4 À L'INTÉRIEUR DE LA FORTERESSE ASSIÉGÉE

À découvrir dans cette partie...

4.1 Malcom – the MALware COMmunication analyzer

Êtes-vous la victime d'une attaque ciblée ? Découvrez comment utiliser l'outil Malcom pour croiser les indicateurs réseaux d'un malware avec ceux des menaces connues. p. 106

4.2 Red Team : le point de vue de l'audité et du commanditaire

Comment optimiser la préparation, le suivi et la restitution d'une campagne de tests Red Team ? Cet article vous présente le point de vue d'un RSSI. p. 118

4 À L'INTÉRIEUR DE LA FORTERESSE ASSIÉGÉE

MALCOM – THE MALWARE COMMUNICATION ANALYZER

Thomas Chopitea

L'analyse de trafic réseau est une des techniques les plus populaires d'analyse dynamique de malwares. Dans le cadre de la réponse à incident, elle permet rapidement d'identifier les points de commande et contrôle (C2) et de produire des marqueurs ou indicateurs de compromission (« Indicators of Compromise », ou IOC) pertinents pour identifier des hôtes infectés. Les marqueurs réseau sont aussi un très bon point de pivot pour étendre sa connaissance sur l'adversaire : infrastructure, autres malwares, etc. Cependant, les outils classiques d'analyse réseau ne permettent pas de partager ses trouvailles avec d'autres chercheurs, obligeant la communauté « blue team » à partir de zéro lors de l'analyse. Nous allons voir comment Malcom essaye de répondre à ces besoins de partage et en quoi il diffère des outils d'analyse réseau classiques.

1. MALCOM IN THE MIDDLE

Malcom est un outil qui se trouve au croisement de deux mondes : l'analyse réseau classique et le « Threat Intelligence ». Malcom est né d'un besoin très opérationnel : croiser les indicateurs réseaux intéressants issus d'une analyse dynamique du malware avec la malveillance connue sur Internet. Le processus manuel de croisement ressemble vaguement à ça :

1. On rencontre un échantillon de malware (*exploit-kit*, remontée utilisateur, partage sur un forum) ;
2. On lance le malware dans une *sandbox* ou une VM ;
3. On lance son outil d'analyse trafic réseau préféré :
   - ⇨ Wireshark (vue très bas-niveau) ;
   - ⇨ Burp (vue très (trop ?) haut niveau) ;
   - ⇨ tcpdump (pour les warriors !) ;
4. On filtre et extrait les marqueurs réseaux des résultats. La méthode varie grandement en fonction de l'outil utilisé pour la capture.

Ces étapes peuvent être simplifiées à l'aide d'un logiciel de sandbox, comme Cuckoo Sandbox, qui sortira un résumé des communications réseau détectées pendant l'exécution.

À ce stade, on se retrouve avec quelques artefacts : on ne peut pas vraiment parler d'IOC avant d'avoir fait une analyse préalable souvent très rapide, les humains étant particulièrement doués pour faire la différence entre **google.com** et **lfzlijqsxcuwgcamrylwsfamz.com**. On a entre nos mains quelques noms de domaines, URL, et adresses IP, que nous avons mis dans la case « potentiellement malveillant ». C'est maintenant que nous allons mettre notre plus belle peau d'ours sur le dos et faire un peu de chamanisme (ou de Threat Intelligence, pour les moins initiés). Pour chaque type de marqueur, nous allons donc nous pencher sur les services en ligne qui vont bien : DomainTools, Passive Total, robtex, VirusTotal, TotalHash, et... Google, qui va sûrement nous montrer d'autres analyses ou des billets de blog où les domaines ou adresses IP ont déjà été vus. On a trouvé une analyse datant d'il y a trois jours sur un blog qui détaille l'analyse d'un malware n'ayant pas le même hash que le nôtre, mais qui contacte le même nom de domaine, et qui requête la même URL. Notre esprit aiguisé de cyber-chamane arrive à faire le rapprochement et en déduit que nous faisons face à la même menace : **un Zbot**. Rien à signaler donc.

Entre le moment où on a rencontré le malware et le moment où on s'est dit que ce n'était pas un implant de la NSA, il s'est bien passé 30 minutes. Et nous avons décrit le cas où une analyse avait déjà été publiée et où notre VM n'a pas voulu se mettre à jour toute seule, polluant l'analyse.

Malcom vise justement à faire une grosse partie de ce travail d'extraction et de recherche pour nous. Il va s'appuyer sur des *feeds*, pour construire une base de malveillance la plus contextualisée possible (cela dépend évidemment de la qualité du feed - nous en parlerons plus tard). Lors d'une analyse réseau, les éléments extraits seront automatiquement croisés avec cette liste de malveillance et l'analyste pourra instantanément savoir si un des marqueurs réseaux est connu ou non pour être malveillant et pour quelle raison.

On peut voir Malcom comme un Wireshark plus haut niveau, qui ne montre que les aspects du trafic qui intéresseront potentiellement l'analyste (avec ou sans visualisation). Avec les modules (dont nous détaillerons le fonctionnement plus tard), on peut aussi l'assimiler à un genre de Volatility pour captures de trafic réseau. Finalement, et presque par « effet de bord », c'est aussi une bonne base de données de malveillance.

2. ARCHITECTURE

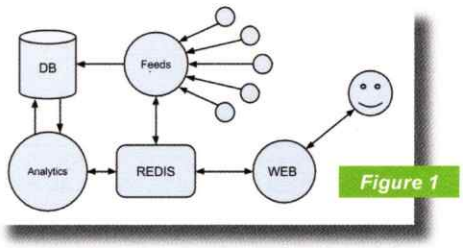


Figure 1

L'architecture de Malcom peut se diviser en trois grosses parties : la partie ***feeds***, la partie ***analytics***, et la partie ***web***. Voici un diagramme de l'architecture à l'heure où ces mots sont écrits (Figure 1).

Les trois pôles, ***feeds***, ***analytics***, et ***web***, communiquent entre eux via une base Redis. Toute interaction humaine avec Malcom est faite via le pôle web. La logique de Malcom est entièrement codée en Python 2.7. La base de données est MongoDB, donc non relationnelle. Le *frontend* web est exposé via le framework web Flask.

2.1 Feeds - l'alimentation de Malcom

Le pôle feeds est celui qui est responsable d'alimenter Malcom en information. Chaque feed est ordonnancé par un moteur de feeds, qui décide quand un feed doit s'exécuter (en fonction de la configuration de ce dernier). Concrètement, un feed est un objet Python qui définit une source de données et une méthode de traitement des informations reçues. Nous verrons l'architecture précise d'un feed plus tard.

Les feeds peuvent récupérer des informations de sources publiques, en web, comme **ZeusTracker**, ou de sources privées. Vu qu'on exécute du code Python, les feeds ne sont pas limités à un format d'entrée particulier. Aujourd'hui, Malcom dispose de *helpers* pour les formats CSV et XML, mais il est tout à fait possible de l'adapter à un format spécifique (JSON, e-mail, syslog...).

Finalement, les feeds définissent aussi le *contexte* dans lequel l'information va vivre dans Malcom. C'est une caractéristique **totalement indispensable** de la Threat Intelligence : ça ne sert à rien d'avoir des informations en base si on ne sait pas à quoi elles correspondent. On privilégiera donc les feeds à fort contexte (« C2 Citadel », « droppee Gootkit »), plutôt que des feeds à contexte très vague (« scanner », « spambot », « malicious »). Ce contexte peut être répercuté dans les champs « evil » de chaque élément stocké en base, ou au niveau de « tags ».

2.2 Analytics - l'augmentation de l'information

L'information récupérée dans les feeds, est ensuite stockée dans la base de données sous forme « d'éléments ». Un élément peut être une adresse IP, une URL, un hostname, ou tout autre *datatype* qui soit défini dans le code python. Chaque élément dispose d'une fonction *analytics*, qui va générer d'autres éléments à partir des données intrinsèques à celui-ci. Ainsi, la fonction *analytics* d'un nom de domaine renverra les enregistrements NS, A, et MX associés. Un élément URL renverra son domaine. L'URI, schéma, etc. seront stockés dans l'élément lui-même. À titre d'exemple, il serait aussi possible de stocker le code HTTP renvoyé par cette URL à un instant *t*. Finalement, chaque élément dispose d'un « temps de rafraîchissement » prédéfini qui définit la fréquence à laquelle ses *analytics* sont exécutées.

Le moteur d'*analytics* boucle sur tous les éléments dont le « temps de rafraîchissement » a expiré, lance leur fonction *analytics*, et stocke les nouveaux éléments en base. Au démarrage de Malcom, il lance des *workers* qui vont monitorer une queue dans laquelle le moteur va pousser tous les éléments pertinents afin qu'ils soient analysés.

2.3 Web - le frontend

Finalement, le *frontend* web est celui qui va régir l'interaction entre l'utilisateur et Malcom. Aujourd'hui, le serveur web fait partie du cœur de Malcom, mais une grosse refonte est en cours pour pouvoir séparer l'interface web des serveurs de *feeding* et d'*analytics*. Cela permettra de lancer une instance de Malcom en mode client en local qui utilisera une instance de serveur distante.

Le serveur web donne accès à des fonctionnalités de recherche, de *sniffing* réseau, et permet aussi de voir l'état des feeds (Figure 2).

La page recherche amène à une page de résultats, où plus de détails sont données sur l'élément (Figure 3).

