

DE LA RECHERCHE À L'INDUSTRIE



## Surveillance et Gestion d'incident

Commissariat à l'Énergie Atomique  
et aux Énergies Alternatives

26 février 2020

- 1 Collecte de logs**
- 2 Gestion des incidents**
- 3 Bonus**

- 1 Collecte de logs**
- 2 Gestion des incidents**
- 3 Bonus**

- Pour surveiller un parc informatique on peut :
  - faire de la **capture réseau** en plaçant des sondes et en analysant les flux de données en certains points ;
    - > voir le cours dédié
  - faire de la collecte d'**événements des systèmes** ;
  - faire de la collecte d'**événements des applications**.

- Ces logs nous permettront :

- de faire de la **détection d'intrusion**;
  - de nous aider lors de l'**analyse d'incident**;
  - de tracer l'activité des administrateurs et des utilisateurs.
    - > Ils vont donc permettre de répondre au problème d'**imputabilité**;
    - Imputabilité : possibilité de considérer une personne, du point de vue matériel et du point de vue moral, comme l'auteur d'une infraction.

- La surveillance d'un parc informatique est complexe :
  - besoin de beaucoup de connaissances et d'expériences (en *pentest* notamment) pour mettre en place une "bonne" surveillance, adaptée à l'environnement surveillé ;
  - en grande partie un empilement d'**heuristiques** ;
  - certaines solutions commerciales prétendent être parfaites mais on peut en douter.

- On peut collecter différents types de logs au niveau système et au niveau applicatif :
  - Logs système :
    - processus;
    - programmes exécutés;
    - connexions de périphériques USB;
    - sessions d'utilisateurs.
  - Logs applicatif :
    - serveur web;
    - serveur SSH;
    - anti-virus.
- Les logs générés par le système et les applications par défaut peuvent ne pas être suffisants :
  - on peut demander au système de générer plus de logs (*auditd* et *Sysmon*);
  - pour les applications le proposant, on peut augmenter la verbosité des logs (mode debug).

- On peut collecter les logs de différents types de système :

- Windows ;
- Linux ;
- Systèmes industriels ;
- ...

1. Collecte

2. Séquestration

3. Extraction

4. Indexation

5. Analyse

## 1. Collecte

Via des agents locaux sur les systèmes, les logs sont envoyés à des machines de collecte.

- à noter, ces machines de collecte sont les seules machines exposées au parc que l'on surveille, tout le reste se fait dans une bulle dédiée à la surveillance ;
- les captures réseau peuvent être agrégées ici pour être traitées de la même façon que les logs systèmes et applicatifs.

## 2. Séquestration

## 3. Extraction

## 4. Indexation

## 5. Analyse

## 1. Collecte

### 2. Séquestration

- La disponibilité et l'intégrité des logs sont critiques ;
  - si des logs sont manquants on peut rater des incidents ou il peut nous manquer des éléments pour comprendre un incident ;
  - si les logs sont altérés on peut rater un incident ou analyser de façon erronée un incident.
- On peut directement analyser les logs (sans mettre en place les étapes suivantes) à l'aide d'outils simples comme *grep* ;
- C'est à cette étape que l'on définit la politique d'archivage des logs (combien de temps doit-on les garder tel quel ? combien de temps sous forme compressée ?).

### 3. Extraction

### 4. Indexation

## 1. Collecte

## 2. Séquestre

## 3. Extraction

- Les logs :
  - contiennent des informations inutiles ;
  - sont hétérogènes ;
  - sont difficilement exploitables tel quel.
- On va donc extraire les informations qui nous intéressent, les formater et les normaliser.

## 4. Indexation

## 5. Analyse

## 1. Collecte

## 2. Séquestration

## 3. Extraction

## 4. Indexation

Les données extraites sont ensuite indexées pour faciliter leur consultation.

## 5. Analyse

## 2. Séquestration

## 3. Extraction

## 4. Indexation

## 5. Analyse

On peut ensuite analyser les logs manuellement et automatiquement. Dans cette étape on met en place :

- les outils de levée d'alerte ;
- les outils de visualisation.

