BEEZH: une plateforme de détonation réaliste pour l'analyse des modes opératoires d'attaquants

Frédéric Guihéry, Alban Siffer, and Joseph Paillard

AMOSSYS

1 Les besoins liés à la détonation

Dans le cadre de la lutte informatique défensive, la connaissance du mode opératoire des groupes d'attaquants est essentielle pour pouvoir se défendre et adapter sa posture face aux nouvelles menaces cyber. Cette connaissance provient généralement de différentes capacités complémentaires, telles que la veille, l'analyse des vulnérabilités et codes d'exploitation, ou encore le suivi des groupes d'attaquants et leurs cibles d'intérêts. L'une de ces capacités, la détonation, consiste à attirer un attaquant en activant des vecteurs d'infection (par exemple, une pièce jointe suspecte) dans un environnement simulé et maîtrisé, à des fins d'observation et d'apprentissage des techniques d'attaque employées. Cette détonation est typiquement réalisée dans un environnement de type *honeypot* faisant croire à l'attaquant qu'il est présent sur un système d'information opérationnel.

Les techniques d'attaques à observer proviennent ainsi soit directement du malware, soit de l'attaquant ayant le contrôle de ce dernier. Sur ces deux cas de figure, le besoin de réalisme de l'environnement honeypot généré est crucial puisque le malware (ou l'attaquant ayant pris le contrôle de la cible) peut exécuter des routines vérifiant la crédibilité de l'environnement attaqué.

D'autre part, une plateforme de détonation doit permettre de produire des renseignements précis et complets sur la menace, en couvrant à la fois les techniques d'attaques connues et, idéalement, celles encore inconnues ou non publiques. De tels renseignements peuvent prendre la forme d'indicateurs de compromission (hashs d'outils d'attaques, IP et DNS de l'infrastructure d'attaque, etc.), ou peuvent caractériser le mode opératoire des groupes d'attaquants (tactiques, techniques et procédures d'attaques usuellement employées, également appelées TTP). Les renseignements produits sont alors généralement partagés à d'autres entités menant des opérations de détection (équipes SOC) ou d'analyse de la menace (équipes CERT / CTI).

Les innovations présentées dans ce papier visent à répondre aux deux besoins exprimés, à savoir :

- **le besoin de réalisme du honeypot** ;
- **la capacité à extraire les traces d'intérêts pour la production de renseignements sur la menace**.

Concernant le besoin de réalisme du honeypot, plusieurs aspects doivent généralement être traités : l'environnement système et applicatif, l'environnement réseau, et également la pertinence de la vie qu'il peut y avoir sur le système d'information simulé (les ressources et les actions de vie manipulant ces ressources). Dans ce papier, le réalisme est ici principalement abordé sous l'angle de la simulation de comportements utilisateur permettant de produire de la vie sur le honeypot.

2 La plateforme de détonation BEEZH

A des fins d'expérimentation et de validation des travaux, la plateforme de honeypot **BEEZH**, développée par AMOSSYS, est ici exploitée. Cette plateforme a reçu en 2020 le premier prix du

défi Deceptive Security organisé par la Direction Générale de l'Armement, le Commandement de la Cyberdéfense et l'Innovation Défense Lab. Cette plateforme apporte notamment les capacités de base suivantes :

- une capacité de construction de systèmes d'information (inventaire SI et topologie) ;
- une capacité de personnalisation avancée et de vieillissement de l'environnement simulé (paramétrage des OS, de l'Active Directory, des arborescences de fichiers, des comptes et groupes utilisateurs, etc.).

Dans l'objectif de simplifier le travail d'un analyste, ces capacités sont construites de manière à être utilisées de manière automatisable et intégrable avec des composants tiers.

2.1 Automatisation de la création d'environnements honeypot

La plateforme BEEZH expose une API REST permettant de contrôler, de manière scriptée, la construction et l'exécution de l'environnement SI du honeypot :

- modélisation dans un format pivot de l'environnement matériel, système et logiciel du honeypot ;
- modélisation dans un format pivot de la topologie réseau (switchs, routeurs, VLAN, adressage statique et dynamique, QOS pour forcer des bandes passantes, latences ou pertes de paquets, ...) ;
- capacité de lancement, arrêt, mise en pause, snapshot et rollback de l'environnement simulé.

Le format pivot de description de la topologie réseau est le format YAML. Cette description peut être produite automatiquement (génération aléatoire d'un environnement, en suivant, ou non, quelques directives comme le nombre et le type de machines) ou être directement écrite par un humain.

Une fois la simulation décrite dans le format pivot, celle-ci peut être instanciée puis exécutée par appel à des fonctions d'API REST. L'exécution de l'environnement se fait par virtualisation, avec l'utilisation des API libvirt, Open vSwitch et Open Virtual Network (OVN) afin de limiter l'adhérence à un hyperviseur spécifique.

Comme le montre la figure 1, l'architecture de la plateforme BEEZH supporte une capacité hybride, permettant le raccordement de l'environnement simulé à une machine physique ou un réseau physique. Il est également possible de rediriger l'ensemble des flux réseau du SI simulé vers une sonde externe, afin de détecter dynamiquement des comportements d'attaque visibles avec des signatures de NIDS.

La possibilité de mettre en pause le SI simulé peut être notamment utilisée pour réaliser de l'analyse forensique, notamment en cas de suspicion de la présence de l'attaquant. De telles opérations peuvent consister à extraire les mémoires RAM des VMs pour rechercher des éléments potentiellement suspects en mémoire (détection de techniques de *dll injection* par exemple). Il est ainsi possible de mieux caractériser le mode opératoire de l'attaquant.

Un catalogue de *basebox* (i.e. machines virtuelles) est mis à disposition pour fournir un ensemble de systèmes pré-déployés. Ces VMs sont proposées vierges ou avec des applications pré-installées (clients mail, navigateur, lecteurs PDF, lecteurs LibreOffice, ...). Le catalogue peut être complété avec de nouvelles basebox.