Nous pouvons voir que le domaine a été « vu » par le *feed* `MalwareDomainList`. Plus de détails sont donnés en dessous, comme la date à laquelle il a été vu pour la première fois, puis un lien où plus d'informations sont disponibles.

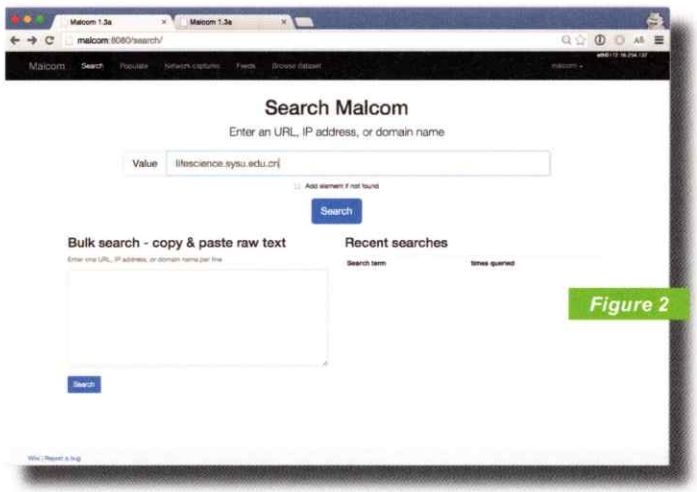


Figure 2

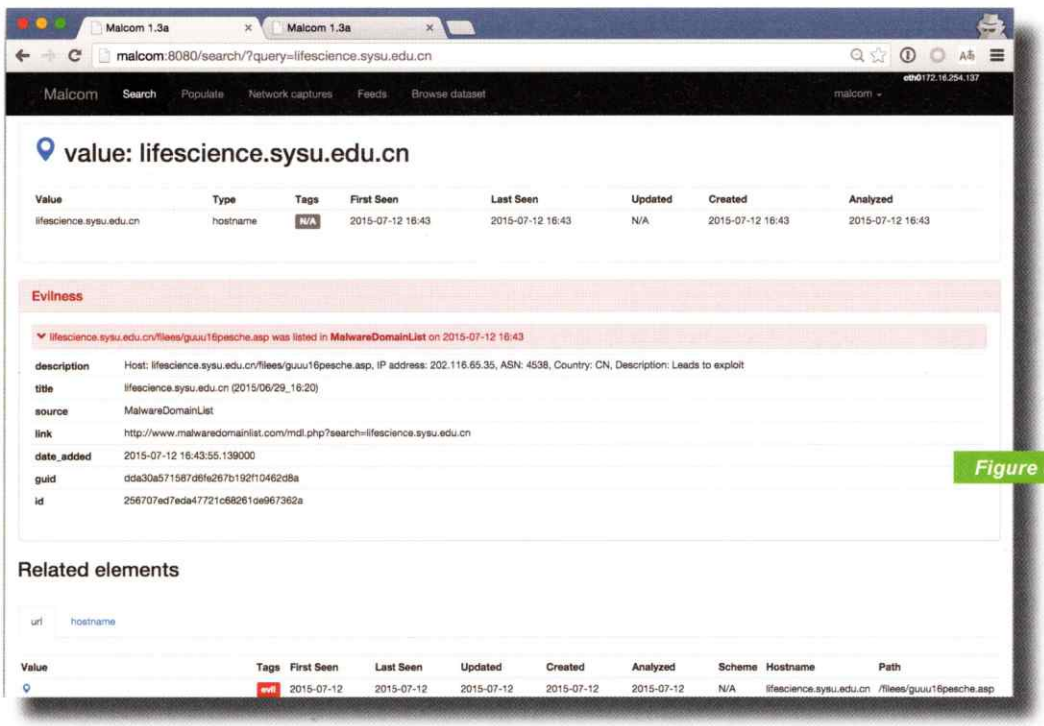


Figure 3

Le champ « description » contient aussi quelques informations, dont une information qui peut être très utile (même si un peu vague ici) : « Leads to exploit ».

3. CRÉATION D'UN FEED

Une des premières customisations qui peut être apportée à Malcom est d'y ajouter ses propres *feeds*. C'est là que vous en tirerez le plus de profit, car les *feeds* privées (vos journaux de pare-feu, une boite mail, un *syslog*) sont souvent plus intéressantes que les *feeds* disponibles publiquement.

Un tutoriel avec un exemple est disponible sur le Wiki public [**https://github.com/tomchop/malcom/wiki/Adding-feeds-to-Malcom**] du *repository*. Au risque de se répéter, nous allons revoir le processus ici.

3.1 __init__

Pour créer un feed, il suffit de créer un objet héritant de la classe générique **Feed**, et d'écrire trois fonctions: **__init__**, **update** et **analyze**. La fonction **__init__** a pour fonction d'initialiser le *feed* avec certains champs :

- **name** : Le nom du feed. Il doit avoir le même nom que la classe (cela sera automatisé à l'avenir) ;
- **source** : Une URL, un fichier, etc. Les *helpers* existants se servent de cet attribut pour localiser la source du feed ;
- **description** : La description du feed. Cet attribut doit répondre à la question « que représente le feed d'où cet élément est venu? ».

Il faut aussi spécifier le constructeur de l'objet parent **Feed** avec un paramètre **run_every** qui définira la fréquence à laquelle le feed sera rafraîchi.

Une fonction **init** se définit typiquement comme suit :

Fichier

```
def __init__(self, name):
 super(ZeusTrackerConfigs, self).__init__(name, run_every="1h")
 self.name = "ZeusTrackerConfigs"
 self.source = "https://zeustracker.abuse.ch/monitor.php?urlfeed=configs"
 self.description = "This feed shows the latest 50 ZeuS config URLs."
```

3.2 Update

La fonction **update** s'occupe de requêter la ressource d'appeler la fonction **analyze** sur chaque élément différent. Elle peut faire trois lignes si un *helper* est utilisé, elle en fera deux ou trois de plus sinon :

Fichier

```
def update(self):
 for dict in self.update_xml('item', ["title", "link", "description", "guid"]):
 self.analyze(dict)
```

Ici, le helper **update_xml** parse un XML ayant la forme suivante :

Fichier

```
<item>
<title>ideasengrupo.mx/meask/lite/file.php (2015-07-11)</title>
<link>https://zeustracker.abuse.ch/monitor.php?host=ideasengrupo.mx</link>
<description>URL: http://ideasengrupo.mx/meask/lite/file.php, status: online,
version: 2, MD5 hash: e75c8954a6a74599eb1960e1b5734825</description>
<guid>https://zeustracker.abuse.ch/monitor.php?host=ideasengrupo.mx&id=89d4464
7347081231a12f8775d818c4f</guid>
</item>
```

3.3 analyze

La fonction **analyze** va définir la logique même des informations qui sont en train d'être assimilées. Elle est donc cruciale, car elle va créer les éléments qui seront ajoutés à la base, mais surtout définir le contexte de ces informations parmi toutes les autres. Ce contexte d'informations est matérialisé sous la forme d'un dictionnaire python avec certaines caractéristiques, dont deux sont obligatoires :

⇨ **source** : le nom du *feed* source. Si des *helpers* sont utilisés dans la fonction **update**, ce champ est rempli automatiquement avec le nom du *feed*.

⇨ **description** : il s'agit d'expliquer pourquoi l'élément est malveillant. Les *feeds* donnent souvent des informations qui peuvent être utiles pour remplir ce champ.

Fichier

```
def analyze(self, dict):

 evil = dict

 url = Url(re.search("URL: (?P<url>\S+),", dict['description']).
group('url'))
 evil['id'] = md5.new(re.search(r"id=(?P<id>[a-f0-9]+)", dict['guid']).
group('id')).hexdigest()

 try:
  date_string = re.search(r"\((?P<date>[0-9\-]+)\)", dict['title']).
group('date')
  evil['date_added'] = datetime.datetime.strptime(date_string, "%Y-%m-%d")
 except AttributeError, e:
  print "Date not found!"

 url.add_evil(evil)
 url.seen(first=evil['date_added'])
 self.commit_to_db(url)
```

La fonction **add_evil** rajoute automatiquement le tag « evil » à l'élément. La vue graphique colore les éléments **evil** en rouge automatiquement, ce qui aide l'analyste à identifier rapidement les éléments malveillants. Attention : seul l'élément de base est tagué **evil** par défaut. Il est aussi possible de taguer d'autres éléments pertinents (comme son nom de domaine par exemple), mais cela doit être fait manuellement.

4. WORKFLOW TYPIQUE DE SNIFFING

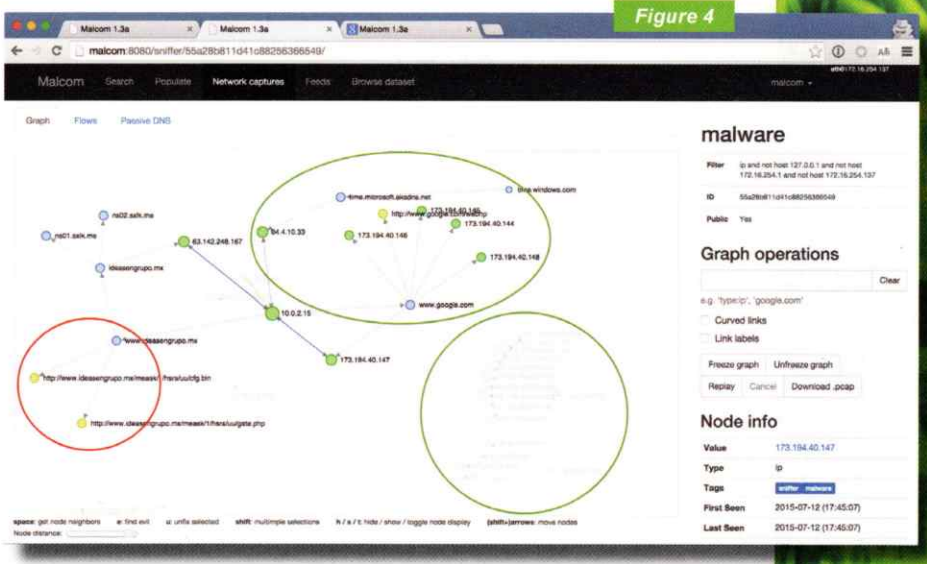
Maintenant que nous savons faire un *feed*, nous pouvons commencer à alimenter Malcom avec des données. Admettons que nous en ayons écrit quelques-unes, il est temps de se mettre à la pratique !