On pourra y faire de la **corrélation d'événements**. Par exemple :

- réception d'un mail ;
- ouverture de la pièce jointe ;
- tentatives de mouvement latéral de l'utilisateur.

1. Collecte

2. Séquestration

3. Extraction

4. Indexation

5. Analyse

- Collecte :

- sous Unix :
  - syslogd;
  - syslog-*ng*;
  - rsyslog.
- sous Windows :
  - *Event log forwarding* (fonctionnalité native);
  - nxlog;
  - Winlogbeat.
- transfert en UDP ou en TCP ?
- utilisation de chiffrement pour l'intégrité et pour l'authentification.

- Il peut être nécessaire d'avoir **plus de logs des systèmes** que ceux fournis par défaut;
- Sous Linux, on peut utiliser `auditd` :
  - permet notamment d'obtenir des logs :
    - détaillant les commandes exécutées;
    - d'accès aux fichiers;
    - pour les appels système considérés suspects.
- Sous Windows, on peut utiliser Sysmon :
  - outil de la suite *Sysinternals*;
  - permet notamment d'obtenir des logs :
    - plus complet sur la création de processus (processus parent, paramètres...);
    - de modification de la base de registre;
    - de modification de fichiers;
    - de connexions réseau;
    - d'accès à la mémoire de processus.

1. Collecte

2. Séquestration

3. Extraction

4. Indexation

5. Analyse

**■ Séquestration :**

- pas d'outils particuliers ;
- juste un système de fichiers et des disques (et du RAID) ;
- dans l'idéal couplé avec un système de contrôle d'intégrité.

1. Collecte

2. Séquestration

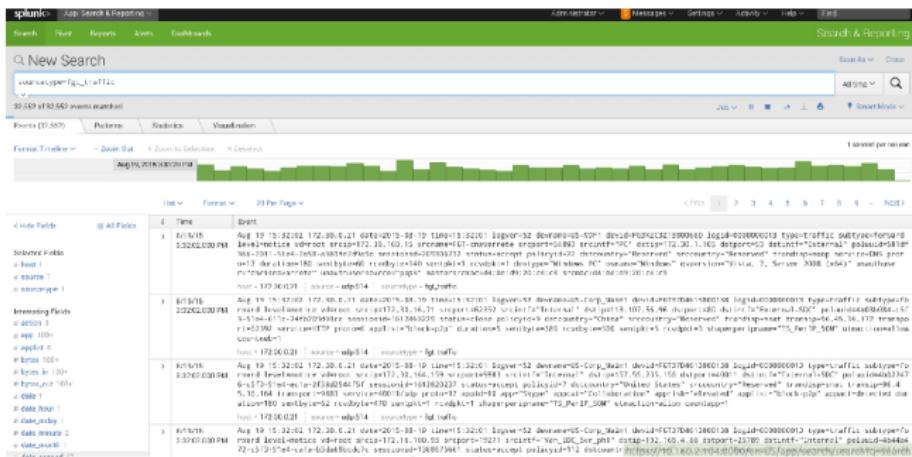
3. Extraction

4. Indexation

5. Analyse

- Extraction/Indexation/Analyse :
  - ELK ;
    - Extraction : Logstash ;
    - Indexation : Elasticsearch ;
    - Analyse : Elasticsearch et Kibana.
  - Splunk.

- Splunk permet de faire la même chose que ELK ;
  - Mais avec un seul **outil tout-en-un** ;
  - Splunk est moins coûteux en temps que ELK ;
  - Mais l'outil est **payant et n'est pas open source**.



+ Avantages :

- *Open source*
- Gratuit
- Pas de solution *open source* aussi solide

- Inconvénients :

- Coûteux en temps (mise en place et maintenance)
- Coûteux en matériel

- Une architecture possible de ELK :

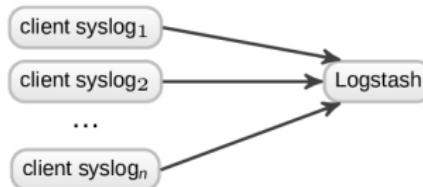
client syslog<sub>1</sub>

client syslog<sub>2</sub>

...

