

```
docker run -d --rm -p 5116:5016 -v
/sources/PatrowlEngines/engines/openvas/openvas.json:/opt/patrowl-engines/openvas/
openvas.json --name="openvas-engine" patrowl-openvas
```

Intégration de l'engine dans PatrowlManager :

On retourne sur http://192.168.0.154:8083/ menu « Engines »/« +Add scan engine instance » et on donne les références de notre docker fraîchement créé :



L'engine doit ensuite apparaître dans le menu « Engines »/ « List Engines », avec Oper.State à « Ready ».

Ajout de la policy par défaut :

Il reste un dernier paramétrage : il n'y a pas par défaut de policy pour l'engine openvas, on va donc la créer : Menu « Engines »/ « Add Policy » :

Name : openvas
Description : openvas
Options : {"enable_start_task": "True", "enable_create_task": "True", "enable_create_target": "True"}
Scopes : System infrastructure

Installation de l'engine Nmap :

Pas besoin d'installer de moteur pour nmap : construire le docker suffira !

```
cd /sources/PatrowlEngines/engines/nmap
docker build --tag "patrowl-nmap" .
docker run -d --rm -p 5101:5001 --name="nmap-engine" patrowl-nmap
```

Intégration de l'engine dans PatrowlManager :

Exactement comme pour l'engine openvas, sauf que cette fois l'Api url est donc :

```
http://192.168.0.154:5101/engines/nmap/
```

Là encore, il doit apparaître « Ready » dans le menu « Engines »/ « List Engines ».

Installation d'Arachni et de son engine :

Le moteur :

```
docker run -d --rm -p 7331:7331 -p 9292:9292 --name arachni arachni/arachni:latest
```

C'est tout ! Vous pouvez contrôler que tout va bien en vous connectant à l'interface web sur http://192.168.0.154:9292 avec admin@admin.admin/administrator

L'engine :

```
cd /sources/PatrowlEngines/engines/arachni
cp arachni.json.sample arachni.json
vi arachni.json
```

On indique :

```
username: arachni
password: password
```

Le MarkupSafe 1.0 indiqué dans requirements.txt ne compile pas, on change la ligne pour MarkupSafe==1.1.1

On builde et on démarre :

```
docker build --tag "patrowl-arachni" .
```

En une ligne :

```
docker run -d --rm -p 5105:5005 -v
/sources/PatrowlEngines/engines/arachni/arachni.json:/opt/patrowl-engines/arachni/
arachni.json --name="arachni-engine" patrowl-arachni
```

Intégration de l’engine dans PatrowlManager :

Exactement comme pour l’engine nmap, mais avec l’url : http://192.168.0.154:5105/engines/arachni/

Là encore, il doit apparaître « Ready » dans le menu « Engines »/ « List Engines ».

Ajout d’une policy :

Par défaut la policy fournie pour Arachni est dédiée XSS. On peut en créer une nouvelle plus complète :

Dans Patrowl : Menu « Engines »/ « Add Policy » :