Nous tombons sur un fichier malveillant, **qwert.exe**, que nous trouvons sur le poste d'un utilisateur de notre réseau. Le but est de savoir ce qu'est ce malware, s'il représente une menace importante pour le SI (s'il est ciblé, par exemple), ou s'il s'agit juste d'un *banker* générique.

Nous faisons tourner le malware dans notre VM préférée, et nous utilisons Malcom soit en coupure (« default gateway » de la VM) soit a posteriori avec une capture de trafic réseau prise séparément (Figure 4).

Après un peu de remaniement du graphe, les communications apparaissent clairement. Les cercles

Figure 4



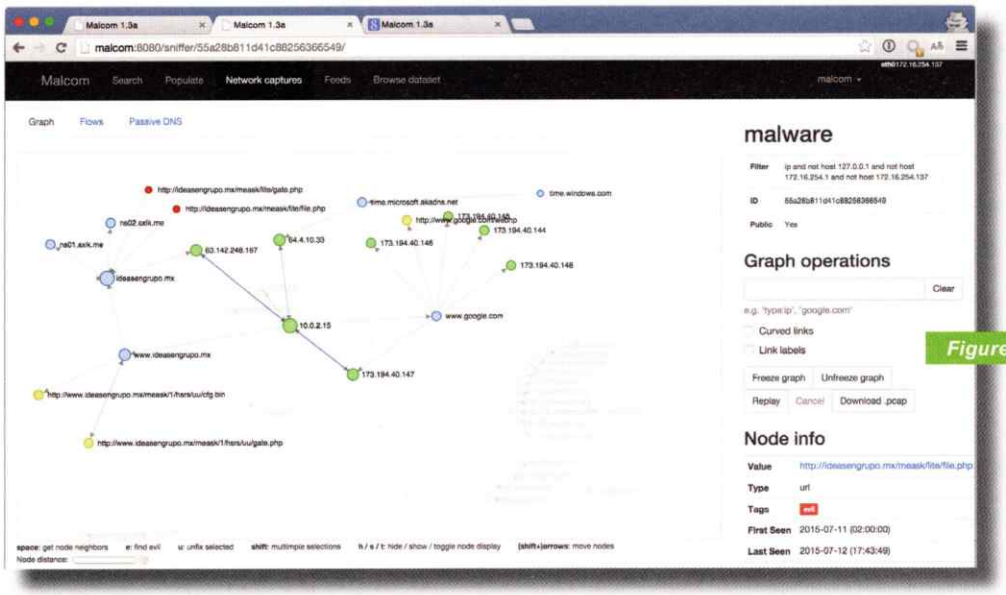


Figure 5

verts sont du trafic à priori non malveillant. Le cercle du bas représente des résolutions DNS qui ne nous intéressent pas forcément (les réponses au CDN d'Akamai). Le cercle supérieur est une requête chez Google, sur l'URL **http://w.google.com/webhp**. Ce trafic n'est pas malveillant en soi, même si beaucoup de malwares de la famille des Zbot l'utilisent pour tester leur connectivité à internet.

Les éléments dans le cercle rouge, liés au domaine `ideasengrupo.mx` sont, eux, beaucoup plus suspects. Si un feed a tagué un élément `evil`, il apparaîtra en rouge. Ici, aucun élément n'est rouge. Cela peut être dû au fait que le domaine en soi n'a pas été tagué, ou qu'il a été vu sans son sous-domaine « www » (Malcom considère les domaines et leurs sous-domaines comme des éléments différents).

Protip : On peut isoler les éléments d'intérêt à l'aide de la barre de recherche sur la droite. En tapant « ideasengrupo », seuls les éléments comprenant cette valeur seront affichés en pleine opacité, ainsi que les éléments directement liés à ces derniers.

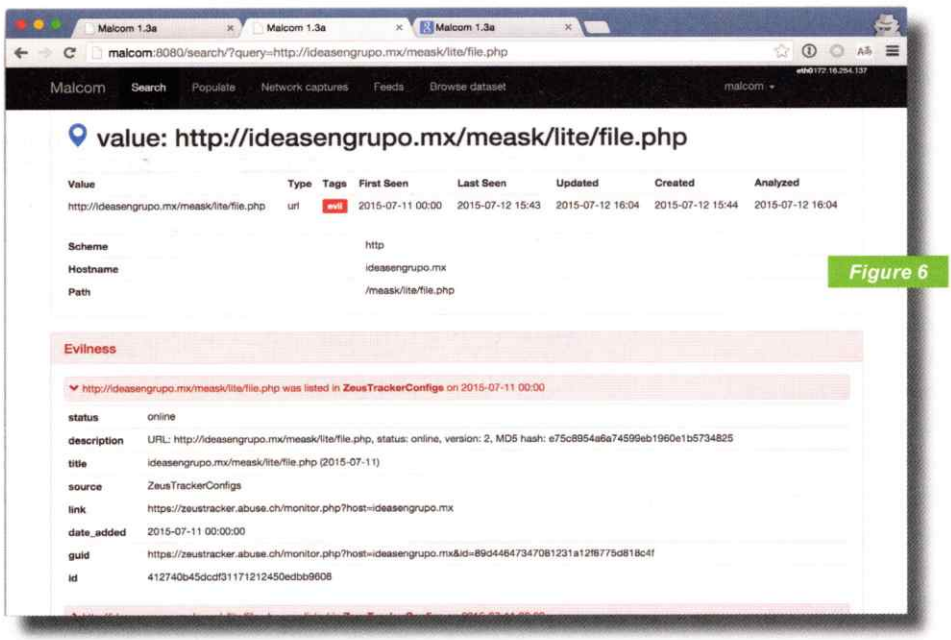


Figure 6

En sélectionnant les *nodes* « suspects » et à l'aide de la fonction **`find evil`**, on peut demander à Malcom de suivre le graphe d'éléments à la recherche d'autres éléments taggués **`evil`**. Ce que nous allons faire tout de suite (Figure 5).

On voit deux éléments **`evil`** apparaître en haut de l'écran. C'est plutôt bon signe ! Ces URL sont déjà connues de Malcom et sont rattachées directement au domaine **`ideasgroupo.mx`** (si notre feed avait tagué les domaines des URL malveillantes en **`evil`**, nous n'aurions même pas eu besoin de faire la recherche via **`find evil`**). L'avantage de pouvoir pivoter ainsi est de pouvoir découvrir des centres de C2 qui sont sur une même adresse IP, mais pas forcément sur un même nom de domaine (un peu comme un passive DNS de facto).

En passant la souris sur l'élément et en cliquant sur le lien de la *sidebar*, on arrive directement à la « fiche » de cette URL (Figure 6).

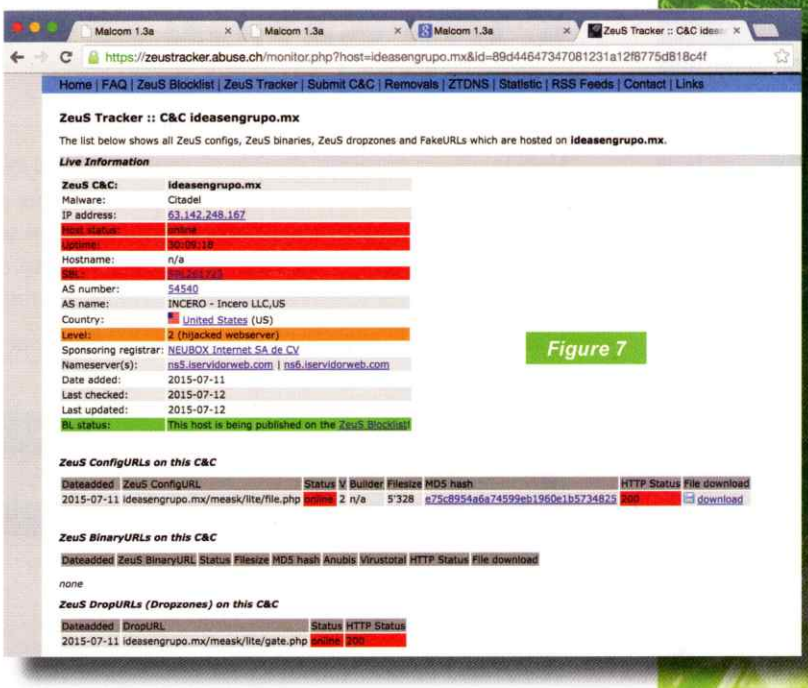


Figure 7

On y retrouve toutes les informations disponibles directement sur le *feed*, y compris un lien (champ GUID) où nous pourrons avoir plus d'informations directement sur le site **abuse.ch** (Figure 7).