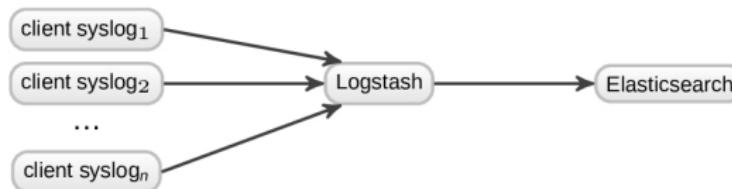
client syslog<sub>n</sub>

- Une architecture possible de ELK :



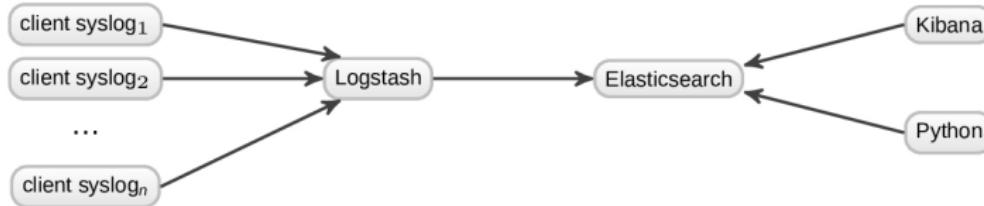
- > Les clients syslog envoient leurs logs à Logstash qui les *parse*nt

- Une architecture possible de ELK :

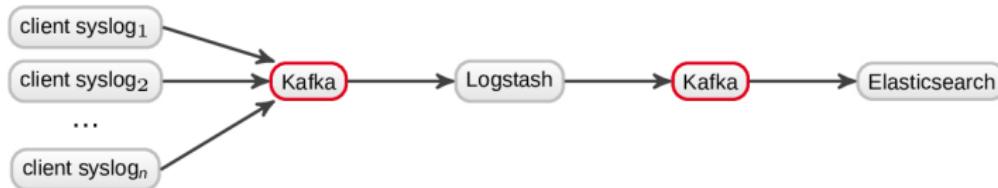


- > Après avoir *parsed* les logs, Logstash les insère dans Elastic

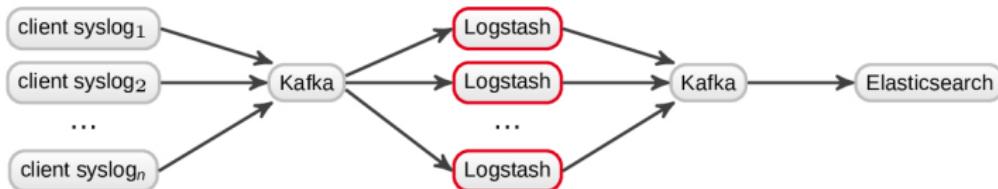
- Une architecture possible de ELK :



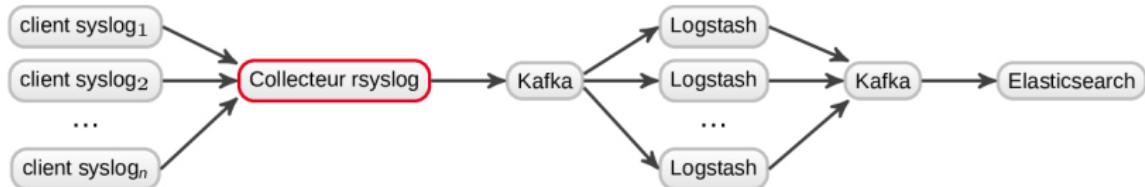
- > On peut ensuite visualiser les logs depuis Kibana ou encore effectuer des requêtes directement à l'API Elasticsearch (en python par exemple)



- > Kafka peut être utilisé comme tampon



- > On peut avoir une instance de Logstash par type de logs pour faciliter la maintenance



Il y a 3 composantes dans la configuration de Logstash :

- *Input* :
- *Filter* :
- *Output* :

```
input {  
    tcp {  
        port => 514  
        codec => "json"  
        type => "syslog"  
    }  
}  
  
filter {  
    ...  
}  
  
output {  
    elasticsearch {  
        hosts => localhost  
    }  
}
```

Il y a 3 composantes dans la configuration de Logstash :

- *Input* :

- définit les sources de logs;
- beaucoup de sources possibles :
  - stdin;
  - TCP ou UDP sur un port;
  - Kafka;
  - ...

