Simulation réaliste d'utilisateurs pour les systèmes d'information en Cyber Range

Alexandre Dey, Benjamin Costé, Éric Totel, Adrien Bécue

Simulation réaliste d'utilisateurs pour les systèmes d'information en Cyber Range

Alexandre Dey[1,2], Benjamin Costé[1], Éric Totel[3] et Adrien Bécue[1]

[1] Airbus CyberSecurity, Rennes, France
[2] IRISA, Rennes, France
[3] Télécom SudParis, Paris, France
alexandre.dey@airbus.com
benjamin.b.coste@airbus.com
eric.totel@telecom-sudparis.eu
adrien.becue@airbus.com

Résumé. La génération d'activité utilisateur est un élément-clé autant pour la qualification des produits de supervision de sécurité que pour la crédibilité des environnements d'analyse de l'attaquant. Ce travail aborde la génération automatique d'une telle activité en instrumentant chaque poste utilisateur à l'aide d'un agent externe; lequel combine des méthodes déterministes et d'apprentissage profond, qui le rendent adaptable à différents environnements, sans pour autant dégrader ses performances. La préparation de scénarios de vie cohérents à l'échelle du SI est assistée par des modèles de génération de conversations et de documents crédibles.

Mots-clé: Cyber range · génération de texte · reconnaissance d'images · simulation de vie · jeux de données · honeynet

1 Introduction

Les avancées récentes et la démocratisation des technologies de virtualisation (e.g., *Software Defined Network* ou SDN, Cloud) ont notamment permis l'essor d'outils dédiés au cyber-entraînement. Appelées communément Cyber Ranges, ces plateformes facilitent le déploiement de Systèmes d'Information (SI) complets pour l'organisation de formations et exercices à destination des opérationnels de cyberdéfense. Outre leurs qualités pédagogiques, elles constituent également une base solide pour l'évaluation et la mise au point des outils de supervision de sécurité [9] (sonde de détection d'intrusion, détection d'anomalies, SIEM, etc.), ainsi que la constitution d'environnements d'analyse des attaquants (*honeynet* et plateformes de détonation).

La simulation d'activité sur les postes utilisateur apporte une crédibilité nécessaire aux plateformes d'analyse des attaquants tout en permettant d'évaluer le comportement des produits de sécurité face au fonctionnement nominal des SI. En effet, les jeux de données disponibles pour l'apprentissage machine dans le domaine de la cybersécurité représentent des attaques, plus ou moins réalistes,

mais n'intègrent pas ou peu de comportement illégitimes (scan de ports, branchement de clés USB sur des postes sensibles, etc.) émanant d'activités d'utilisateur légitimes. Cette absence d'activité, pourtant foisonnante sur un SI réel, complexifie l'immersion dans le cas du cyber-entraînement, biaise les données collectées sur les terminaux pour l'entraînement de méthodes de supervision basées sur l'apprentissage machine, et diminue grandement la crédibilité des plateformes d'analyse des attaquants.

La génération automatique de vie réaliste, objet notamment du challenge IA & Cyber 2020 de l'ECW, est un sujet complexe qui demande de résoudre plusieurs problèmes. Tout d'abord, l'instrumentation des machines doit se faire par des méthodes extérieures à celles-ci afin de limiter les traces de simulation laissées sur les postes utilisateurs. En second lieu, il est nécessaire d'adapter en temps réel le scénario pré-établi aux réactions aléatoires de l'environnement de simulation (e.g., position d'une fenêtre, arrêt imprévu). Cette adaptation se fait via des vérifications automatiques de l'environnement qui doivent par ailleurs être effectuées en un temps inférieur ou égal au temps de réaction humain. Enfin, une assistance à l'opérateur est nécessaire pour la mise au point de scénarios à grande échelle.

Ce travail s'inspire des méthodes de génération de vie, reposant sur un agent, employées par la plateforme de détonation BEEZH, présentée par Amossys à la conférence C&ESAR 2020 [7], pour lesquelles nous proposons plusieurs améliorations :

- Découpage de l'agent en plusieurs couches d'abstraction successives pour gagner en modularité (e.g., s'affranchir de la technologie de virtualisation, adaptabilité à des machines physiques, etc.);
- Combinaison efficiente de méthodes déterministes et d'apprentissage profond pour la réaction en temps réel de l'agent;
- Assistance à la création d'actions exécutables par l'agent;
- Assistance à la création de scénarios de vie à partir des profils des utilisateurs simulés et de leurs interactions.