Il s'agit donc d'un *sample* issu d'une famille dérivée de Zeus : Citadel.

On pourrait aussi se dire : « voyons tous les C2 Zeus actifs en ce moment ». Il est possible de faire ça grâce au champ « online » que notre feed a produit, et d'effectuer une recherche dans l'onglet « browse dataset » (Figure 8).

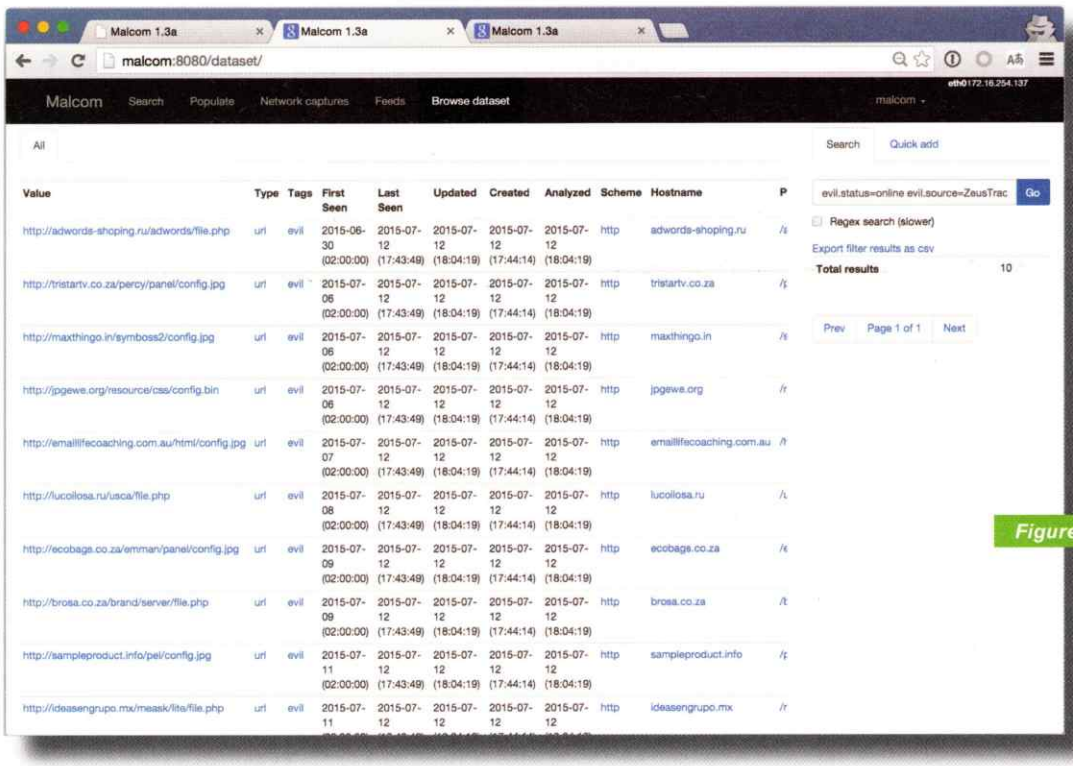


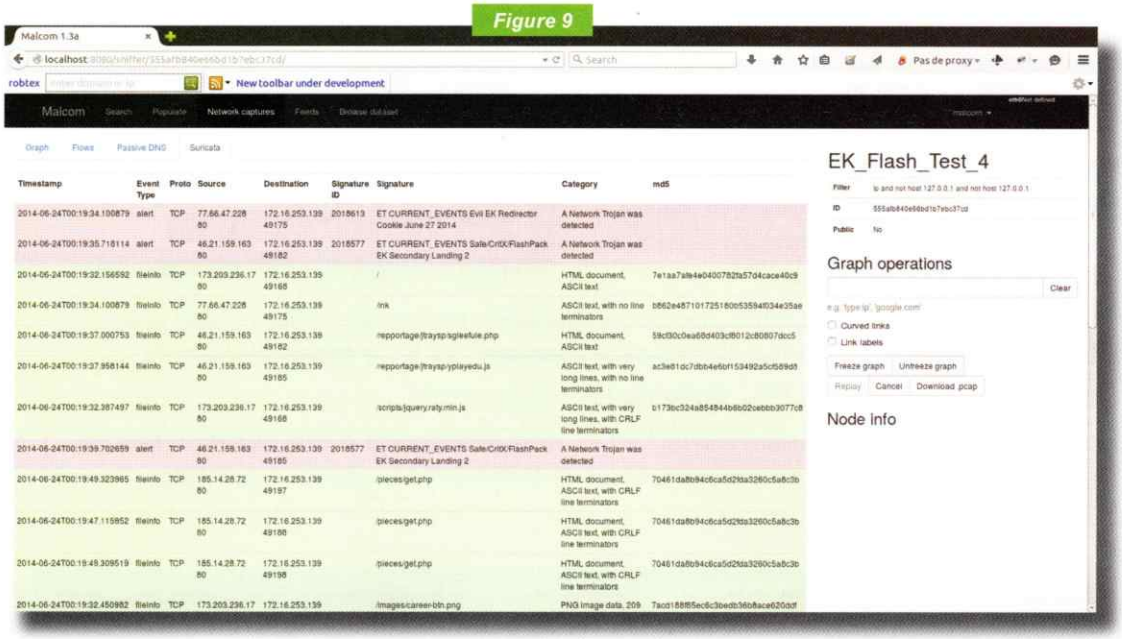
Figure 8

5. UTILISATION ET CRÉATION D'UN MODULE

Malcom offre la possibilité d'utiliser des modules d'analyse sur des captures réseau. Ces modules peuvent effectuer n'importe quelle opération sur les flux d'une capture ou sur chaque paquet individuel. Aujourd'hui, Malcom dispose de deux modules : un module de **passive DNS** et un module **Suricata**. Les modules sont *standalone*, c'est-à-dire qu'ils sont complètement indépendants du *core* et peuvent offrir des méthodes de visualisation aussi créatives que son développeur le souhaite. Voici un exemple de la sortie du module Suricata (Figure 9).

Nous allons voir comment créer un module *sniffer* pour Malcom qui permettra de passer **des règles Yara** sur chacun de nos flux.

Figure 9



5.1 Architecture basique

Les modules sont, vous l'aurez deviné, des classes Python héritant de la classe `Module`. Lorsqu'une session de *sniffing* est chargée, Malcom va charger les modules activés et leur donner accès au PCAP dans le système de fichiers ainsi qu'aux flux parsés. L'API web de Malcom fait en sorte qu'on puisse appeler les méthodes de ces modules, qui doivent donc retourner en général le code HTML que le module souhaite afficher. Comme chaque module est indépendant, il peut utiliser le moteur de *templating* de Flask (Jinja) pour générer des fichiers HTML, et il peut aussi fournir des fichiers « statiques » (CSS, JS) pour la mise en forme.

5.2 Création d'un module Yara

Pour notre module Yara, nous allons simplement renvoyer une table contenant les flux suspects avec les alertes qui auraient été levées. Là où ça devient assez puissant, c'est que tous les appels vers les fonctions des modules passent par l'API de Malcom. Il est donc tout à fait possible, en utilisant uniquement l'API, de lancer une session de *sniffing* et de récupérer les résultats d'un module donné sur cette session.

Chaque module a une fonction `bootstrap`, qui est appelée impérativement au chargement du module. Cette fonction doit être implémentée par tous les modules, et permet au module de générer le code HTML de base, qui pourra éventuellement appeler d'autres fonctions du module.

Nous créons donc un fichier `yara.py` dans un dossier `yara`, lui-même dans le dossier `sniffer/modules/`. Très basique, il ressemble à ça :

Fichier

```
rule http_requests
{
    strings:
        $get = "GET"
        $post = "POST"

    condition:
        $get or $post
}
```

La fonction `__init__` de notre module est comme suit :

Fichier

```
def __init__(self, session):
        self.session = session
        self.display_name = "Yara"
        self.name = "yarascan"

        self.rules = self.load_yara_rules(os.path.dirname(os.path.realpath(
__file__)))
        self.matches = self.load_entry() or {}

        super(self.__class__, self).__init__()
```

`load_yara_rules` est une fonction qui va parcourir les fichiers `.yar` dans le répertoire courant puis « compiler les règles ».

Le seul rôle de la fonction `bootstrap` sera d'appeler une fonction `content`, que nous aurions pu très bien appeler `toto`.

Fichier

```
    def bootstrap(self, args):
        content = self.content()

    def content(self):
        content = "<table class='table table-condensed'><tr><th>Flow</th>
<th>Rule</th><th>Param</th><th>Match</th><th>Offset</th></tr>"

        for flow in self.session.flows.values():
            matches = self.match_yara(flow.payload)
            if matches:
                for rule, match in matches.items():
                    m = match[0][0]
                    content +=  "<td>{}</td><td>{}</td><td>{}</td><td>{}
</td></tr>".format(rule, m[1], repr(m[2]), m[0])
        content += "</table>"
        return content
```

Là certains diraient que le code devient « un peu crade ». Ce n'est toujours pas du Perl, mais on mélange quand même du HTML et du code Python. Mais c'est à titre de démonstration. Il va de soi que le programmeur consciencieux utilisera des fichiers HTML et les fonctionnalités de *templating* de Jinja pour peupler cette table.

Ici, le code itère sur `session.flows`, mis à disposition par Malcom et contenant toutes les données sur les flux réseau reconstitués. Les données brutes des flux peuvent être utilisés avec l'attribut `.payload` de l'objet `flow`.