- *Filter* :

- *Output* :

```
input {  
    tcp {  
        port => 514  
        codec => "json"  
        type => "syslog"  
    }  
}  
  
filter {  
    ...  
}  
  
output {  
    elasticsearch {  
        hosts => localhost  
    }  
}
```

Il y a 3 composantes dans la configuration de Logstash :

- *Input* :
- *Filter* :
  - définit comment les logs sont filtrés et formatés;
  - les logs sont traités un par un :
    - on ne peut pas faire de la corrélation entre les logs.
- *Output* :

```
input {  
    tcp {  
        port => 514  
        codec => "json"  
        type => "syslog"  
    }  
}  
  
filter {  
    ...  
}  
  
output {  
    elasticsearch {  
        hosts => localhost  
    }  
}
```

Il y a 3 composantes dans la configuration de Logstash :

- *Input* :
- *Filter* :
- *Output* :
  - définit où les logs sont envoyés en sortie ;
  - beaucoup de sorties possibles :
    - stdout;
    - Elasticsearch;
    - Kafka;
    - ...

```
input {  
  tcp {  
    port => 514  
    codec => "json"  
    type => "syslog"  
  }  
}  
  
filter {  
  ...  
}  
  
output {  
  elasticsearch {  
    hosts => localhost  
  }  
}
```

- Dans la partie ***filter***, les logs sont :
  - sous forme de **JSON**;
  - traités **un par un**.
- On peut y utiliser plusieurs fonctions :
  - `mutate`;
  - `grok`;
  - `clone`;
  - ...

- Pour extraire de l'information d'un log on peut utiliser le filtre **grok** ;
- *grok* permet :
  - de comparer le log à une **expression régulière** ;
  - de choisir d'en extraire des parties ;
- Les parties extraites sont placées dans de **nouveaux champs du JSON** ;
- Logstash propose des *patterns* pour nous simplifier la tâche comme :
  - `%{IP}` pour *matcher* une adresse IP ;
  - `%{INT}` pour *matcher* un entier.

- Si l'on prend le log suivant :

```
Feb 13 11:20:32 vmserver sshd[2847]: Accepted password for user
from 192.168.56.1 port 35188 ssh2
```

- On pourrait ajouter le grok suivant au fichier de configuration de Logstash :

```
filter {
  grok {
    match => { "message" => "Accepted password for %{DATA} from
      %{IP} port %{INT} ssh2" }
  }
}
```

- Si l'on prend le log suivant :

```
Feb 13 11:20:32 vmserver sshd[2847]: Accepted password for user
from 192.168.56.1 port 35188 ssh2
```

- On pourrait ajouter le grok suivant au fichier de configuration de Logstash :

```
filter {
  grok {
    match => { "message" => "Accepted password for %{DATA:user}
      from %{IP:ip} port %{INT:port} ssh2" }
  }
}
```

- Avec ce grok, on extrait trois informations : le nom de l'utilisateur (*user*), l'IP source (*ip*) et le port source (*port*).

- Elasticsearch est une **base de données** orientée document ;
- Basé sur Lucene ;
  - un moteur de recherche de données développé par Apache ;
- Les requêtes se font via HTTP (interface REST) :
  - requête PUT pour ajouter ou modifier des données ;
  - requête GET pour récupérer des données ;
  - les données échangées sont au format JSON.

**Cluster**

Ensemble de *nodes* qui communiquent entre elles.

**Node**

Instance Elasticsearch composant un *cluster*.

**Index**

Ensemble de documents qui présentent des caractéristiques semblables.

**Shard**

Fragment d'index permettant de répartir un index sur plusieurs *nodes*.

**Document**

Unité basique d'information que l'on peut indexer.