Dans la section 2, nous présentons l'état de l'art. Les sections 3 et 4 décriront les travaux réalisés. Nous conclurons en discutant notre approche en sections 5 et 6.

2 État de l'art

De nombreuses études ont souligné l'importance de la simulation d'activité utilisateur pour l'étude des logiciels malveillants [4, 1]. Ainsi, l'absence de plusieurs artefacts (e.g., présence de fichiers dans la corbeille, présence de cookies, historique de navigation) permet d'identifier les machines n'ayant jamais été utilisées [15]. Certains logiciels malveillants contournent les systèmes de détection par le biais de *tests de Turing inversés* tels que des mécanismes à retardement (scroll dans un document Word) ou la vérification de la vitesse d'utilisation de la souris

[25]. La simulation en temps réel d'actions utilisateurs permet cependant de mettre en évidence le comportement malveillants de ces logiciels.

L'utilisation automatisée de l'interface graphique est un sujet historiquement étudié pour le contrôle qualité des logiciels. Deux approches s'y distinguent : le rejeu d'actions pré-enregistrées [27, 24, 16], long à configurer, et l'automatisation complète qui souffre d'un manque de réalisme [23, 6]. Chacune de ces méthodes requiert un agent installé sur les machines instrumentées, ce qui fournit un indicateur à l'attaquant, et teinte les journaux d'événements dans le cas de la collecte de jeux de données.

Plus récemment, MORRIGU [14] instrumente des machines virtuelles via l'API VirtualBox depuis un hôte Windows. Malgré ses résultats positifs sur la détection de comportements malveillants, le manque de portabilité de la solution ne permet pas d'envisager son emploi à des fins de cyber-entraînement qui implique l'instrumentation de multiples machines en réseau.

L'outil de simulation des utilisateurs intégré à la plateforme BEEZH [7] instrumente des machines virtuelles au travers de la fonction de déport d'écran de l'hyperviseur, reposant sur la technologie VNC. Les scénarios de vie sont découpés en actions unitaires simples (e.g., ouvrir le navigateur, recherche web) dont l'exécution est assistée par des méthodes d'analyse d'images (captures d'écran) à l'aide de la librairie desker [2], publiée sous licence GPL v3. Cette approche pionnière a montré l'importance mais également la complexité du sujet. Le recours à VNC offre en effet un large éventail de possibilités vis-à-vis de l'interaction avec la machine instrumentée mais se restreint néanmoins à l'instrumentation de machines virtuelles. La création des scénarios et leur adaptation aux spécificités de l'environnement nécessitent de surcroît des actions entièrement manuelles. Enfin, la solution d'analyse d'image retenue (Faster-RCNN [22]) sollicite d'importantes ressources de calcul incompatibles avec notre contrainte de performance. Faster-RCNN [22] est un algorithme très répandu (car pertinent) pour la détection de zones d'intérêt. D'autres initiatives plus récentes telles que RetinaNet [12] ou SSD [13] fournissent cependant de meilleurs résultats en un temps plus court.

La génération automatique de texte repose fréquemment sur les modèles de Markov cachés [8] (HMM, *Hidden Markov Model*) mais les résultats produits deviennent rapidement incohérents lorsque la séquence générée s'allonge. Des solutions à base de réseaux de neurones comme Transformer [26] apparaissent plus adaptés à la modélisation du langage. Les modèles basés sur le système GPT [18], performants et polyvalents, produisent toutefois des modèles massifs (e.g., 11 milliards de paramètres pour T5 [20], 175 milliards pour GPT-3 [3]). Au regard de leurs performances, nos travaux s'inspirent de GPT-2 [19] (117 millions de paramètres) et CTRL [10] pour la génération conditionnelle de texte.