`match_yara` est la fonction qui va faire lancer les règles Yara sur notre *payload* binaire et qui va renvoyer un résultat (défini par la librairie Yara en l'occurrence). La chaîne de caractères représentant notre `<table>` est construite puis renvoyée à la vue (Figure 10, page suivante).

Figure 9

Voilà qu'en 5 minutes nous avons écrit un *plugin* capable de passer des règles Yara sur nos flux individuellement. Mais la colonne ***Flow ID*** est assez moche. Nous voudrions faire en sorte qu'elle affiche les adresses IP de notre flux et qu'en cliquant dessus, nous soyons automatiquement renvoyés vers le flux correspondant dans l'onglet ***Flow***.

5.2.1 Un peu de glaçage en JavaScript

Pour ça, nous allons faire un peu de JavaScript, ce qui nous permettra aussi de démontrer comment on ajoute des fichiers statiques (JS, CSS, images) à un module. On modifie légèrement notre fonction `bootstrap` :

Fichier

```
def bootstrap(self, args):
        content = self.add_static_tags(self.content())
        return content
```

On aurait peut-être pu utiliser des décorateurs Python pour accomplir la même fonction, mais cela sera laissé pour un *commit* ultérieur. La fonction `add_static_tags` parcourt le dossier `static` du module (il faut bien sûr l'avoir créé au préalable) et ajoute au code HTML renvoyé par `self.content()` tous les fichiers `.js` et `.css` qui s'y trouvent. En l'occurrence, nous n'avons qu'un fichier : `yara.js`, dont voici le contenu :

Fichier

```
$(  function() {

 $(".switcher").click(function(e){
  e.preventDefault();
  yara_switch($(this).data('flowid'));
 });

 console.log('yara js loaded');
});

function yara_switch(fid) {
 $("#flows-tab").tab("show");
 window.location.hash = "#"+fid;
}
```

Les lignes 3 à 8 seront exécutées une fois le fichier JavaScript chargé. Il va *binder* l'action `yara_switch` à tous les éléments de classe `switcher`. Évidemment, il faut répercuter ces quelques changements dans notre code HTML :

Fichier

```
def content(self):
        content = "<table class='table table-condensed'><tr><th>Flow</
th><th>Rule</th><th>Param</th><th>Match</th><th>Offset</th></tr>"

        for flow in self.session.flows.values():
            matches = self.match_yara(flow.payload)
            if matches:
                for rule, match in matches.items():
                    m = match[0][0]
```

```
                    content +=  "<tr><td><a class='switcher' data-
flowid='{}'".format(flow.fid) +\
                                " href='#'>{} &#8594; {}</a></td>".
format(flow.src_addr, flow.dst_addr)
                    content += "<td>{}</td><td>{}</td><td>{}</td><td>{}</td>
</tr>".format(rule,
m[1],
repr(m[2]),
m[0],
)
```

Et voilà ! Notre module est totalement fonctionnel, avec un minimum d'effort.

5.2.2 Sauvegarder les données d'un module

Les modules peuvent parfois être consommateurs en ressources et prendre un certain temps à s'exécuter. Il est donc possible de sauvegarder les résultats d'un module en base (tant que celui-ci est *BSON-serializable* : un dictionnaire, une liste, ce que vous voulez) à l'aide de la fonction `save_entry`. La fonction `load_entry` charge et renvoie le résultat du module en base. Pour tous les détails du code, le module Yara (codé en temps réel, pendant la rédaction de cet article !) est disponible sur le *repository* GitHub de Malcom [**https://github.com/tomchop/malcom**].

6. ROADMAP

Comme nous disions au début, Malcom est né d'un besoin opérationnel. Les besoins « de base » étaient assez simples, mais ils n'ont cessé d'évoluer avec le temps : « tiens, et si je mettais ça aussi ? », « la ligne de commandes c'est bien, mais une GUI web et des bubulles c'est quand même mieux »... Aussi, des demandes externes ont fait que l'outil a pris une tournure différente avec d'autres fonctionnalités. « Ça serait quand même pas mal si t'avais une API, non ? ». Heureusement, des contributions ici et là ont fait en sorte que le projet avance même plus vite que prévu, avec notamment la contribution de @Sebdraven sur le module Suricata et l'API web.

En bref, Malcom évolue constamment. Dans les améliorations à prévoir, il y a deux gros chantiers. L'un consiste à séparer le *core* du client. À terme, il y aura un serveur web qui tournera à distance et qui s'occupera exclusivement du *feeding* et des *analytics*. Sur son poste (« en local »), on aura un client qui se connectera sur l'API du serveur distant. Cela permettra de partager l'information accumulée sur un serveur Malcom entre plusieurs clients, plus légers. À ce stade, « Malcom » restera Malcom en tant que client, mais il est bien possible que la partie serveur change de nom totalement... Le deuxième chantier consiste à apporter une dimension un peu plus haut niveau de l'information qui est présente dans la base. On aura ainsi la notion de « threat actors », « malware », « campagnes », etc. Les lecteurs avisés auront sûrement deviné qu'il en découlera certainement un export STIX, et pourquoi pas une interaction avec d'autres plates-formes du genre, comme MISP ou CRITS.

REMERCIEMENTS

Merci à tous ceux qui m'ont aidé, directement ou indirectement, dans le développement de cet outil. Merci aussi à ceux qui y ont contribué (que ce soit un module complet comme un simple `__init__.py`), merci à ceux qui s'en servent occasionnellement ou tous les jours (c'est de là qu'on puise la motivation pour continuer à travailler dessus !). Merci à l'équipe du CERT Société Générale qui a attribué de son temps à l'amélioration de l'outil ! Merci aussi à ceux qui le trollent^Wcritiquent, ce n'est qu'une motivation en plus pour le rendre un peu meilleur ! Et merci aussi à Guillaume (a.k.a. Gilmo !) et à Renaud Feil pour la relecture et l'opportunité de le présenter ici :) ■

4 À L'INTÉRIEUR DE LA FORTERESSE ASSIÉGÉE

RED TEAM : LE POINT DE VUE DE L'AUDITÉ ET DU COMMANDITAIRE

Nicolas Gaudin

Cet article présente mon retour d'expérience en tant que RSSI ayant commandité et suivi une large campagne de tests d'intrusion de type Red Team. Cette campagne s'est déroulée sur une durée de 3 mois et a mis notre organisation à l'épreuve face à plusieurs scénarios de menaces avec, par exemple le spear-phishing, des intrusions physiques, des intrusions sur les systèmes industriels embarqués et la simulation du vol du PDA d'un employé.

1. CHOISIR L'APPROCHE RED TEAM : DANS QUEL BUT ET POUR QUELS BÉNÉFICES

La nécessité pour une entité (entreprise, administration...) d'auditer régulièrement ses systèmes d'information (SI) n'est plus à démontrer aujourd'hui. Le périmètre de ces SI est en constante évolution : de nouveaux projets voient régulièrement le jour, des systèmes existants sont modifiés, enrichis, migrés, interfacés, de nouvelles technologies et de nouveaux usages font leur apparition. Dans le même temps, les vulnérabilités et menaces impactant ces SI se multiplient et se diversifient elles aussi. S'il arrive que des projets informatiques soient conçus et mis en œuvre de manière sécurisée dès leur lancement, ce n'est pas toujours le cas et rarement exhaustif. Il est également fréquent que des brèches apparaissent au cours du cycle de vie d'un projet, accompagnant l'obsolescence de certains pans des SI. Lorsqu'elles sont identifiées, elles sont le plus souvent corrigées (à plus ou moins long terme...) par le déploiement de mesures de sécurité préventives. Il demeure cependant périlleux pour un RSSI de se reposer sur la seule supposition que les mesures de sécurité ainsi déployées soient efficaces sans jamais s'en assurer en les évaluant.

Pour répondre à ce besoin, la pratique d'audits de sécurité a donc vu le jour et s'est aujourd'hui banalisée, le plus souvent réalisée par des spécialistes externes, indépendants de l'entité. Avec le temps, ces audits se sont professionnalisés, codifiés, diversifiés et spécialisés, voire même automatisés (en partie) et l'offre s'est bien étoffée : tests d'intrusion depuis l'externe, depuis l'interne, avec ou sans connaissance préalable, audit de code applicatif, revue de configuration, système SCADA, Wi-Fi, audit de conformité, de maturité organisationnelle...

La taille et l'hétérogénéité des SI à sécuriser allant croissant, s'accompagnant de la disparition d'un périmètre bien défini des SI (émergence du Cloud et de la mobilité, objets connectés, Shadow IT, porosité avec la sphère privée des utilisateurs...) et considérant (objectivement) que l'aspect sécurité ne revêt bien souvent qu'une faible partie des budgets alloués aux SI, il devient de plus en plus difficile d'auditer la sécurité dans une approche globale et rationalisée. Il faut donc toujours faire des choix dans ce que l'on veut auditer. Or, le choix du périmètre à cibler (une application ou un projet sensible, un réseau local, filaire ou Wi-Fi) et de l'approche à retenir (audit de configuration en « boîte blanche », revue de code, test d'intrusion) demeure cornélien pour un RSSI (même avec une approche par le risque), car il induit nécessairement l'exclusion d'un pan très large du SI. Longtemps, les tests se sont focalisés sur le seul aspect réseau et système, avant que l'évolution des tendances en ce qui concerne les menaces ne les fasse infléchir vers les applications (principalement web) et les développements, puis des technologies encore plus spécifiques à un métier (comme le SCADA). Aujourd'hui, toutes les études récentes s'accordent pour dire que l'angle d'approche le plus couramment ciblé par les attaquants (de plus en plus méthodiques et organisés) est l'utilisateur (interne), contournant ainsi les barrières périmétriques qui sont aujourd'hui très largement déployées. En effet, il est clairement admis que beaucoup d'attaques commencent désormais par l'envoi d'un e-mail comportant une pièce jointe infectée ou invitant l'utilisateur à se connecter à un site malveillant.