- Pour requêter Elasticsearch on peut utiliser curl :

- pour lister les indexés présents :

```
curl 'localhost:9200/_cat/indices?v'
```

- pour lister toutes les entrées de l'index "logstash-2019.02.26" :

```
curl 'localhost:9200/logstash-2019.02.26/_search?pretty' -d '  
{  
    "query": { "match_all": {} },  
}'
```

- pour lister les entrées provenant de SSHD de l'index "logstash-2019.02.26" :

```
curl 'localhost:9200/logstash-2019.02.26/_search?pretty' -d '  
{  
    "query": { "match": { "programname": "sshd" } },  
}'
```

- Pour requêter Elasticsearch on peut utiliser du python :

```
from elasticsearch import Elasticsearch  
es = Elasticsearch()
```

- pour lister les indexes présents :

```
es.indices.get_aliases().keys()
```

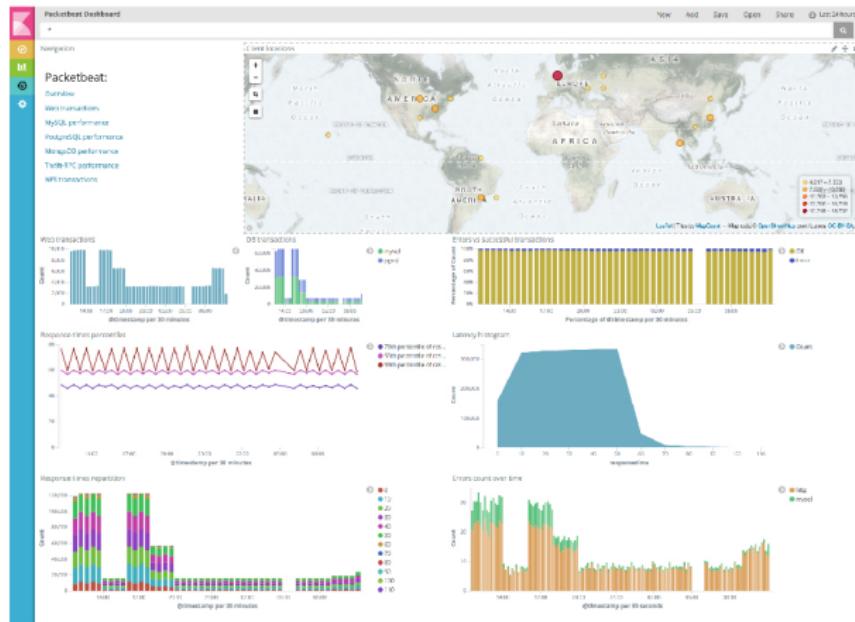
- pour lister toutes les entrées de l'index "logstash-2019.02.26" :

```
es.search(index="logstash-2019.02.26", body={"query":  
{"match_all": {}}})
```

- pour lister les entrées provenant de SSHD de l'index "logstash-2019.02.26" :

```
es.search(index="logstash-2019.02.26", body={"query":  
{"match": { "programname": "sshd" }}})
```

- Kibana va nous permettre de visualiser les données stockées dans Elasticsearch;
- Les pages de visualisation que l'on va créer seront appelées des *dashboards*.



- Les alertes sont générées au cours de la phase d'analyse ;
- Dans le cas où l'on a opté pour ELK les alertes peuvent être générées :
  - par Logstash (*email* ou *exec*) ;
  - par des requêtes Elasticsearch (lancées via *cron* par exemple).
    - > permet d'apporter plus d'intelligence dans les règles de génération d'alerte qu'avec Logstash ;
    - les alertes Logstash se basent sur un seul log.
- Après avoir généré l'alerte il faut la remonter aux personnes compétentes :
  - la solution basique consiste à envoyer des mails ;
    - mais cette solution présente un manque de centralisation, de possibilité de suivi...
  - une solution consiste à détourner les produits de *bug tracking* comme Flyspray pour les utiliser pour la gestion d'alerte.

- Alertes levées à l'aide d'un seul log :
  - connexion d'un appareil USB non autorisé ;
  - utilisation d'un outil d'attaque (exemple : Mimikatz) ;
  - détection d'un virus par un anti-virus.
- Alertes levées en corrélant plusieurs logs :
  - mouvements latéraux d'un compte administrateur ;
  - synchronisation suspecte de *Domain Controllers* (DCSync) ;
  - *upload* d'un volume important de données ;
  - important nombre d'échecs d'authentification.