3 Exécution et enregistrement d'actions

L'instrumentation de machines virtuelles ou physiques repose sur la conception d'un agent externe (Section 3.1) capable de reconnaître son environnement et

de s'y adapter en conséquence (Section 3.2). La réalisation d'actions complexes pré-enregistrées (Section 3.3) est également une fonctionnalité recherchée.

3.1 Conception de l'agent

Afin de simplifier la configuration du générateur de vie par les opérateurs, nous avons choisi une approche en couches d'abstraction (Fig. 1). En effet, une approche de bout-en-bout devient difficile à maintenir le nombre de scénarios grandissant (e.g., code dupliqué) et impose à l'opérateur de définir manuellement un grand nombre de détails lorsqu'il crée un scénario. Les scénarios d'activité utilisateur sont ainsi transcris par un agent en actions bas niveau (clavier, souris et écran) qu'il effectue ensuite sur la machine instrumentée.

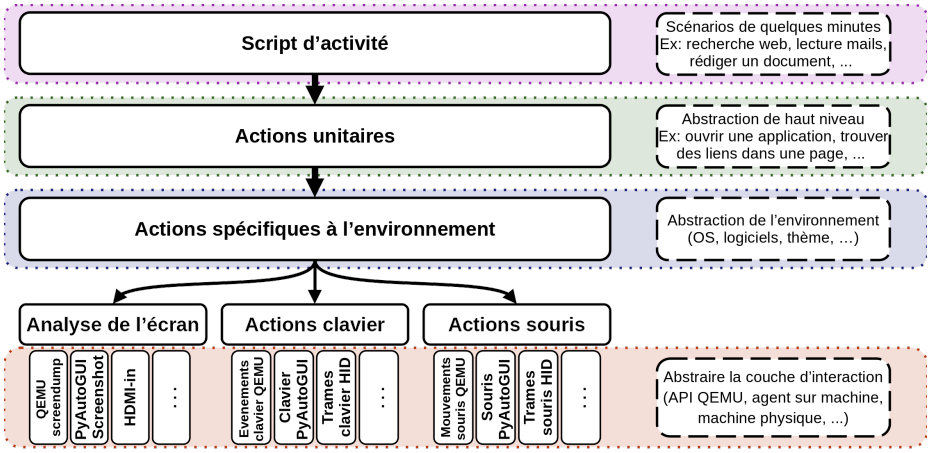

Fig. 1. Architecture fonctionnelle de l'agent

Au plus bas niveau, l'utilisateur virtuel interagit avec la machine instrumentée par le biais de la souris, du clavier et de l'écran. L'objectif est de pouvoir, sans modifier les scénarios de vie, adapter l'agent à plusieurs plateformes de virtualisation (e.g. VMWare ESXi, Proxmox, etc.) mais également à des machines physiques. Nous avons considéré pour cela trois méthodes d'interaction différentes mais complémentaires : l'API Qemu Monitor[4], une connexion VNC et un agent installé sur la machine à instrumenter. Concernant les machines physiques, l'approche la plus pertinente semble la communication des actions clavier et souris par USB selon le protocole HID[5] en capturant la sortie vidéo.

Pour contrer des méthodes d'évasion de sandbox employées par certains logiciels malveillants, nous rajoutons de l'aléa dans les frappes clavier et les mouvements de souris. Pour ces derniers, chaque déplacement est découpé en une

[4] https://qemu-project.gitlab.io/qemu/system/monitor.html
[5] https://www.usb.org/hid

multitude de petits déplacements, et la vitesse varie en fonction de la distance à parcourir (i.e., un mouvement court sera moins rapide qu'on mouvement long). Ceci fluidifie le mouvement (augmentant le réalisme) et trompe les malware qui détectent les déplacements instantanés de la souris. De l'aléa et une inertie dans le mouvement sont rajoutés pour éviter les mouvements de souris rectilignes et trop précis (détectés par certaines méthodes d'évasion). Une latence aléatoire est également ajoutée entre chaque frappe clavier afin de rendre inopérantes les méthodes de détection recherchant une régularité. Enfin, lorsque l'utilisateur virtuel est en attente (e.g., lors de la lecture d'un texte, ou lorsque l'agent effectue un calcul long), des mouvements aléatoires de souris sont effectués pour simuler les mouvements spontanés lorsque la main est posée sur la souris. Les modèles d'aléa présentés ici visent à contrer les mécanismes de sécurité les plus simples. Ces modèles pourront être complexifiés par la suite afin de modéliser plus finement des utilisateurs humains (e.g., vitesse de frappe différente d'un utilisateur à un autre).