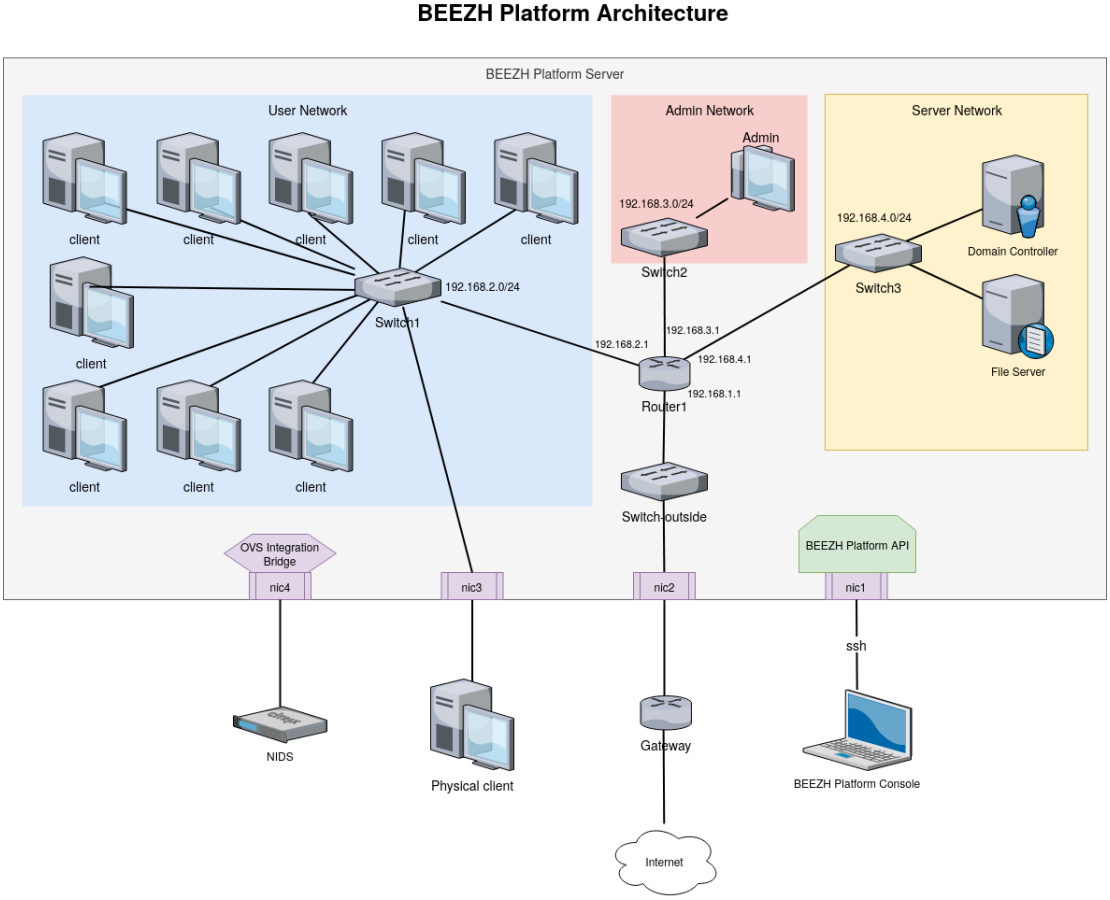

Fig. 1. Architecture réseau de la plateforme BEEZH. La plateforme permet d'instancier des sous-réseaux (utilisateurs, serveurs, administrateurs, etc.) au sein de l'environnement simulé, de relier une machine ou un sous-réseau physique, et de rediriger l'activité réseau vers une sonde externe

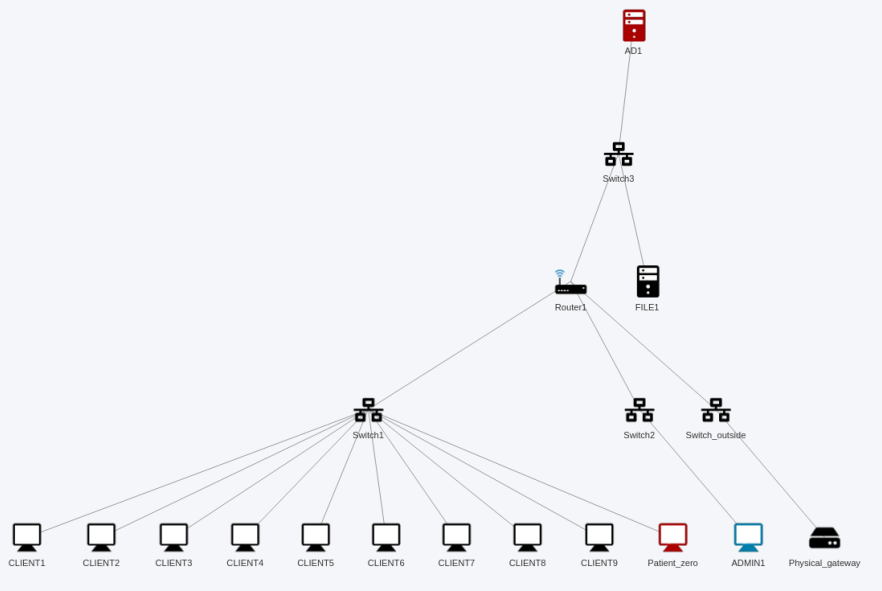

Fig. 2. Exemple de topologie réseau simulée

2.2 Avantages et limites de la virtualisation

Le composant de simulation de SI est conçu pour passer à l'échelle, grâce à l'apport de la virtualisation. Bien que le prototype actuel est prévu pour fonctionner sur un seul serveur physique (permettant ainsi de simuler une vingtaine de machines virtuelles, comme illustré sur la topologie visible sur la figure 2), des travaux sont en cours pour permettre de rattacher d'autres serveurs physiques afin d'augmenter le nombre de postes/serveurs simulés. Les technologies employées (Open vSwitch et Open Virtual Network) permettent aisément de constituer un environnement virtualisé réparti sur plusieurs serveurs physiques.