- SOC :
  - *Security Operations Center* ;
  - lieu où est surveillé un site ;
  - il s'agit donc du lieu où sont analysés les logs.
- SIEM :
  - *Security Information and Events Management (System)* ;
  - il s'agit de produits qui prennent en charge les cinqs étapes dont nous avons parlé (collecte, séquestration, extraction, indexation, analyse) ;
  - exemples : OSSEC, QRadar ;
  - quand on met en place une architecture syslog/nxlog + ELK, on met en place un SIEM.
- CERT :
  - *Computer Emergency Response Team* ;
  - établissement et maintenance d'une base de donnée des vulnérabilités ;
  - diffusion de précautions à prendre pour minimiser les risques d'incident ou leurs conséquences ;
  - les informations sont généralement accessibles à tous ;
  - CERT-EU, CERT-FR, CERT-Orange...

## ■ SOC :

- *Security Operations Center* ;
- lieu où est surveillé un site ;
- il s'agit donc du lieu où sont analysés les logs.

### **Atos lance le premier Security Operations Center (SOC) prescriptif au monde avec réponse automatisée**

Avec un temps de détection et réponse inédit, le SOC d'Atos contrecarrer les cyberattaques avant même qu'elles ne se produisent

Paris, Bruxelles, le 4 juillet 2017

Atos, leader international de la transformation digitale, lance aujourd'hui le premier Security Operations Center (SOC) prescriptif au monde avec réponse automatisée. Associantes compétences d'analyse Big Data d'Atos et la puissance des serveurs Bullion, la nouvelle solution de sécurité permet aux clients de prédire les cybermenaces avant même qu'elles ne se produisent. L'amélioration du temps de détection et de neutralisation est sans commune mesure avec les offres actuellement sur le marché.

## ■ SIEM :

- *Security Information and Events Management (System)* ;
- il s'agit de produits qui prennent en charge les cinqs étapes dont nous avons parlé (collecte, séquestration, extraction, indexation, analyse) ;
- exemples : OSSEC, QRadar ;
- quand on met en place une architecture syslog/nxlog + ELK, on met en place un SIEM.

## ■ CERT :

- *Computer Emergency Response Team* ;
- établissement et maintenance d'une base de donnée des vulnérabilités ;
- diffusion de précautions à prendre pour minimiser les risques d'incident ou leurs

- SOC :
  - *Security Operations Center* ;
  - lieu où est surveillé un site ;
  - il s'agit donc du lieu où sont analysés les logs.
- SIEM :
  - *Security Information and Events Management (System)* ;
  - il s'agit de produits qui prennent en charge les cinqs étapes dont nous avons parlé (collecte, séquestration, extraction, indexation, analyse) ;
  - exemples : OSSEC, QRadar ;
  - quand on met en place une architecture syslog/nxlog + ELK, on met en place un SIEM.
- CERT :
  - *Computer Emergency Response Team* ;
  - établissement et maintenance d'une base de donnée des vulnérabilités ;
  - diffusion de précautions à prendre pour minimiser les risques d'incident ou leurs conséquences ;
  - les informations sont généralement accessibles à tous ;
  - CERT-EU, CERT-FR, CERT-Orange...

- Beaucoup d'entreprises tentent d'utiliser le **machine learning** pour aider à la surveillance des systèmes ;
- Le problème de l'analyse de logs semble adapté au *machine learning* :
  - grande quantité de données difficilement analysable humainement ;
  - difficulté pour mettre en place des moyens de détection automatiques triviaux ;
- Une approche : faire apprendre un profil de logs sain et détecter toutes dérives par rapport à ce profil ;
- Il s'agit d'un sujet en cours de développement.