La seconde couche dite d'interaction permet de s'adapter aux spécificités de l'environnement de la machine instrumentée. En effet, bien que les méthodes de reconnaissance d'images utilisées (section 3.2) soient peu sensibles aux variations graphiques mineures (e.g., résolutions d'écran différentes, couleurs légèrement différentes, etc.), certaines variations demandent de modifier les images à cibler. Par exemple, les actions à effectuer pour envoyer un mail avec Thunderbird ou Outlook sont fonctionnellement similaires mais les éléments d'interface sont suffisamment différents pour nécessiter des interactions différentes avec la machine.

La couche supérieure décrit des actions unitaires simples (e.g., ouvrir le navigateur web, rechercher un mot clé, identifier les liens dans du texte, etc.) et sert d'interface de programmation (API) pour les scénarios. Les scénarios ainsi créés constituent l'ultime couche d'abstraction.

3.2 Analyse des captures d'écran

L'analyse des captures d'écran permet à un agent de contrôler son état en temps réel par reconnaissance des zones d'intérêts, telles les boutons d'interface à cliquer, les liens dans une page web, etc. Pour garantir l'efficience calculatoire, primordiale dans notre contexte, nous limitons systématiquement la quantité d'information (i.e. taille et nombre d'images) et la complexité des modèles statistiques. Trois techniques de reconnaissance d'image sont ainsi employées. La première, connue sous le nom de *template matching*, consiste à trouver dans une image (ici une capture d'écran) le ou les éléments correspondant le plus à une cible recherchée. Bien que de faibles variations (e.g., résolution, couleurs, etc.) suffisent à la rendre inopérante, cette méthode a l'avantage d'être peu coûteuse en ressources et d'être déterministe, ce qui la rend appropriée à la détection d'éléments d'interface qui varient peu voire pas pour un même environnement (e.g., bouton du menu démarrer, des fenêtres, etc.).

Dans les cas où le *template matching* n'est pas satisfaisant, nous proposons d'employer des algorithmes d'apprentissage profond ayant montré des performances remarquables pour la détection d'objet. Les méthodes les plus efficientes

telles SSD [13] ou YOLO [21] requièrent cependant d'importantes ressources calculatoires qui limitent le passage à l'échelle (plusieurs machines instrumentées). De plus, il n'existe pas de jeux de données publics contenant des captures d'écran avec les zones d'intérêt détourées et annotées. Collecter et annoter un tel jeu de données prend un temps considérable. Pour ces raisons, nous avons choisi une approche qui exploite la géométrie des formes à repérer à l'écran (Fig. 2) en appliquant une méthode de seuillage adaptatif [11]. Les objets au premier plan (e.g., icônes, texte) ainsi que les séparations entre les différentes zones à l'écran (e.g., contour des fenêtres) deviennent ainsi simples à identifier. En appliquant des règles de filtrage (e.g., une icône est à peu près aussi large que haute, un bouton sera plutôt plus large que haut, etc.) il est possible, pour un coût calculatoire faible, d'identifier les zones d'intérêt. Des modèles statistiques (e.g., réseaux de neurones à convolution) filtrent les potentiels faux positifs et classifient les autres.

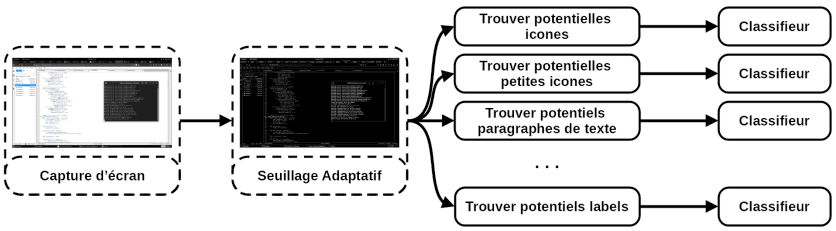

Fig. 2. Méthodologie d'analyse d'images

Enfin, la reconnaissance de caractères est utilisée pour la navigation dans l'interface (e.g., label des fichiers et dossiers, éléments des menus, etc.) ainsi que la reconnaissance des liens dans les pages web. Notre implémentation utilise la librairie tesseract [17] pour l'analyse des éléments textuels de l'interface, et reprend la même méthode que desker [2] pour la détection de liens.