Name : arachni default
Description : full scan
Options : {"http": {"user_agent": "Arachni/v2.0dev-FullScan", "request_queue_size": 1000, "request_concurrency": 30, "request_redirect_limit": 5}, "audit": {"xmls": true, "forms": true, "jsons": true, "links": true, "cookies": false, "headers": false, "ui_forms": true, "ui_inputs": true, "link_templates": [], "parameter_values": true, "with_both_http_methods": false, "exclude_vector_patterns": [], "include_vector_patterns": []}, "input": {"force": false, "values": {}, "without_defaults": true}, "scope": {"https_only": false, "exclude_binaries": false, "include_subdomains": false, "auto_redundant_paths": 10, "exclude_file_extensions": ["pdf", "css", "ico", "jpg", "svg", "png", "gif", "jpeg"]}, "checks": ["xss", "xss_dom", "xss_dom_script_context", "xss_event", "xss_path", "xss_script_context", "xss_tag", "xxe", "common_directories", "allowed_methods", "backdoors", "backup_directories", "backup_files", "captcha", "code_injection", "code_injection_php_input_wrapper", "code_injection_timing", "common_admin_interfaces", "common_directories", "common_files", "cookie_set_for_parent_domain", "credit_card", "csrf", "cvs_svn_users", "directory_listing", "emails", "file_inclusion", "form_upload", "hsts", "htaccess_limit", "html_objects", "http_only_cookies", "http_put", "insecure_client_access_policy", "ldap_injection", "localstart_asp", "mixed_resource", "no_sql_injection", "no_sql_injection_differential", "origin_spoof_access_restriction_bypass", "os_cmd_injection", "os_cmd_injection_timing", "password_autocomplete", "path_traversal", "private_ip", "response_splitting", "rfi", "session_fixation", "source_code_disclosure", "sql_injection", "sql_injection_differential", "sql_injection_timing", "ssn", "trainer", "unencrypted_password_forms", "unvalidated_redirect", "unvalidated_redirect_dom", "webdav", "x_frame_options", "xpath_injection", "insecure_cookies", "insecure_cors_policy", "insecure_cross_domain_policy_access", "insecure_cross_domain_policy_headers"], "max_timeout": 3600, "browser_cluster": {"pool_size": 12, "job_timeout": 10, "ignore_images": true, "worker_time_to_live": 100}, "no_fingerprinting": true}
Is default : oui
Scopes : Web App

Attention ! Dans la liste des checks, ne pas ajouter le check « interesting_responses » qui fait planter l’engine.

Installation de TheHive :

On y est presque, TheHive nous permettra de traiter en collaboratif et de manière efficace les alertes générées par les outils installés précédemment ! Et oui, toutes ces alertes, vous n’y arriverez pas seul ! :-)

On se base sur cette doc :
https://github.com/TheHive-Project/TheHiveDocs/blob/master/installation/install-guide.md#docker

On prépare donc les répertoires qui vont accueillir les données persistantes :

```
mkdir /data/elasticsearch
chmod 770 /data/elasticsearch
mkdir /data/thehiveconf
```

On prépare le docker-compose.yml :

```
mkdir /sources/thehive
cd /sources/thehive/
vi docker-compose.yml
```

Voici le contenu du fichier, avec stockage persistent pour l'ElasticSearch et la conf TheHive :

```
version: "2"
services:
  elasticsearch:
    image: elasticsearch:6.8.8
    environment:
      - http.host=0.0.0.0
      - discovery.type=single-node
    ulimits:
      nofile:
        soft: 65536
        hard: 65536
    ports:
      - "0.0.0.0:9200:9200"
    volumes:
      - /data/elasticsearch:/usr/share/elasticsearch/data
  cortex:
    image: thehiveproject/cortex:latest
    depends_on:
      - elasticsearch
    ports:
      - "0.0.0.0:9001:9001"
  thehive:
    image: thehiveproject/thehive:latest
    depends_on:
      - elasticsearch
      - cortex
    ports:
      - "0.0.0.0:9000:9000"
    volumes:
      - /data/thehiveconf/application.conf:/etc/thehive/application.conf
    command: --no-config
```

On installe le fichier de conf par défaut :

```
cd /data/thehiveconf/
```

en une ligne :

```
wget https://raw.githubusercontent.com/TheHive-Project/TheHive/master/conf/application.sample -O application.conf
```

```
vi application.conf
```

Il faut dé-commenter `play.http.secret.key` et y placer une chaîne de caractères aléatoires, puis indiquer l'uri suivante pour elasticsearch : `uri = "http://192.168.0.154:9200/"`

On peut à présent télécharger et installer nos 3 dockers (note :j'ai laissé cortex pour l'instant inutilisé, vous me remercierez plus tard :-))

```
cd /sources/thehive/
docker-compose up -d
```

Après quelques instants, vous pouvez accéder à http://192.168.0.154:9000/ et cliquer sur le bouton « Update Database ».