- 1** Collecte de logs
- 2** Gestion des incidents
- 3** Bonus

# Étapes de la gestion d'un incident

0. Détection d'incident

1. Qualification / Identification des ressources impactées

2. Confinement de l'attaque

3. Analyse *forensic*

4. Retour à un état sain

## 0. Détection d'incident

L'incident peut être remonté :

- par notre système de surveillance (s'il est en mesure de le détecter et s'il a été configuré pour le détecter) ;
- par une entité externe qui aurait repéré par exemple une exfiltration de données liée à notre entité ;
- par un utilisateur.

## 1. Qualification / Identification des ressources impactées

## 2. Confinement de l'attaque

## 3. Analyse *forensic*

## 4. Retour à un état sain

## 0. Détection d'incident

### 1. Qualification / Identification des ressources impactées

- L'incident est généré par un utilisateur malveillant ? par un virus ?
- L'incident est-il un faux positif (mauvaise manipulation d'un administrateur) ?
- Des serveurs sont impactés ? Quelle est leur criticité (serveur web frontal, *Domain Controller*...) ?
- Des postes sont impactés ? Utilisateur ou administrateur ?
- Y a-t-il une communication avec l'extérieur ? Une exfiltration de données ?

### 2. Confinement de l'attaque

### 3. Analyse *forensic*

### 4. Retour à un état sain

## 0. Détection d'incident

## 1. Qualification / Identification des ressources impactées

## 2. Confinement de l'attaque

Suivant la nature de l'incident et le périmètre impacté on peut :

- verrouiller des comptes utilisateur ou administrateur ;
- *black-list*er des noms de domaine ou des adresses IP ;
- bloquer des flux réseau ;
- éteindre des machines.

## 3. Analyse *forensic*

## 4. Retour à un état sain

# Étapes de la gestion d'un incident

0. Détection d'incident

1. Qualification / Identification des ressources impactées

2. Confinement de l'attaque

3. Analyse *forensic*

- voir le cours dédié !
- analyse des logs et des captures réseau pour comprendre :
  - le chemin de l'attaque ;
  - les failles qui ont amené à l'incident ;
  - le périmètre impacté par l'incident.

4. Retour à un état sain

## 1. Qualification / Identification des ressources impactées

## 2. Confinement de l'attaque

## 3. Analyse *forensic*

## 4. Retour à un état sain

- D'après les éléments récupérés jusqu'ici, on peut déterminer ce qui va devoir être nettoyé.
- Il est particulièrement important de bien définir le périmètre de la compromission ;
  - Cas d'une infection par un virus : si l'on réinstalle seulement le poste à l'origine de l'infection, on laisse peut être des postes et des serveurs porteurs du virus :
    - > L'infection continue.
- Après le retour à l'état sain, il faut définir les mesures qui peuvent être prises pour empêcher l'incident de se reproduire.

## Marqueurs de compromission

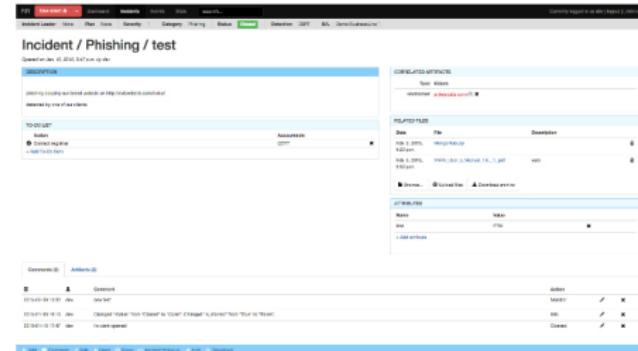
Les marqueurs de compromission (*Indicator of Compromise*, IOC) sont des éléments réseau, système ou applicatif qui indiquent une compromission.

Exemples :

- URL d'un *command and control* ;
  - hash d'un virus.
- 
- Lors de la réponse à incident, il arrive souvent qu'on trouve des marqueurs de compromission ;
  - Il peut être intéressant de les partager avec d'autres entités pour qu'elles puissent bloquer les attaques ;
  - Outils permettant le partage entre organisations :
    - MIPS, Yeti, CRITs ou encore CIF.
  - Il faut donc que notre entité accepte de partager des éléments de ces incidents (qui peuvent contenir des informations sensibles) ;
  - Ce partage lève des problèmes de confiances (doit on bloquer tout les IOC qui sont partagés ? Qu'arrive-t-il en cas d'erreur ?).