3.3 Aide à la création d'actions

La reconnaissance d'image présentée dans la section 3.2 donne à l'agent la capacité de reconnaître l'environnement dans lequel il évolue et s'y adapter. Par exemple, l'interface de Firefox diffère de celle de Chrome, bien que les fonctionnalités de ces navigateurs soient équivalentes. Par conséquent, l'agent cherchera des repères visuels différents pour les manipuler. Nous proposons un outil de création d'actions qui exploite ces similarités pour faciliter la collecte de données destinées aux techniques de reconnaissance d'images présentées plus haut.

Cet outil enregistre les mouvements et clics de la souris, les frappes du clavier ainsi que des captures d'écran. Ces données brutes sont ensuite agrégées pour découper l'enregistrement en petites actions (e.g., mouvement de la souris de A vers B, puis double clic gauche sur B). Les zones d'intérêts dans les captures

d'écran sont extraites selon deux méthodes différentes. La première consiste à comparer une image prise au moment d'un clic de souris avec l'images prise au début du mouvement précédent ce clic. La plupart des interfaces graphiques actuelles mettent en surbrillance les éléments survolés par la souris ce qui permet d'extraire simplement ces éléments, qui serviront de cibles de *pattern matching*. La seconde méthode consiste à employer la technique d'identification des zones potentiellement intéressantes décrite dans la section 3.2 (Fig. 2) ce qui accélère grandement l'annotation des jeux de données pour l'entraînement de modèles de reconnaissance de zones d'intérêt.

4 Gestion de scénarios de vie à l'échelle du système

Les outils et méthodes présentées permettent à notre agent d'interagir et de s'adapter à son environnement. La présente section aborde la réalisation et l'exécution de scénarios de complexités variées à l'échelle d'un SI comportant plusieurs machines et utilisateurs.

4.1 Orchestration des communications entre utilisateurs

Dans le contexte du cyber-entraînement, pour améliorer la cohérence et simplifier la mise en place de l'exercice, il est intéressant de créer des avatars pour chacun des utilisateurs simulés et de leurs interactions. En fonction de leurs rôles dans l'entreprise et des relations qu'ils entretiennent avec leurs collègues, les employés vont interagir différemment avec le système d'information simulé. Par exemple, deux collègues amis en dehors de leur travail sont plus susceptibles de discuter par mail de manière informelle que deux collègues ne se connaissant pas. Similairement, un développeur utilisera plus souvent l'éditeur de code qu'un manager.

Nous générons des canevas de scénarios de vie à l'échelle du système par le biais d'un graphe relationnel des utilisateurs de ce système (Fig. 3). Ce graphe est composé des différents avatars, des projets sur lesquels ils travaillent, de leurs groupes de travail et de la nature des relations qu'ils entretiennent entre eux (e.g., amicales, client-fournisseur, partenaires, lien hiérarchique, etc.).

4.2 Modèles de génération de texte

L'ajout de contenu réaliste renforce notablement la crédibilité de la vie simulée. Notamment, l'absence d'échanges de mails ou de documents sur les postes utilisateur est un indicateur fort pour un adversaire, y compris si le contenu des mails et des documents est manifestement non crédible (e.g., succession de mots vide de sens). Cependant, la génération manuelle de ce contenu est laborieuse. En effet, rédiger du texte à la manière d'un avatar est en soi une tâche fastidieuse qu'il est pourtant nécessaire de reproduire pour plusieurs dizaines d'avatars dans le cas de scénarios complexes.