Tous ces éléments concourent à ce que les tests Red Team gagnent en popularité auprès des décideurs informatiques et des RSSI. En effet, par leur caractère « réaliste » (la simulation de menaces concrètes et actuelles, pas nécessairement sophistiquées, mais déterminées), l'aspect multicanal des attaques réalisées (phishing, ingénierie sociale, intrusion physique

avec dépôt d'implants puis attaques depuis les réseaux internes, attaques depuis un terminal mobile dérobé, attaques des services exposés à l'Internet...) et le ciblage des vulnérabilités les plus aisément exploitables et ayant le plus fort impact, les audits Red Team constituent un compromis particulièrement intéressant pour permettre au RSSI de dégager les axes d'amélioration les plus significatifs et prioritaires pour la sécurité de son SI. Le fait que les attaques réalisées en Red Team ciblent l'entité et ses collaborateurs correspond bien aux scénarios de risques les plus redoutés par les décideurs (fraudes internes ou externes, espionnage industriel, sabotage), par opposition à des attaques plus opportunistes. En outre, ces tests contribuent également à la démarche de sensibilisation générale du personnel, toujours nécessaire, mais jamais suffisante, et notamment des dirigeants (les décideurs) et des équipes opérationnelles (remise en cause des pratiques et des modèles de référence utilisés).

2. À QUEL MOMENT CONVIENT-IL DE RÉALISER UN AUDIT RED TEAM ?

Il est très pertinent pour un RSSI de déclencher un audit dès sa prise de fonction dans une nouvelle entité (qu'il s'agisse d'une création de poste ou de la poursuite de l'action d'un prédécesseur). Cela lui permet, d'une part, de cartographier son SI (et ainsi d'évaluer la surface d'attaque et son degré d'exposition), de se faire sa propre idée du niveau de sécurité en place (présence et efficacité des dispositifs de sécurité, maintien à jour, maturité des équipes et des processus de sécurité, qualité des prestataires en jeu, localisation des faiblesses principales : développements, infrastructure, bureautique, personnel...) et donc des défis qu'il aura à relever en priorité. D'autre part, cela lui fournit (selon le résultat des tests) d'excellents arguments pour ensuite justifier le déblocage de budgets et leur affectation aux sujets les plus critiques à traiter, en définissant par la même occasion sa feuille de route.

Les caractéristiques du test Red Team (réalisme, couverture large, mais ciblage des failles les plus béantes) répondent pleinement à ces objectifs pour peu que l'on dispose d'un premier budget suffisant, dont le montant dépend largement du spectre à couvrir, mais aussi de la nature des activités de l'entité ciblée. Un pur acteur SaaS n'aura pas les mêmes enjeux ni les mêmes risques majeurs qu'une entreprise du secteur industriel ou qu'une administration publique. Les résultats des tests Red Team peuvent, par exemple, mettre en évidence le besoin de se doter :

- d'un système de détection des intrusions ou de collecte et analyse des événements de sécurité (SIEM) si les assauts n'ont pas laissé de traces ou que celles-ci n'ont pas été suffisantes ou mises en évidence pour provoquer une réaction des équipes ;
- d'une structure opérationnelle de traitement des incidents de sécurité (SOC) et d'une chaîne d'escalade si les équipes ont montré une désorganisation ou un manque d'automatismes lorsqu'une attaque a été décelée.

Ce type de tests d'intrusion reste également complémentaire des autres formes d'audit déjà évoquées et s'insère donc très bien au cours d'un programme à long terme.

Il s'avère tout aussi pertinent de dérouler un test Red Team après un ou plusieurs cycles d'un programme de sécurité, afin de mesurer les progrès effectués par rapport à un état initial. Il peut en effet provoquer un sentiment d'abattement au sein des équipes s'il est initié trop tôt, c'est-à-dire à un moment où il est évident et admis par tous que le niveau de maturité en matière de sécurité est insuffisant. À l'inverse, lorsque les briques de sécurité essentielles sont en place (notamment la structure de gestion des incidents), les tests Red Team constituent un excellent exercice d'entraînement dans des conditions réelles permettant d'aiguiser les réflexes ou de peaufiner le dispositif. Le tout pouvant s'effectuer dans une atmosphère relativement ludique.

3. LA DÉMARCHE ET L'ORGANISATION PRÉALABLE

Une analyse des risques, même minimale, apparaît requise avant le lancement d'une campagne Red Team, ce afin de déterminer une liste finie d'objectifs précis à atteindre correspondant à l'exploitation des scénarios de risques les plus redoutés. Quelques exemples de « trophées » à décrocher : prise de contrôle d'un serveur censé être inaccessible, usurpation d'identité, accès à la boîte aux lettres d'un VIP, accès non autorisé à une application sensible, extraction de données à caractère personnel, réalisation d'une opération non autorisée, fraude...

La définition de ces objectifs peut bien sûr légèrement fausser l'approche « boîte noire » propre aux tests Red Team par la révélation d'informations contextuelles à l'entité ciblée, mais elle permet d'exploiter au mieux le temps imparti, nécessairement limité, pour aller le plus loin possible dans l'intrusion et aide à clarifier la restitution des résultats qui sera faite aux dirigeants par la présentation d'impacts « parlants ».

Il est également préférable de limiter le niveau d'information fourni aux équipes au sujet des tests pour rester le plus possible dans des conditions proches d'un cas réel d'attaque et évaluer au mieux leurs capacités de réaction, sans fausser le résultat par une vigilance anormalement élevée. Le maintien d'une certaine confidentialité reste toutefois difficile : les dirigeants de l'entité auditée doivent au minimum en être informés et donner leur accord, la direction financière signe le devis, la direction juridique valide les conditions contractuelles et les engagements de confidentialité et le recours à certains prestataires pour la délivrance de services d'hébergement ou de sécurité (un SOC externalisé par exemple) peut imposer une notification préalable. De même, il peut être nécessaire d'impliquer les directions opérationnelles pour éviter que le test soit mal vécu par les équipes ou qu'une réaction excessive soit déclenchée dans le cas, par exemple, où l'intrusion physique est interceptée par un salarié ou par le personnel de sécurité physique, ou si l'une des cibles visées se situe dans un lieu public. Pour éviter tout risque d'incident, il est donc nécessaire, outre les engagements classiques de confidentialité entre le prestataire d'audit et l'organisme audité, de munir les auditeurs d'une « carte de sortie de prison » sous la forme d'une lettre expliquant la raison de leur présence et la personne à contacter. Il faut donc trouver un bon dosage entre une certaine transparence pour ôter le caractère hostile ou inquisiteur de la démarche tout en restant suffisamment flou, notamment sur les dates exactes, pour préserver son aspect réaliste.

Par ailleurs, il peut s'avérer extrêmement précieux de configurer en amont les équipements qui seront visés afin de disposer a posteriori de traces des assauts logiques (journaux d'événements) afin de faire l'exercice d'une analyse post-mortem et ainsi enrichir la détection d'intrusion ou d'autres dispositifs pour le futur. Cette tâche, qui peut par ailleurs s'avérer fastidieuse d'autant que les angles d'attaque ne sont pas définis à l'avance, ne favorise évidemment pas la conservation du caractère confidentiel de l'exercice si elle est effectuée peu de temps avant l'audit Red Team.

4. LES LIMITES DE L'EXERCICE

Bien entendu, les tests Red Team ont aussi leurs limites. La largeur du spectre des assauts et la focalisation sur les failles les plus rapidement exploitables ne conviennent pas si votre objectif est d'être exhaustif sur un angle d'attaque précis ou d'auditer en profondeur une cible particulière. Si votre but est de vérifier la sécurité d'une application Internet, le test d'intrusion web classique voire l'approche « boîte blanche » sont sans aucun doute plus adaptés. Cela confirme la complémentarité des différentes approches d'audit.

En outre, la durée de chaque attaque est forcément limitée. Il se peut donc qu'un test n'aboutisse pas à une compromission à l'issue du temps imparti. Cela peut être vu comme positif, mais il subsistera une

incertitude pour le RSSI quant à la sécurité de son SI. Il faut donc faire une estimation de l'effort à fournir et accepter qu'au-delà de celui-ci la cible visée sera jugée suffisamment protégée. C'est la même approche que pour l'acceptation d'un risque à l'issue d'une analyse de risques.

Par ailleurs, le déroulé des opérations d'une campagne Red Team est rarement linéaire et plusieurs itérations d'un même type d'attaque sont souvent réalisées, les différentes attaques étant bien souvent entrelacées, voire simultanées (car bien souvent réalisées à plusieurs). Par exemple, une campagne de phishing peut se dérouler en plusieurs phases, afin de ne pas éveiller les soupçons et ainsi d'augmenter son efficacité, ce qui peut perturber le suivi des opérations par le commanditaire, mais permettre d'élargir la fenêtre de temps allouée pour une attaque. Cette particularité permettant aussi la réutilisation avec succès dans certaines phases d'attaques d'éléments récupérés à des étapes précédentes (tels que des identifiants de connexion, des mots de passe, de la documentation technique ou du code applicatif) fait partie des tests Red Team. Cela peut malheureusement provoquer une certaine déception chez le commanditaire :

- pour le RSSI, qui réalise que les auditeurs n'ont pas eu besoin d'auditer concrètement la cible et sont allés « au plus facile », laissant peut-être de côté d'autres failles importantes ;
- et au niveau de la Direction, qui peut avoir la fausse sensation d'une insuffisance de protection « technique » de sa plateforme.