Un aspect important du piégeage est de ne pas rendre trop évident la présence du honeypot vis-à-vis d'un attaquant potentiel. Bien qu'une approche par virtualisation soit ici employée, des techniques de masquage de la présence de l'hyperviseur et de l'environnement virtualisé sont employées (renommage des périphériques avec des noms réalistes, non-utilisation des additions invitées, choix de l'adressage MAC pour caractériser des fabricants traditionnels, etc.). La problématique de la furtivité reste un sujet ouvert. Des travaux complémentaires sont en cours afin de renforcer cet aspect. Il convient néanmoins de noter que l'architecture de la plateforme permet d'associer une machine physique au SI simulé, permettant ainsi de réaliser la première phase de détonation sur un socle non virtualisé.

Enfin, du fait de l'utilisation de la virtualisation, le risque de *VM escape* (i.e. échappement par l'attaquant de l'environnement virtualisé et accès potentiel à l'hyperviseur) doit également être considéré. Des mécanismes de durcissement (pour limiter les possibilités d'exploitation) et de cloisonnement (pour qu'un attaquant maîtrisant la plateforme ne puisse pas rebondir sur le SI de l'entreprise) sont déployés. D'un point de vue opérationnel, il est également primordial d'appliquer une politique de mise à jour des composants de la plateforme.

2.3 Automatisation du provisioning et du vieillissement du honeypot

Un point important est de pouvoir simuler le vieillissement d'un système d'information, de sorte que l'attaquant ne puisse pas déduire trop rapidement qu'il est sur un environnement fraîchement déployé. Le travail consiste dans un premier temps à identifier les éléments spécifiques à l'environnement Windows permettant de caractériser l'ancienneté du système et le fait qu'il y ait eu de la vie pendant un longue période de temps. Dans un second temps, il s'agit de modéliser puis implémenter la surcouche de la capacité de *provisioning* qui permet d'assurer le vieillissement du système d'information.

La capacité de provisioning de la plateforme BEEZH permet de déployer automatiquement des ressources au sein des basebox et d'exécuter des commandes au sein des machines virtuelles, en amont de la phase opérationnelle. Ces ressources peuvent être des applicatifs, des fichiers ou des paramètres système (ex : altération de la base de registres sous Windows).

Cette capacité de provisioning est contrôlables à l'aide d'une API REST. Il est ainsi possible de déployer des ressources factices propres à l'entreprise ou l'organisme ciblé (fausse documentation avec l'entête de l'entreprise, documents faussement sensibles, etc.). Par ailleurs, une fois déployées, ces ressources sont manipulables par le composant de génération d'activité utilisateur.

Actuellement, les possibilités de configuration se concentrent notamment sur le serveur Active Directory :

- Ajout automatisé du rôle « Active Directory » sur une basebox Windows Server.
- Gestion des liens AD entre utilisateurs, unités organisationnelles et machines.
- Gestion de la notion de première connexion d'un utilisateur sur un poste
- Désactivation aléatoire de certains comptes et de la machine associée.
- Déploiement de GPO paramétrables.
- Support de l'expiration des mot de passe pour les utilisateurs.
- Capacité de simulation d'au moins 1000 utilisateurs factices sur l'annuaire AD.
- Tentatives aléatoires de connexion avec de mauvais mots de passe.
- Vieillissement des attributs de connexion des utilisateurs et machines.
- ...

Par ailleurs, le composant de provisioning facilite le déploiement de faiblesses de configuration sur le SI simulé. De la même manière, il permet l'ajout de fichiers factices, faussement sensibles, sur des partages SMB ou des répertoires utilisateur. Ces "vulnérabilités" servent alors de leurres vis-à-vis de l'attaquant, et ont pour objectif de faire exposer les TTP de celui-ci.

3 Simulation de comportements utilisateur réalistes

Cette partie présente les enjeux à produire des actions utilisateur réalistes. Nous présentons ici notre démarche et la manière dont elle se distingue des travaux issus de l'état de l'art.

3.1 Motivations

Concernant le cas d'usage de la détonation, il est courant de rencontrer des malwares s'assurant de la présence d'un vrai utilisateur derrière l'écran. Plusieurs programmes malveillants disposent en effet de routines de détection d'environnement afin d'adapter leur comportement. Par exemple, le cheval de Troie Zebrocy s'exécute uniquement à la fermeture d'un document. De même le cheval de Troie BaneChan exploite les interactions de l'utilisateur pour échapper à la détection: il

attend un nombre donné de clics souris avant de commencer à s'exécuter. La génération d'actions utilisateur devient ici cruciale pour leurrer ces programmes et récolter davantage d'informations.

Au delà du cas d'usage de la détonation, l'intérêt à produire de l'activité utilisateur réaliste est multiple. L'évaluation de produits logiciels constitue un autre domaine d'application pour lequel un environnement réaliste est nécessaire. En particulier, les produits de sécurité tels que les sondes (hôte ou réseau), les antivirus, les EDR, ou encore les SIEM, ont besoin d'être mis en conditions "réelles" pour attester de leur efficacité. En pratique, il s'agit de créer des évènements systèmes et des traces réseau mixant activité de vie et actions d'attaque pour évaluer la capacité des produits de sécurité.

La capacité à créer synthétiquement des données réalistes est également un atout pour les méthodes à base d'intelligence artificielle, que ce soit pour les phases de test ou d'apprentissage. Les techniques émergentes de détection d'intrusion se basent généralement sur des méthodes de détection d'anomalies qui font l'hypothèse que les actions malveillantes sont très minoritaires par rapport au trafic légitime. Il y a donc un besoin de reproduire un "bruit de fond" légitime. Cependant le domaine souffre d'un manque de données publiques ne serait-ce que pour évaluer les algorithmes. L'annotation d'évènements demeure une tâche longue et laborieuse qui pourrait être simplifiée par cette capacité à générer des actions de vie.