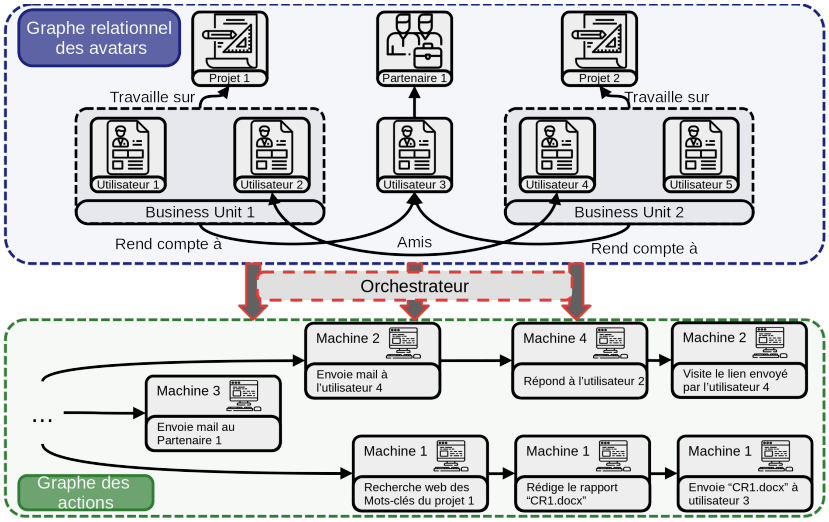

Fig. 3. Vue d'ensemble de l'orchestrateur

Le domaine du traitement du langage naturel (NLP, *Natural Language Processing*) a récemment connu une avancée majeure avec la démocratisation des réseaux de neurones type Transformer [26]. Le mécanisme d'attention, principale particularité de ces modèles, leur permet de modéliser avec précision la syntaxe et la sémantique de plusieurs langages naturels. OpenAI a démontré l'efficacité de l'apprentissage par transfert avec des modèles comme GPT [18], ainsi que ses capacités dans le domaine de la génération de texte. Le modèle employé génère du texte dans la continuité d'un contexte fourni au préalable (souvent des mots commençant une phrase). Dans notre cas, nous affinons le modèle pré-entraîné GPT-2 [19] pour la génération conditionnelle de texte. Plus spécifiquement, le contexte fourni en entrée du modèle contient des informations sur le ton à employer dans le texte (e.g., professionnel, informel, à un partenaire, à un client) ainsi que les thématiques à aborder (e.g., le sujet du mail, des mots clés).

Pour vérifier l'efficacité de notre méthode de génération de texte, nous avons entraîné un modèle sur un jeu de donnée de 35 000 mails récupérés en source ouverte. Ces mails sont associés à un ensemble de mots-clés, une polarité (positif, négatif ou neutre) et une tonalité (formel, informel).

5 Discussion

5.1 Actions exécutées par l'agent

Afin de ne pas laisser de traces d'instrumentation sur la machine cible, nous avons développé un agent qui communique au travers d'interfaces externes (clavier, souris, écran), le rendant ainsi indétectable pour un attaquant ou un outil de supervision (e.g., EDR). L'agent développé est apte à effectuer des activités de