Il s'agit pourtant bien d'un procédé que pratiquent concrètement les véritables attaquants hostiles et la réutilisation de mots de passe dans différents environnements constitue une faille très sérieuse, contraire aux bonnes pratiques. C'est donc aussi l'intérêt du Red Team : simuler de façon réaliste une véritable attaque.

Il est important de noter que l'intervalle de temps entre le début d'une campagne Red Team et la restitution des résultats peut également être conséquent, pouvant provoquer une certaine impatience au niveau de la Direction ou une tension légèrement paranoïaque chez les équipes opérationnelles si la campagne a été détectée. Ce délai peut aussi nuire à la préservation des traces de l'audit pour une analyse post-mortem, si celles-ci n'ont pas été convenablement paramétrées ou conservées.

L'aspect financier est également à considérer : le coût d'une campagne Red Team peut s'avérer important comparé à un audit plus classique et ciblé, surtout si le périmètre à couvrir est large ou avec une surface d'attaque multiple. Là encore, la définition de scénarios de compromission précis et correspondant à de vrais risques redoutés permet de rationaliser. Le budget consacré peut néanmoins donner une idée du montant qui serait requis par un organisme malveillant pour commanditer une attaque ciblée (sur un concurrent par exemple) et l'on peut confronter ce montant à la valeur des actifs informationnels à protéger ou de l'impact sur la réputation de l'entité.

En interne, l'impact en termes de charge sur les équipes peut également être non négligeable, mais uniquement si l'organisation sécurité en place est déjà mature (risque de « branle-bas de combat » au sein du SOC en cas d'attaque détectée). Enfin, l'étendue de la compromission peut être de nature à effrayer une Direction, qui peut (légitimement) se dire qu'elle a payé pour « donner les clés de sa boutique à des inconnus ». Il est donc crucial de bien choisir un prestataire de confiance, reconnu dans le milieu de la sécurité pour son sérieux et de porter une attention particulière aux clauses du contrat. Mais c'est aussi le cas pour les autres types d'audit, même si les « dégâts » sont souvent plus limités.

5. LE DÉROULÉ DE LA CAMPAGNE

La campagne Red Team doit logiquement commencer par une phase de collecte d'informations et de découverte du périmètre à cibler : c'est la phase dite « de reconnaissance ». Cette phase permet notamment aux auditeurs de découvrir, en recherche par sources ouvertes, les sites de l'entité qui sont exposés publiquement sur Internet, les noms de domaines, les plages d'adresses IP, une liste

de collaborateurs avec leurs coordonnées e-mail ou téléphoniques, voire d'autres informations techniques ou contextuelles pouvant s'avérer utiles par la suite (dépôts GitHub, documentation technique...). Il peut être pertinent de fournir aux auditeurs quelques éléments de ciblage (nom de la société et des filiales, projets, noms de domaines) afin de limiter le temps qu'ils consommeront sur cette phase et le réserver pour les phases suivantes. Mais il reste préférable de leur laisser le soin de faire cette recherche au maximum par eux-mêmes et de confronter ensuite les résultats de celle-ci à la vision interne de la cartographie du SI de l'entité visée. Cela peut donner lieu à de réelles surprises, bonnes si peu d'informations s'avèrent publiquement disponibles, ou mauvaises lorsque des serveurs accessibles depuis l'Internet et non comptabilisés dans l'inventaire sont découverts ou qu'une fuite d'informations est constatée. Dans tous les cas, il est toujours utile pour un RSSI de savoir ce que pourrait trouver un véritable assaillant.

Une fois la phase de reconnaissance terminée, le périmètre découvert doit alors être validé par le commanditaire. Le RSSI peut alors choisir de restreindre les attaques qui seront portées à un sous-ensemble du périmètre, exclure des éléments qui auraient été à tort inclus dans celui-ci (par exemple, un site quasi-homonyme sans rapport avec l'entité) et même retirer parmi les adresses e-mail qui auraient été collectées celles des personnes ayant quitté la société ou celles de VIP sensibles.

Une fois le périmètre validé et les différents mandats d'audit éventuellement nécessaires dûment signés, les attaques proprement dites peuvent commencer, d'abord à distance. Il n'est pas rare que différentes phases d'attaque démarrent simultanément, celles-ci pouvant prendre un certain temps avant de donner des résultats (c'est notamment vrai pour le phishing), et soient même réitérées après l'issue de phases intermédiaires. Par exemple, il est pertinent de démarrer la première vague de phishing en même temps que les scans de ports. Dans mon propre cas, la présence de dispositifs filtrants de type « fail2ban » a considérablement ralenti l'étape des scans de ports. Les scans de ports et de vulnérabilités vont ensuite permettre aux auditeurs d'identifier des vulnérabilités exploitables depuis l'Internet (applications Web, API mobiles, passerelles VPN). Ils feront ensuite leur choix parmi celles-ci en privilégiant soit les sites d'importance ou à forte visibilité, soit les plus vulnérables. Les phases de phishing permettront, quant à elles, de glaner des identifiants de connexion, des mots de passe (qui pourront ensuite être réutilisés dans d'autres phases d'attaque, internes ou externes) ou de prendre le contrôle du poste de l'utilisateur visé situé sur le réseau interne et ensuite rebondir sur les infrastructures informatiques centrales.

Viennent ensuite les phases « sur site », avec l'attaque des réseaux Wi-Fi de l'entité (depuis l'extérieur ou dans l'enceinte du site) et surtout l'intrusion physique dans les locaux. Un repérage géographique préalable via Google Maps/StreetView/Earth permet facilement de préparer le terrain et repérer les lieux avant l'opération (présence d'entrées secondaires pour l'intrusion physique ou de terrasses de cafés pour mener l'attaque Wi-Fi). L'attaque Wi-Fi permet, si elle aboutit, de fournir aux auditeurs un accès au réseau interne en vue de rebondir ensuite sur l'infrastructure informatique centrale. L'intrusion physique permet, quant à elle, de démontrer la possibilité de dérober des documents sensibles ou un terminal mobile (ordinateur portable ou ordiphone), même si cette possibilité sera rarement mise en œuvre en pratique. Elle permet surtout d'obtenir un accès sur le réseau local filaire interne par le branchement d'un ordinateur portable ou d'un implant miniature disposant à la fois d'une connexion Ethernet et d'une antenne 3G, demeurant ainsi accessible à distance par les auditeurs. Le but pour l'auditeur est alors d'obtenir cet accès le plus discrètement possible afin de le conserver suffisamment longtemps sans être détecté pour pouvoir mener à bien l'ensemble des attaques suivantes et rebondir le plus loin possible dans le SI « visité ». L'implant miniature répond parfaitement à cet objectif, car il peut être beaucoup plus aisément camouflé.

Viennent enfin les phases de rebond depuis l'interne, incluant la prise de contrôle de postes de travail et de serveurs, de domaines Microsoft, l'exploration des réseaux internes, de la ToIP,

des partages de fichiers, des Intranets, des applications métiers, des infrastructures réseau, des hyperviseurs... Ces étapes sont les mêmes que celles des audits plus classiques, dont l'approche est aujourd'hui bien connue.

Il est important de mentionner qu'en abordant cette étape, les auditeurs ont encore une vision incomplète de leur cible. Il est donc quasiment certain que durant ces attaques, des composants nouveaux viendront s'ajouter au périmètre cartographié en phase de reconnaissance, certains pouvant même s'avérer totalement hors du périmètre autorisable. Le RSSI pourra alors préférer échanger avec les auditeurs sur la suite à donner sur les composants ainsi découverts ou bien les laisser poursuivre selon leur gré. À ces étapes d'attaques de SI classiques peuvent venir se greffer des modules supplémentaires plus spécifiques aux métiers de l'entité (comme par exemple l'audit de systèmes industriels ou de terminaux de paiement). Les nouveaux usages de l'Internet avec l'émergence des services Cloud, la forte tendance à la mobilité et la prolifération des systèmes embarqués et des objets connectés ont également généré de nouvelles activités et de nouveaux angles d'attaques, qui peuvent également faire l'objet d'un volet d'une campagne Red Team. Enfin, comme il l'a déjà été évoqué dans cet article, je recommande d'ajouter en option une phase d'analyse post-mortem des journaux, particulièrement utile pour l'amélioration de la réaction des équipes si certaines attaques n'ont pas été détectées par celles-ci.

6. LES RÉSULTATS ET LES LIVRABLES

Comme évoqué, le fait qu'une campagne Red Team puisse s'étaler sur une durée assez longue nécessite de rester en contact avec les auditeurs à intervalles réguliers pour ne pas perdre le fil. Il faut donc convenir avec eux de comptes rendus téléphoniques réguliers (un mini compte-rendu écrit, par mail ou dans un autre format, peut aussi convenir). Pour ces points intermédiaires, une fréquence hebdomadaire est parfaitement suffisante, mais il faut exiger des auditeurs des appels ponctuels lorsqu'une faille sérieuse nécessitant une correction rapide est découverte ou lorsque des progrès multiples sont réalisés dans un laps de temps très court. Les auditeurs sauront toutefois garder une certaine latence sur l'information fournie au sujet de leur progression afin de ne pas fausser les résultats par une divulgation trop anticipée qui provoquerait une réaction inopportune chez l'audité ou manquerait de recul pour l'interprétation.