Par ailleurs, il est difficile de récolter des données sur des environnements réels (mise en place technique et prise en compte de la protection des données). De plus, les données réelles sont riches mais bien souvent non maîtrisées. Elles peuvent être contaminées par des éléments spécifiques au lieu de collecte (architectures ou services spécifiques). La contamination des données peut rendre leur usage non pertinent, que ce soit à des fins de test ou d'apprentissage (biais). La génération de données synthétiques peut ainsi instaurer un compromis entre réalisme et capacité de généralisation.

3.2 Etat de l'art

La recherche autour de la génération d'activité se concentre principalement sur la partie réseau. Plusieurs outils existent pour générer du trafic réseau, comme Iperf ou Ostinato. Ils permettent de générer des paquets réseau avec une paramétrisation plus ou moins fine selon les protocoles supportés. Ils sont utilisés principalement pour des tests de performance.

Pour la génération d'activité réaliste, d'autres outils permettent de rejouer des captures réseau (Tcpreply, TCPivo [2]). L'inconvénient est alors que ces données ne sont généralement pas réutilisables car contenant des données sensibles [5].

Pour pallier ce problème, des méthodes permettent de reproduire du trafic réseau à partir de véritable captures. Par exemple, Harpoon [9] est capable de générer de l'activité (trames TCP et UDP) à partir de l'analyse de données netflow. Swing [10] modélise quant à lui les utilisateurs et applications, et génère des données en lien avec les modèles appris. Plus de méthodes sont détaillés dans [1].

Peu de travaux ont proposé l'émulation d'activité utilisateur. Dans [11], l'idée originale des auteurs consiste à utiliser des outils de test d'interfaces graphiques. En particulier, leurs expérimentations utilisent la plateforme de test LARIAT [8], issue de précédents travaux de la DARPA. Ces mêmes travaux avaient notamment produit des données pour la détection d'intrusion (datasets DARPA98 et DARPA99). Cette approche est utilisée par les auteurs de [5] qui utilisent AutoIt pour effectuer des actions utilisateurs (clics souris, frappes clavier).

Effectuer automatiquement des actions utilisateur a l'avantage de produire tous types de données (système ou réseau) en un minimum d'action. Par exemple, la capacité à interagir avec un client mail permettra à la fois de générer des logs au niveau du système et de créer des traces réseau relatives aux protocoles utilisés lors de l'envoi et de la réception de mail (DNS, SMTP...). De plus toutes ces traces sont cohérentes, elles évitent donc un éventuel post-traitement fastidieux.

Outre les méthodes et algorithmes utilisés par les travaux les plus avancés, ces derniers font systématiquement usage d'un agent sur la machine cible, dont la responsabilité est naturellement d'émuler l'utilisateur. Un tel design permet de bénéficier de tout la puissance du système: dans [11] la génération d'activité passe notamment par l'API COM de Microsoft. Cependant cette architecture a deux principaux inconvénients:

- **Faible furtivité**. Dans le cadre d'un honeypot, un attaquant pourrait tenter de chercher des artefacts liés à cet agent pour évaluer son environnement.
- **Contamination des données**. Dans le contexte de la génération de données, il est clair que les communications avec l'agent introduisent des données non pertinentes, à supprimer par la suite. Cette contamination des données apparaît de la même manière sur les évènements système.

3.3 Démarche de simulation de comportements utilisateur réalistes

Dans ces travaux, le choix a été fait de ne pas dépendre d'un agent sur les systèmes simulés (machines virtuelles). Sans accès aux objets ou API du système hôte, la simulation de comportements utilisateurs est rendue possible via les fonctionnalités de l'hyperviseur : accès à l'affichage (déport VNC) et envoi d'évènements clavier ou souris (voir figure 3). L'interaction avec la cible est donc très similaire à celle d'un utilisateur.

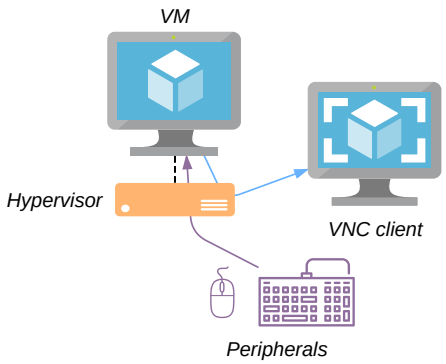

Fig. 3. Principe de simulation d'activité utilisateur au sein de la plateforme BEEZH

La simulation de comportements utilise ainsi la reconnaissance d'image pour envoyer des actions utilisateurs. En particuliers, nous détaillons deux méthodes pour l'émulation:

- **actions pré-définies**. Utilisation d'un framework pour implémenter un catalogue d'actions unitaires et permettant de construire des scénarios séquentiels.
- **apprentissage**. Génération d'actions utilisateurs réelles (clic souris uniquement) et apprentissage par réseau de neurones.

3.3.1 Catalogue d'actions et construction de scénarios. Au sein du framework, une action utilisateur correspond à une succession de mouvements de souris et/ou de frappes clavier afin d'obtenir un effet élémentaire. Le catalogue mis en oeuvre est constitué d'un ensemble d'actions permettant d'obtenir ces effets. Les principales actions sont les suivantes : `ouverture de session`, `fermeture de session`, `ouverture de l'explorateur de fichiers`, `navigation dans les répertoires`, `ouverture d'un fichier`, `enregistrement d'un fichier ouvert`, `navigation dans un fichier`, `navigation sur une URL`, `écriture de données`, `exécution de commandes`, etc.