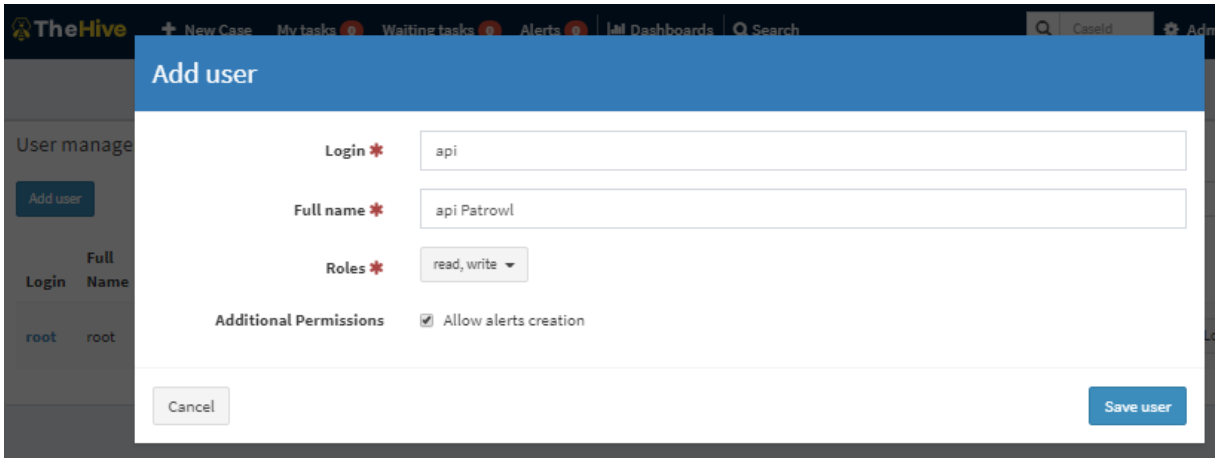
Après avoir initialisé ses bases Elastic, TheHive vous proposera la création d'un compte admin puis vous pourrez découvrir l'interface.

Paramétrage du connecteur Patrowl => TheHive:

La connexion :

Afin que Patrowl puisse « pousser » des alertes dans TheHive, il faut d'abord générer et récupérer une clef api dans celui-ci.

Dans votre instance TheHive, allez dans le menu « Admin »/ « Users ». Cliquez sur « Add User » puis indiquez les éléments suivants :



Ne pas oublier de cocher « Allow alerts creation »

Une fois le user enregistré, cliquez sur son bouton « Create API Key », puis « Reveal : » copiez la valeur qui s'affiche.

Dans Patrowl à présent, allez dans le menu « Admin » / « Settings » puis cliquez sur « Add new setting » :

- Dans le champ Key, indiquez : alerts.endpoint.thehive.apikey
- Dans le champ Value, collez la valeur de votre API Key générée dans TheHive

Recommencez l'opération pour l'url, avec les valeurs suivantes :

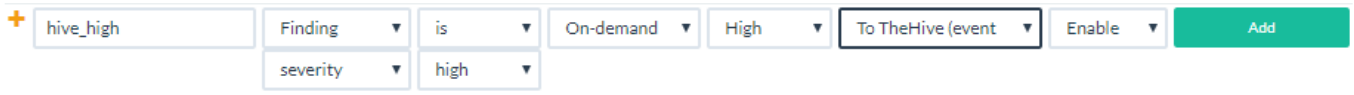
- Dans le champ Key, indiquez alerts.endpoint.thehive.url
- Dans le champ Value, indiquez http://192.168.0.154:9000

Recommencez l'opération pour le user, avec les valeurs suivantes :

- Dans le champ Key, indiquez alerts.endpoint.thehive.user
- Dans le champ Value, indiquez api

Les règles d'import :

Toujours dans Patrowl, il va falloir établir des règles pour l'export des findings vers TheHive : Rendez vous dans le menu « Rules » / « List rules » puis remplissez tel que ci-dessous :



Vous l'aurez compris, il faut répéter l'opération pour les alertes low, medium, etc.. C'est un peu pénible mais à ne faire qu'une fois !