Comme dans tous les audits « classiques », les tests Red Team donnent lieu à un rapport détaillé, une synthèse managériale et un tableau de suivi des vulnérabilités et des corrections. Libre au commanditaire de convenir avec l'auditeur du niveau de détail désiré, du format ou des points à accentuer dans le rapport. Une rédaction thématique du rapport aura le mérite de grouper les failles par type ou par cible et simplifiera la mise en place du plan de correction et la répartition des tâches entre les différents acteurs internes de l'entité auditée. Une organisation chronologique du rapport sera en revanche préférable pour une meilleure compréhension du chemin critique emprunté par les auditeurs. Une telle présentation permettra en effet de supprimer en premier lieu les vulnérabilités sans lesquelles certains objectifs d'attaque n'auraient pu être atteints et de réaliser ce qui a été réellement audité et ce qui a été laissé de côté (utilisation des chemins les plus courts et non exhaustivité des tests). En outre, elle facilitera l'analyse post-mortem des journaux d'événements des serveurs compromis.

À ces livrables classiques, il peut être tout à fait pertinent de faire ajouter une frise chronologique des étapes de l'audit, qui constitue une très bonne alternative ou un bon complément à la rédaction chronologique du rapport détaillé. À cette fin, il faut donc veiller à s'accorder avec les auditeurs dès le départ pour constituer au mieux cette chronologie au fur et à mesure, notamment par la collecte régulière d'éléments de preuve: e-mails envoyés, photos des lieux visités et des implants branchés lors des intrusions physiques, copies d'écran, vidéos.

Il est préférable de limiter les enregistrements vidéo aux intrusions logiques. L'enregistrement vidéo des intrusions physiques ou l'enregistrement audio des conversations avec des personnes rencontrées ou « manipulées » par téléphone restent toutefois délicats à produire tels quels en tant que livrables, principalement pour des questions éthiques et juridiques vis-à-vis des salariés. L'objectif d'un audit n'est pas de pointer du doigt un salarié en particulier, mais bel et bien d'améliorer des pratiques et une sensibilisation globales à l'organisme. Il est donc préférable de retranscrire les phrases marquantes par écrit en prenant soin d'anonymiser leurs auteurs. Dans ce même objectif, l'anonymisation des autres résultats de l'audit, comme les noms des utilisateurs dont les comptes ou les postes de travail ont été compromis semble aller de soi. Cela pose cependant quelques difficultés. Il me paraît essentiel en tant que commanditaire d'obtenir des auditeurs la liste exhaustive des comptes utilisateurs ayant été compromis. En effet, selon le parc ciblé, il peut être irréaliste ou très coûteux de renouveler l'ensemble des mots de passe à l'issue de l'audit. Il me paraît également essentiel d'exiger des auditeurs de fournir la liste des serveurs compromis ou sur lesquels ils ont réalisé des actions.

Enfin, la liste des documents collectés constitue également un élément important, permettant d'une part de comprendre ce qui a été ciblé, ce qui a pu fuiter et aider par la suite à la mise en place (ou à la révision) d'une classification, de mécanismes de DLP et d'une sensibilisation plus ciblée. Ces éléments permettront éventuellement de rejouer en interne (pour peu que la compétence existe) certaines étapes de l'audit afin de vérifier leur remédiation ou, si besoin, de compléter l'audit.

Pour ce dernier besoin, le détail des commandes et des outils utilisés par les auditeurs a également une utilité a posteriori. On comprendra aisément que certains outils fabriqués par les auditeurs eux-mêmes ne puissent être communiqués au mandant, mais nombre d'entre eux sont publics et le savoir-faire (méthode, rapidité, expérience, connaissance technique) reste l'intérêt premier des cabinets, pas les outils.

7. LA RESTITUTION ET LE PLAN D'ACTION

La restitution des résultats, que ce soit au travers d'une synthèse managériale ou d'une soutenance a également son importance. Encore une fois, il est important que l'audit reste bien vécu par le personnel de l'organisme audité. Selon le contexte, les résultats et l'organisme audité, une approche ludique dans le déroulé comme dans la restitution contribue à l'implication des équipes pour la suite du travail, à savoir l'application du plan de correction qui suivra l'audit. Il ne s'agit pas là de minimiser l'impact, mais plutôt de susciter une saine émulation et une envie de « faire mieux la prochaine fois ». Le résultat peut aussi être présenté en tant que séance de sensibilisation. Il reste préférable de commencer par la direction et de convenir avec elle de la forme et de l'opportunité d'une restitution au reste du personnel (par petits groupes en focalisant sur leurs périmètres respectifs ou en assemblées générales, lorsque c'est réalisable).

À cette occasion, il est important de souligner aussi bien les points faibles que les points positifs. De bonnes réactions peuvent (doivent ?) en effet se produire durant l'audit et tout encouragement est bon à donner : détection du phishing par une majorité des salariés visés, escalade au service de gestion des incidents de sécurité, découverte visuelle de l'implant dans les locaux ou détection par le réseau, interception d'un intrus dans les locaux, confinement d'une attaque par la désactivation d'un compte dont un administrateur a constaté la compromission, révocation d'un certificat SSL VPN ou d'une clé SSH compromise, réaction de l'antivirus ou du firewall personnel d'un collaborateur. À l'inverse, un orgueil mal placé doit également être dégonflé, mais de façon subtile pour pousser à une réaction constructive.

Il faut noter aussi que la perception de l'auditeur est différente de celle de l'audité. Les objectifs sont en effet « opposés ». De ce fait, la présentation d'objectifs atteints pour les « auditeurs » (trophées

capturés) est souvent perçue par une direction comme un échec de la sécurité de son SI. Il faut donc prendre soin de présenter les résultats de manière non ambiguë (par exemple en évitant d'utiliser un code couleur vert pour lister les objectifs atteints par les auditeurs).

Enfin, chaque audit apporte son lot de recommandations. Il arrive souvent que celles-ci donnent lieu à débat, quant à leur applicabilité dans le contexte de l'organisme (plan de charge, lourdeur de la solution). Il est donc de bon ton de discuter de ces recommandations avec les équipes opérationnelles pour qu'elles se les approprient et adoptent les solutions qui conviendront le mieux à leurs usages et aux besoins de sécurité.

Le cas du phishing est toujours complexe. Le RSSI ne peut que miser sur un taux d'échec de la campagne de phishing. Lorsque les messages sont bien faits (c'est-à-dire qu'ils traversent les anti-spam et autres anti-malwares sans encombre), il y a toujours une chance qu'ils soient ouverts par quelqu'un. Ainsi, la seule recommandation permettant de s'en prémunir est de compter sur la vigilance de la victime et sa sensibilisation à ce type de menace. Son impact peut toutefois être limité par le durcissement des machines utilisateurs, la restriction des droits administrateurs, la surveillance des accès sortants, et de bonnes campagnes de sensibilisation.

De même, la mise en place d'une authentification pour l'accès au réseau local (type 802.1X ou NAC) est un bon moyen de détecter sinon de stopper la phase d'intrusion physique, mais son usage au quotidien peut s'avérer difficile à fluidifier.

Le rebond est quant à lui fortement compliqué par un bon cloisonnement réseau, mais il peut s'avérer difficile à mettre en œuvre à court terme.

Enfin, la fuite d'informations se trouve complexifiée par un bon filtrage sortant, c'est en tout cas un excellent point de détection, tout comme le DNS.

8. EN SYNTHÈSE

La nécessité pour un organisme d'auditer ses systèmes d'information du point de vue de la sécurité est aujourd'hui indéniable. Par leur approche globale (canaux d'attaque multiples), l'évaluation des dispositifs à la fois techniques et humains (bonnes pratiques, processus de réaction, sécurité physique, réflexes du personnel) et leur proximité du cas d'attaque réel, pouvant aussi cibler des risques spécifiques à une activité, les campagnes de tests d'intrusion Red Team sont un excellent moyen d'évaluer le niveau de sécurité d'un SI.

La conduite d'une campagne Red Team peut s'avérer pertinente à tout moment dans un programme de sécurité pour un RSSI : pour évaluer un niveau de départ ou vérifier l'efficacité de dispositifs récemment déployés dans le cadre d'un programme de sécurité.

Toutefois, la portée de ces campagnes comporte certaines limites, notamment la non-exhaustivité des vulnérabilités découvertes, même si celles-ci sont généralement les plus critiques. Une certaine préparation est donc requise pour éviter les écueils et en tirer le meilleur bénéfice : analyse des risques pour définir des objectifs concrets à cibler, information totale des décisionnaires, mais discrétion vis-à-vis du personnel, sélection d'un prestataire de confiance, collecte des journaux d'événements. La démarche, déjà bien codifiée, requiert l'implication du commanditaire tout au long de la campagne, pour la validation des cibles qui seront découvertes (utilisateurs, applications, adresses IP) et éviter des dérives hors périmètre. Les résultats délivrés doivent figurer la chronologie de l'audit, l'obtention datée de trophées permettant de sensibiliser la direction, mais aussi de retracer la démarche et ainsi déterminer les actions à mener en priorité pour corriger les failles ayant permis les plus grandes progressions dans les attaques. La restitution de ces résultats doit également être bien préparée et impliquer le personnel, entrant dans le cadre de la sensibilisation et permettant de débattre des recommandations. Comme pour tout audit, le Red Team restitue une image à un instant donné, il faut donc réitérer après l'application, même partielle, du plan de correction. ■