Sur la base de ce catalogue, il est possible de spécifier un scénario d'actions unitaires à séquencer. Un exemple de scénario basique est le suivant : `[ouverture de session, ouverture de l'explorateur de fichiers, ouverture d'un fichier, écriture de données dans ce fichier, enregistrement du fichier, fermeture de session]`.

Il est par ailleurs possible de savoir si une action a réussi. Cette information peut être nécessaire pour enchaîner des actions dépendantes entres elles. Le langage d'écriture des scénarios permet également de disposer de boucles et de conditions booléennes. De fait, il est possible de construire des répétitions d'enchaînements d'actions ou d'avoir des actions de repli en cas d'impossibilité d'exécution d'une action.

Le moteur d'exécution des actions de vie autorise l'application de scénarios utilisateur en parallèle sur un ensemble de cibles (voir figure 4). Il est ainsi possible de jouer de la vie sur tout un parc.

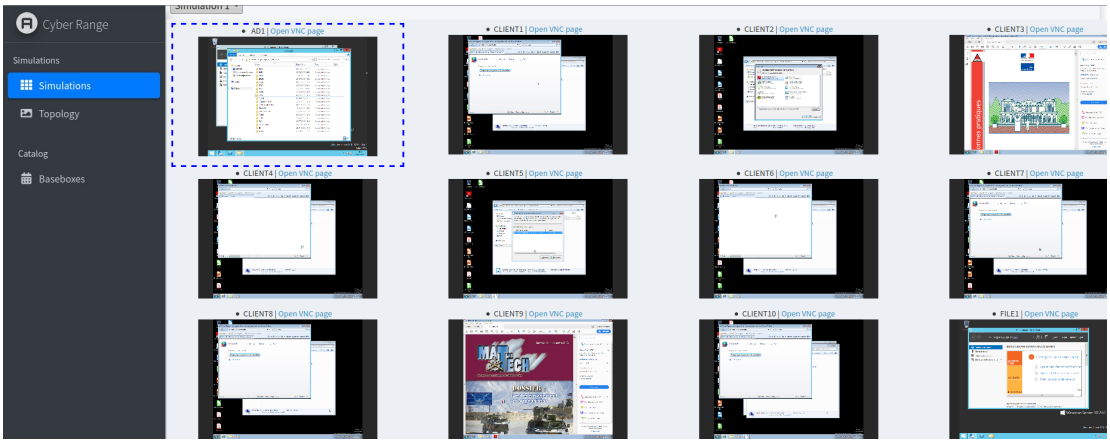

Fig. 4. Vue matricielle d'exécution des actions de vie en parallèle sur un ensemble de cibles Windows

Afin de construire le catalogue, chaque action doit être implémentée en amont. Cette tâche est à accomplir pour chaque OS et applicatifs cibles. Cette approche à l'avantage d'être totalement maîtrisée, ce qui signifie que l'on peut précisément évaluer son impact sur le système cible : pour une action unitaire donnée (par exemple, une ouverture de session), on peut en déduire l'ensemble des évènements qu'elle a générés. Dans un environnement de type honeypot, cette connaissance des traces produites par les actions simulées permet ainsi de faire émerger les évènements en lien avec une attaque.

3.3.2 Apprentissage utilisateur. Dans cette partie nous explorons la possibilité d'apprendre un comportement utilisateur. Dans notre contexte, les captures d'écran constituent les données

d'entrée. Simuler totalement un utilisateur (clics, double-clics, frappes clavier, enchaînement d'actions cohérentes...) est une tâche difficile c'est pourquoi nous nous concentrons sur une phase initiale fondamentale: la détection des zones d'intérêts (fichiers, dossiers, icônes, boutons...), soit les zones où un utilisateur est susceptible de cliquer.

Les méthodes de détection d'objets sont principalement issues de l'apprentissage profond (*deep learning*) et utilisent principalement des réseaux de neurones convolutifs (CNN). En plus de leur efficacité, ces techniques ne nécessitent que d'un faible pré-traitement sur les images. On distingue notamment trois modèles effectuant de la détection d'objets :

1. Region Proposals (Faster R-CNN) [7]
2. You Only Look Once (YOLO) [6]
3. Single Shot MultiBox Detector (SSD) [4]

Nos tests nous ont incités à utiliser Faster-R-CNN qui propose le meilleur compromis vitesse/précision. En pratique nous utilisons la bibliothèque PyTorch qui dispose d'une implémentation de Faster R-CNN basée sur le *backbone* ResNet-50-FPN [3] pré-entraînées sur COCO (2017).[1] La figure 5 synthétise notre approche pour la détection de zones d'intérêt à partir de captures d'écran.

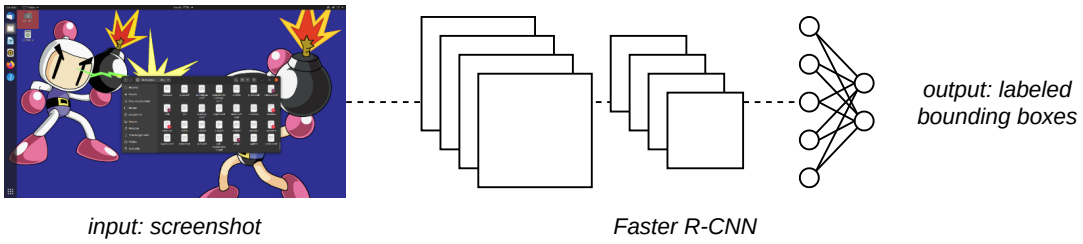

Fig. 5. Schéma de la solution d'apprentissage des zones d'intérêt

Dataset. Les objets à détecter sont naturellement différents de ceux présentés dans les données de pré-entraînement. Ces dernières permettent en particulier d'optimiser les premières couches du CNN pour l'extraction d'attributs (*feature extraction*). En pratique ces couches sont figées et seules les dernières sont optimisées sur les données d'intérêt (*fine-tuning*).