- Pour aider à la gestion d'incident, on peut s'appuyer sur des outils que l'on appelle IRP (*Incident Response Platform*) ou SIRP (*Security Incident Response Platform*) ;
  - Il existe des produits non *open source* comme :
    - *Resilient* de IBM ;
    - *Archer Security Operations* de RSA.
  - Et des outils *open source* comme FIR (*Fast Incident Response*) :

- disponible sur Github ;
  - permet :
    - de cadrer la gestion des incidents ;
    - d'archiver des informations liées aux incidents ;
    - de faciliter le travail collaboratif ;
    - de faire de la corrélation avec d'anciens incidents.



- 1** Collecte de logs
- 2** Gestion des incidents
- 3** Bonus

## Définition

Un *honeypot* est un système exposé à de potentiels attaquants qui a pour but d'être scanné, attaqué et compromis. Idéalement les attaquants ne doivent pas avoir conscience qu'il s'agit d'un *honeypot*.

- Intérêts :
  - découvrir de nouvelles vulnérabilités, exploits ou virus ;
    - permettra donc d'identifier de nouveaux IOC pour faire de la levée d'alerte.
  - occuper un pirate ;
  - toute interaction avec un *honeypot* est suspecte.
    - réduction des faux positifs.
- Important : Toutes les actions du pirate doivent être tracées et enregistrées.
  - > mise en place d'un "SIEM" pour le *honeypot*.

- Difficultés organisationnelles :
  - Legalité du procédé ? (Incitation?) ;
  - En cas de compromission, risque que le *honeypot* devienne un point de rebond pour de nouvelles attaques ;
  - Un système complet et complexe de plus à gérer ;
- Difficultés techniques :
  - Analyser l'activité peut être complexe et demander une expertise ;
  - Spécificité de la maintenance d'un tel système : doit à la fois paraître vulnérable et être assez sécurisé pour éviter une véritable compromission.
- Autres :
  - Surenchère vis-à-vis de l'attaquant : déployer des *honeypots* peut inciter les pirates à s'intéresser à l'ensemble du réseau et donc avoir l'effet inverse de celui souhaité !

- Interaction :
  - Faible interaction :
    - Simulation de services sans réel système sous-jacent ;
    - Relative simplicité au niveau de l'émulation.
  - Forte interaction :
    - Le *honeypot* représente un véritable système d'exploitation.
- Nature :
  - Physique :
    - Le *honeypot* est une machine réelle, munie d'outils adaptés pour tracer l'activité.
  - Virtuelle :
    - Tout le système est émulé (machine virtuelle par exemple).

- Il s'agit d'un type particulier de *honeypot* dont la structure a été travaillée et standardisée par le Honeynet Project ;
  - Ils reposent sur des *honeypots* à forte interaction ;
  - Il ne s'agit pas d'un produit, mais d'une proposition d'architecture.
- 
- Architecture réseau fortement contrôlée :
    - NIDS classiques ;
    - Analyse de flux ;
    - Tout trafic entrant ou sortant du honeynet est suspect ;
    - Il existe différentes générations de cette architecture.

- Il existe un grand nombre d'implémentations de *honeypot*. Une importante liste de ces implémentations :
  - > <https://github.com/paralax/awesome-honeypots>

- *Intrusion Prevention System* ;
- Les IDS sont capables de détecter une attaque ;
  - > Pourquoi ne pas en profiter pour la neutraliser ?

## Risques

- l'IDS ne sera plus **invisible** ;
- c'est une mauvaise idée en cas de **faux-positif**.

## Exemple

*fail2ban* :

- lit les logs de divers services (SSH, Apache, FTP) ;
- cherche des erreurs d'authentification répétées ;
- ajoute une règle iptables pour bannir l'adresse IP de la source.

Commissariat à l'énergie atomique et aux énergies alternatives  
Centre de Bruyères-le-Châtel | 91297 Arpajon Cedex  
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 40 00  
Établissement public à caractère industriel et commercial  
RCS Paris B 775 685 019

CEA