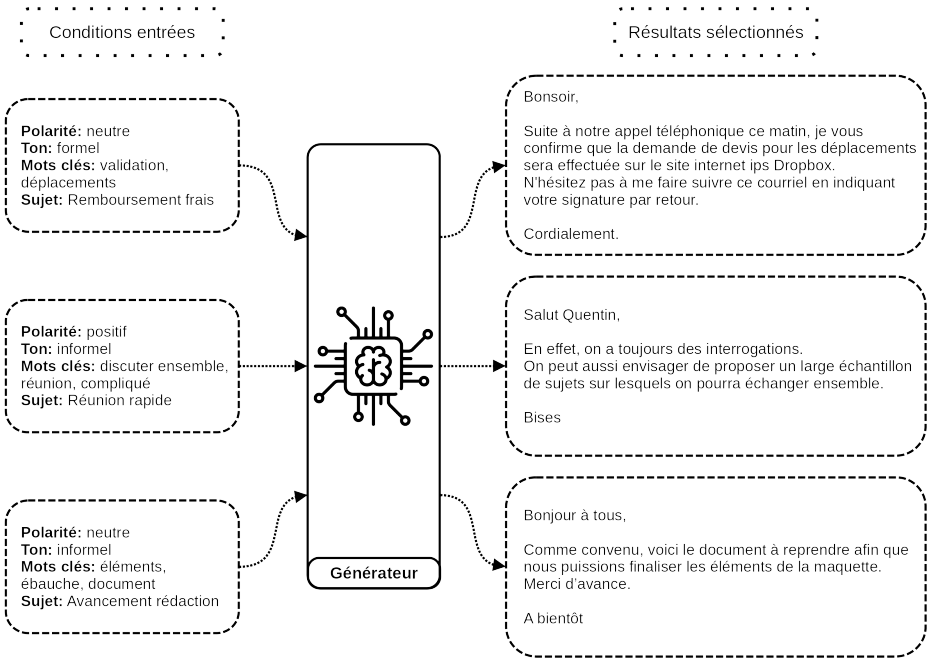

Fig. 4. Exemple de mails générés conditionnellement avec le modèle dérivé de GPT-2

bureautique simples (lecture/écriture de mail, navigation web, traitement de texte, etc.). L'automatisation est de plus masquée par l'ajout d'un aléa dans les mouvements de souris et les frappes clavier qui compliquent la détection automatique de cette instrumentation (ce que font certains logiciels malveillants pour détecter des sandbox [25]), obligeant ainsi un attaquant à évaluer la crédibilité du scénario en lui même.

Le panel des actions que l'agent peut réaliser améliore significativement la robustesse des systèmes actuels face à des malware utilisant des tests de Turing inversés. En revanche, la bibliothèque d'actions devra être complétée pour créer des jeux de données réalistes destinés à valider le fonctionnement d'algorithmes de détection. En particulier, la réalisation de tâches d'administration et plus largement des actions générant du bruit au niveau des outils de supervision (e.g., un administrateur ne respectant pas les procédures, un utilisateur utilisant des outils de scan réseau, etc.) permettraient de différencier les niveaux techniques des utilisateurs simulés et, par la même occasion, d'enrichir les jeux de données pour la détection.

L'enregistreur d'actions permet d'adapter en quelques minutes les actions existantes à des environnements nouveaux. Ceci permet d'une part d'intégrer rapidement, sans connaissances préalables sur le fonctionnement de notre outil, de nouvelles actions supportant des cas non gérés (e.g., une nouvelle pop-up pour la gestion des cookies lors de la navigation web, une mise à jour majeure d'un

logiciel, etc.) et d'autre part, de réaliser des actions complexes qui nécessiteraient un temps de développement important. L'adaptation de scénarios complets, bien que grandement facilitée, demeure toutefois une activité complexe.

5.2 Génération de scénarios

Ce travail présente la conception d'un agent seul et ne traite donc pas de l'orchestration des agents au sein d'un environnement multi-machines. Cette problématique comprend la communication et l'organisation entre les agents (e.g., répondre à mail) mais également la gestion de profils utilisateurs brièvement abordée dans la Section 4.1.

De plus, le modèle de génération de texte actuellement développé génère une majorité de résultats non crédibles : environ 20% sont exploitables avec des modifications mineures. Ceci nous empêche de générer la totalité des échanges sans intervention humaine. Bien que la génération ne soit pas entièrement automatique, celle-ci est toutefois grandement facilitée. En effet, il est possible, à partir du même contexte, de générer plusieurs candidats que l'opérateur peut utiliser pour gagner du temps dans la génération des mails et documents du scénario. Nous envisageons deux axes majeurs pour améliorer ces capacités de génération :

1. Augmenter la quantité de données et la taille du modèle (e.g., utiliser une autre variante de GPT-2);
2. Rajouter une étape d'entraînement type GAN (Generative Adversarial Network) [5], pour inciter le modèle à générer des candidats plus réalistes.