Pour l'optimisation des dernières couches, nous avons constitué un jeu de 70 images annotées pour détecter une dizaine objets d'intérêt sur une capture d'écran: fichier, dossier, bouton retour, bouton de fermeture, icône de Firefox... Ce nombre d'images peut sembler faible mais il est suffisant pour adapter Faster R-CNN à notre problème et permettre une généralisation à des environnements non connus. Par ailleurs, ce *dataset* peut facilement être enrichi (ajout d'images ou de nouvelles classes) mais le temps nécessaire à l'annotation peut être conséquent.

Entraînement & Performances. Pour nos expérimentations, nous avons utilisé une carte graphique NVIDIA GTX 1660, disposant de 6Go de mémoire. Le modèle nécessitant environ 4.5Go de mémoire, l'entraînement ne peut pas utiliser un nombre d'images (*batch*) trop important à chaque pas d'optimisation (< 5). Avec 70 images, un passage dans complet dans le jeu de données (1 *epoch*) nécessite par ailleurs moins de 30 secondes. Le modèle entraîné peut ainsi

[1] https://cocodataset.org/#home

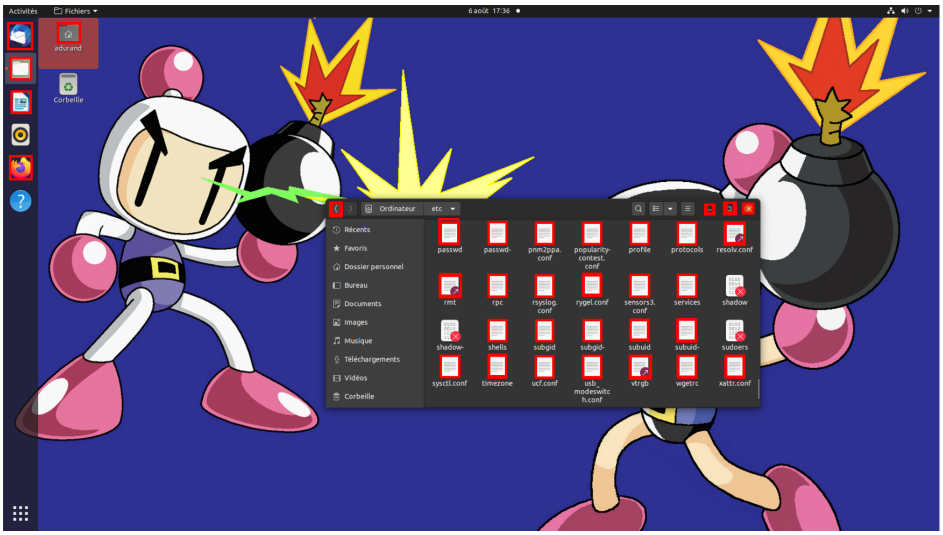

Fig. 6. Exemple de prédiction: les rectangles rouges (*boxes*) désignent les zones d'intérêt détectées. L'algorithme fournit naturellement un label pour chacune de ces zones

effectuer environ 5 prédictions par seconde, ce qui est adapté à notre contexte de simulation d'utilisateur. Un exemple de prédiction est présenté en figure 6.

Autres fonctionnalités. L'algorithme présenté précédemment est un composant d'une bibliothèque plus générale que nous développons activement: **`desker`** (*DESKtop Element Recognition*). Elle présente d'autres fonctionnalités telles que l'extraction de texte, la détection de liens, la détection d'écran de *login*... Ces composants sont également employés dans la construction de scénarios de vie utilisateur.

4 Extraction des traces d'intérêts pour la production de renseignements sur la menace

Lors d'une opération de détonation, la production de renseignements sur la menace nécessite d'identifier les traces système et réseau produites par les actions de l'attaquant, puis d'en décliner le mode opératoire. Sur un environnement honeypot, dans lequel de l'activité légitime est également jouée, cela revient à différencier ces deux types d'activité.

Dans nos travaux, le processus d'extraction des traces d'intérêts repose sur un lien fort avec le composant de simulation d'activité utilisateur. Le principe suivi est de capturer un ensemble d'évènements système et réseau grâce à des sondes de collecte sur les systèmes d'exploitation et sur le réseau simulé, puis de séparer les logs liés à l'activité de vie des logs liés à l'attaque. L'hypothèse suivie est que cette séparation est possible dès lors que les actions de vie sont maîtrisées.

Le processus est construit autour de trois étapes :

1. Production des métadonnées des actions de vie.
2. Production du dataset des traces système et réseau.
3. Extraction des traces d'attaque.

4.1 Production des métadonnées des actions de vie

Lors de la simulation des actions utilisateur, des métadonnées liées à l'exécution de ces actions sont sauvegardées. Ces métadonnées contiennent les *timestamps* de début et de fin des actions de vie, ainsi que les éléments de paramétrage de ces actions (fichiers ouverts, URL accédées, login/mot de passe lors d'une ouverture de session, etc.). Concernant les actions de vie produites à partir du moteur de clic, les métadonnées sauvegardées contiennent le timestamp de chaque action unitaire (i.e. l'instant du clic).