Conclusion

Une telle plateforme peut sembler « usine à gaz ».. Il faut plutôt y voir un assemblage d'outils dédiés dans la pure philosophie KISS. Il ne faudra pas sous-estimer la charge de travail de maintenance, les mises à jour, etc.. mais ces opérations sont grandement simplifiées par l'usage de docker.

Pendant la phase de migration il faudra redéfinir dans Patrowl l'ensemble des cibles de scan qui étaient auparavant éparpillées dans différents outils. Des modules d'import CSV sont disponibles et si vous changez de scanner demain, vous n'aurez plus cette manipulation à faire :-)

Si vous ne disposez pas de suffisamment de main d'œuvre en administration système, n'hésitez pas à vous faire accompagner par l'éditeur ou un prestataire pour le déploiement ! Ce serait dommage de passer à côté de cette solution : le gain de temps sur la gestion des incidents vaut largement l'investissement initial.

Autres avantages de cette solution : les API REST, les nombreuses perspectives d'interconnexion (IDS, Cortex,…) l'usage de produits français (TheHive, Patrowl) ou européens (OpenVAS) et tous OpenSource. La jeunesse du projet Patrowl ne semble pas un problème car il est déjà très stable et surtout soutenu et utilisé par une très grande organisation financière.

Pour résumer, **il est rare qu'une solution technique réponde aussi bien à mes problématiques SSI**, c'est ce qui m'a décidé à écrire cette documentation afin d'en faire profiter mes collègues RSSI ! Je précise également que je n'ai aucune action chez les éditeurs cités ici :-)

Annexes:

Script de démarrage de la plateforme

Par défaut, seul le docker postgres de Patrowl est démarré à l'allumage de la VM. Voici un script d'exemple pour démarrer les autres dockers :

```
#!/bin/sh

echo "Demarrage scanner openvas"

/usr/bin/docker run -d --rm -p 4443:443 -p 9390:9390 -e PUBLIC_HOSTNAME=192.168.0.154 -v /data/openvas/:/var/lib/openvas/mgr/ --name openvas mikesplain/openvas:9



echo "Demarrage engine openvas"

/usr/bin/docker run -d --rm -p 5116:5016 -v /sources/PatrowlEngines/engines/openvas/openvas.json:/opt/patrowl-engines/openvas/openvas.json --name="openvas-engine" patrowl-openvas



echo "Demarrage engine nmap"

/usr/bin/docker run -d --rm -p 5101:5001 --name="nmap-engine" patrowl-nmap



echo "Demarrage scanner arachni"

/usr/bin/docker run -d --rm -p 7331:7331 -p 9292:9292 --name arachni arachni/arachni:latest



echo "Demarrage engine arachni"

/usr/bin/docker run -d --rm -p 5105:5005 -v /sources/PatrowlEngines/engines/arachni/arachni.json:/opt/patrowl-engines/arachni/arachni.json --name="arachni-engine" patrowl-arachni



echo "Demarrage TheHive"

/usr/bin/docker-compose -f /sources/thehive/docker-compose.yml up -d



echo "Demarrage PatrowlManager"

/usr/bin/docker-compose -f /sources/PatrowlManager/docker-compose.yml up -d
```

Les URLs des applications :

OpenVAS : https://192.168.0.154:4443/

Arachni : http://192.168.0.154:9292

PatrowlManager : http://192.168.0.154:8083/

TheHive : http://192.168.0.154:9000/index.html#!/cases

Les sites des différents projets :

https://www.patrowl.io

https://thehive-project.org/

https://www.openvas.org/

https://www.arachni-scanner.com/