L'extrait suivant présente un exemple de rapport intégrant les métadonnées des actions de vie jouées sur l'environnement cible. Cet extrait illustre une ouverture de session par un utilisateur, l'accès à l'explorateur Windows, l'ouverture de documents et la fermeture de la session :

```
action 'OpenSession', starting at: 17:52:29.590, ending at: 17:53:42.354
action 'RandomSleep', starting at: 17:53:42.356, ending at: 17:53:49.357
action 'MoveMouseRandom', starting at: 17:53:49.366, ending at: 17:53:59.368
action 'OpenFileExplorer', starting at: 17:53:59.376, ending at: 17:54:03.112
action 'NavigateInFileExplorer', starting at: 17:54:03.116, ending at: 17:54:07.851
action 'SelectRandomFile', starting at: 17:54:07.854, ending at: 17:54:38.078
action 'MoveMouseRandom', starting at: 17:54:38.080, ending at: 17:54:48.082
action 'MoveInDocument', starting at: 17:54:48.091, ending at: 17:55:18.093
action 'MoveMouseRandom', starting at: 17:55:18.095, ending at: 17:55:28.097
action 'Open', starting at: 17:55:28.105, ending at: 17:55:34.646
action 'Sleep', starting at: 17:55:34.653, ending at: 17:55:49.653
action 'MoveMouseRandom', starting at: 17:55:49.655, ending at: 17:55:59.656
action 'MoveInDocument', starting at: 17:55:59.658, ending at: 17:56:29.659
action 'MoveMouseRandom', starting at: 17:56:29.660, ending at: 17:56:39.661
action 'MoveInDocument', starting at: 17:56:39.662, ending at: 17:57:09.663
action 'MoveMouseRandom', starting at: 17:57:09.664, ending at: 17:57:19.665
action 'OpenFileExplorer', starting at: 17:57:19.666, ending at: 17:57:23.072
action 'NavigateInFileExplorer', starting at: 17:57:23.073, ending at: 17:57:26.930
action 'SelectRandomFileCurrentDir', starting at: 17:57:26.933, ending at: 17:58:11.531
action 'MoveMouseRandom', starting at: 17:58:11.534, ending at: 17:58:21.535
action 'MoveInDocument', starting at: 17:58:21.537, ending at: 17:58:51.538
action 'MoveMouseRandom', starting at: 17:58:51.539, ending at: 17:59:01.540
action 'CloseSession', starting at: 17:59:01.547, ending at: 17:59:24.966
```

Par ailleurs, les logs produits par l'activité des services système des OS est également prise en compte comme trace de vie légitime.

4.2 Production du dataset des traces système et réseau

Comme le montre la figure 7, en parallèle de la génération des métadonnées relatives aux actions de vie, l'ensemble de l'activité système et réseau est capturée.

La collecte des évènements système est réalisée à l'aide d'agents Sysmon et Nxlog déployées sur chaque machine (postes de travail et serveurs). Bien que potentiellement suspectes, à priori, pour un attaquant, il convient de noter que ce type d'agent de collecte, à l'instar des EDR, apparaît de plus en plus déployé sur des SI opérationnels ces dernières années. La présence de tels agents sur un honeypot peut aujourd'hui être vue comme légitime.

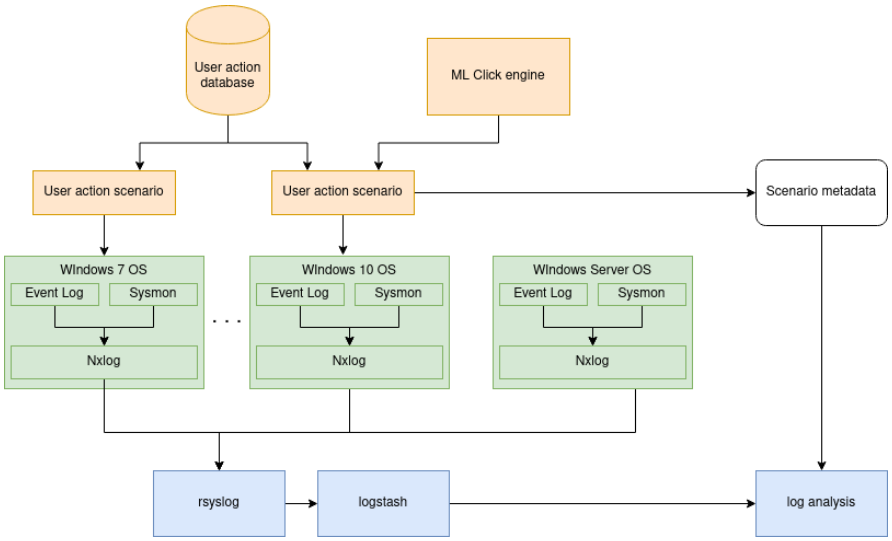

Fig. 7. Processus de génération du dataset des traces système

L'agent Sysmon est paramétré avec un fichier de configuration s'inspirant de celui proposé par SwiftOnSecurity. Le module de capture Nxlog `im_msvistalog` est configuré pour collecter les évènements Windows provenant des journaux d'évènements suivants :

- `Application/*`
- `System/*`
- `Security/*`
- `Microsoft-Windows-Sysmon/Operational/*`
- `Microsoft-Windows-Windows Defender/Operational/*`

Ce choix de configuration, en complément du déploiement de GPO d'audit spécifiques, permet de collecter les traces laissées par un spectre important de techniques d'attaque. En particulier, les types d'évènements suivants sont notamment pris en compte : ouverture/fermeture de session, accès distants, création de processus, chargement de DLL, création de thread distant, accès au système de fichiers, résolutions DNS, écriture dans la base de registre, alertes de sécurité, ...

Ces évènements sont ensuite redirigés vers un puits de collecte rsyslog, destinés à centraliser les logs provenant de l'ensemble du SI simulé.

Dans un second temps, une phase de normalisation et de filtrage est appliquée sur les évènements collectés par le composant rsyslog. La normalisation et le filtrage sont réalisés avec l'outil logstash.

En ce qui concerne la collecte des évènements réseau, celle-ci est réalisée par le composant de virtualisation réseau OVN qui permet de router les flux, de manière transparente vis à vis du SI simulé, vers une sonde de collecte réseau.

4.3 Extraction des traces d'attaque

L'analyse des logs est ensuite réalisée sur le dataset produit. Comme le montre la figure 8, les étapes d'analyse sont les suivantes :

1. Labellisation des logs liés aux actions de vie du catalogue, puis suppression de ces logs. La labellisation est ici réalisée en s'appuyant sur trois éléments : les timestamp de début et de fin de l'action, les métadonnées de l'action ainsi que sur une connaissance à priori des logs produits par chaque action. Cette connaissance est obtenue lors d'une phase d'apprentissage en amont, où chaque action du catalogue est jouée unitairement dans un environnement avec le minimum de bruit parasite. Lors de cette phase d'apprentissage, les évènements des services systèmes et de l'OS sont également traités.
2. Labellisation des logs liés à l'exploitation des vulnérabilités volontairement déployées sur le système d'information simulé. La labellisation peut ici porter sur des objets Active Directory accédés, des fichiers leurres accédés ou encore sur la capture de flux réseau sur des ports précis en écoute.
3. Labellisation des logs liés aux actions du moteur de clic, puis suppression éventuelle de ces logs suivant le niveau de confiance accordé à cette labellisation. Le calcul du niveau de confiance repose sur la pondération de l'évènement ; cette pondération étant plus forte lorsque le log produit est proche du timestamp de l'action.

Ce processus d'analyse en trois étapes permet de produire un ensemble de logs dont certains dénotent des actions d'attaques connues et donc labellisées, des actions de vie où le niveau de confiance sur la labellisation est modéré, ainsi que des logs liés à des actions d'attaques non connues. L'analyste peut alors prendre le relai et travailler sur ce dataset épuré et en partie labellisé. Avec cette base de travail, l'analyste peut avoir notamment comme objectif de comprendre les TTP employés par l'attaquant, reconstruire la *kill chain* (par exemple, relier deux attaques connues avec une attaque non connue, nécessaire pour relier les attaques connues) et potentiellement identifier/caractériser de nouvelles méthodes d'attaque.

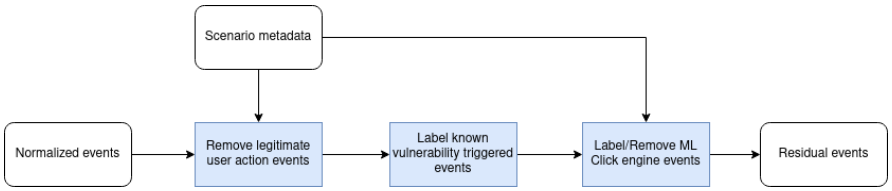

Fig. 8. Workflow d'extraction des traces d'intérêts depuis le dataset

5 Conclusion et perspectives

A travers cet article, nous avons vu l'importance du caractère réaliste d'un environnement honeypot lorsque celui-ci est utilisé dans un cas d'usage de détonation. Ce réalisme sert à exposer les techniques et modes opératoires (TTP) des attaquants. Avec la plateforme BEEZH, il est possible de construire, de manière automatisée, un environnement SI crédible, provisionné et vieilli. Sur cette base, nous avons proposé une contribution visant à produire des comportements utilisateur réalistes sur le SI simulé, dans l'objectif de faire croire à l'attaquant que des utilisateurs humains se cachent derrière les écrans. Avec cet environnement de détonation, nous avons également proposé un procédé permettant d'extraire une approximation des traces d'attaque, facilitant ainsi le travail des analystes pour la compréhension du mode opératoire des groupes d'attaquants.

En termes de perspectives, des améliorations techniques sont à implémenter, notamment en ce qui concerne la furtivité de la virtualisation et le passage à l'échelle, ainsi qu'une meilleure prise

en compte des évènements réseau dans le processus d'extraction des traces d'intérêts. D'un point de vue métier, il serait pertinent d'établir un rapprochement entre la capacité de honeypot de la plateforme BEEZH et son utilisation comme mécanisme de détection sur un SI opérationnel. Cette plateforme pourrait en effet permettre de lever une alerte auprès d'un SIEM lorsqu'un attaquant pénètre sur la portion du SI qui est simulée par le honeypot. Sur ce sujet, plusieurs problématiques se posent, dont notamment la difficulté d'intégration du SI simulé au sein d'un parc de grande dimension.

Bibliographie

[1] Botta, A., Dainotti, A. and Pescapé, A. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*. 56, 15 (2012), 3531–3547.

[2] Feng, W.-c., Goel, A., Bezzaz, A., Feng, W.-c. and Walpole, J. 2003. TCPivo: A high-performance packet replay engine. *Proceedings of the acm sigcomm workshop on models, methods and tools for reproducible network research* (2003), 57–64.

[3] He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep residual learning for image recognition. *Proceedings of the ieee conference on computer vision and pattern recognition* (2016), 770–778.

[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C. 2016. Ssd: Single shot multibox detector. *European conference on computer vision* (2016), 21–37.

[5] Megyesi, P., Szabó, G. and Molnár, S. 2015. User behavior based traffic emulator: A framework for generating test data for DPI tools. *Computer Networks*. 92, (2015), 41–54.

[6] Redmon, J. and Farhadi, A. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. (2018).

[7] Ren, S., He, K., Girshick, R. and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* (2015), 91–99.

[8] Rossey, L.M., Cunningham, R.K., Fried, D.J., Rabek, J.C., Lippmann, R.P., Haines, J.W. and Zissman, M.A. 2002. Lariat: Lincoln adaptable real-time information assurance testbed. *Proceedings, ieee aerospace conference* (2002), 6–6.

[9] Sommers, J. and Barford, P. 2004. Self-configuring network traffic generation. *Proceedings of the 4th acm sigcomm conference on internet measurement* (2004), 68–81.

[10] Vishwanath, K.V. and Vahdat, A. 2009. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking*. 17, 3 (2009), 712–725.

[11] Wright, C.V., Connelly, C., Braje, T., Rabek, J.C., Rossey, L.M. and Cunningham, R.K. 2010. Generating client workloads and high-fidelity network traffic for controllable, repeatable experiments in computer security. *International workshop on recent advances in intrusion detection* (2010), 218–237.