6 Conclusion

Nous avons présenté une méthode pour simuler des utilisateurs à l'échelle d'un SI complet en favorisant l'adaptabilité et la simplicité de mise en œuvre. Notre approche emploie un agent externe découpé en plusieurs niveaux d'abstraction, avec, au plus bas, une interaction avec les machines instrumentées au travers du clavier, de la souris et de l'écran. Pour faire face à la diversité des interfaces utilisateur des systèmes modernes (e.g., OS différents, logiciels différents, etc.) l'agent embarque des techniques de reconnaissance d'images reposant à la fois sur des méthodes déterministes (template matching) et d'apprentissage profond (réseaux de neurones à convolution). Ceci équilibre la quantité de ressources calculatoires requises par l'agent avec la flexibilité offerte par les méthodes probabilistes. Un enregistreur d'actions simplifie la configuration de l'agent pour tout nouvel environnement en extrayant automatiquement les images cibles, et en facilitant la collecte et l'annotation de données d'entraînement pour les modèles statistiques. La création de scénarios de vie se base sur les avatars des utilisateurs virtuels et de leurs interactions dont se nourrissent nos modèles de génération conditionnelle de texte qui produisent en masse des conversations e-mail réalistes et des documents crédibles.

Notre proposition enrichit la méthode de génération de vie initialement proposée pour la plateforme BEEZH par l'amélioration des performances et le

réalisme de l'activité générée. Nous complétons également la proposition initiale par une assistance pour la génération de scénarios à grande échelle. En cours d'implémentation, nos travaux bénéficient de premiers résultats encourageant qui restent toutefois à consolider avant de valider expérimentalement notre proposition.

Acknowledgments Les auteurs remercient chaleureusement Alexandre De Beaudrap dont les travaux de stage ont permis de consolider l'approche.

References

1. Afianian, A., Niksefat, S., Sadeghiyan, B., Baptiste, D.: Malware Dynamic Analysis Evasion Techniques: A Survey. arXiv:1811.01190 [cs] (Nov 2018), http://arxiv.org/abs/1811.01190, arXiv: 1811.01190
2. Amossys: desker. https://gitlab.com/d3sker/desker (2021)
3. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
4. Bulazel, A., Yener, B.: A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion: PC, Mobile, and Web. In: Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium. pp. 1–21. ROOTS, Association for Computing Machinery, Vienna, Austria (nov 2017). https://doi.org/10.1145/3150376.3150378, https://doi.org/10.1145/3150376.3150378
5. Croce, D., Castellucci, G., Basili, R.: GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 2114–2119. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.191, https://aclanthology.org/2020.acl-main.191
6. Feng, P., Sun, J., Liu, S., Sun, K.: UBER: Combating Sandbox Evasion via User Behavior Emulators. In: Zhou, J., Luo, X., Shen, Q., Xu, Z. (eds.) Information and Communications Security, vol. 11999, pp. 34–50. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-41579-2_3
7. Guihéry, F., Siffer, A., Paillard, J.: Beezh: une plateforme de détonation réaliste pour l'analyse des modes opératoires d'attaquants (2020)
8. Jelinek, F.: Markov source modeling of text generation. In: The impact of processing techniques on communications, pp. 569–591. Springer (1985)
9. Jiang, H., Choi, T., Ko, R.K.: Pandora: A cyber range environment for the safe testing and deployment of autonomous cyber attack tools. arXiv preprint arXiv:2009.11484 (2020)
10. Keskar, N.S., McCann, B., Varshney, L.R., Xiong, C., Socher, R.: CTRL: A conditional transformer language model for controllable generation (2019)
11. Lie, W.N.: Automatic target segmentation by locally adaptive image thresholding. IEEE Transactions on Image Processing **4**(7), 1036–1041 (1995)
12. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection (2018)

13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. Lecture Notes in Computer Science pp. 21–37 (2016)
14. Mills, A., Legg, P.: Investigating Anti-Evasion Malware Triggers Using Automated Sandbox Reconfiguration Techniques. Journal of Cybersecurity and Privacy **1**(1), 19–39 (Mar 2021). https://doi.org/10.3390/jcp1010003, https://www.mdpi.com/2624-800X/1/1/3
15. Miramirkhani, N., Appini, M.P., Nikiforakis, N., Polychronakis, M.: Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 1009–1024. IEEE, San Jose, CA, USA (May 2017). https://doi.org/10.1109/SP.2017.42, http://ieeexplore.ieee.org/document/7958622/
16. Nguyen, B.N., Robbins, B., Banerjee, I., Memon, A.: Guitar: an innovative tool for automated testing of gui-driven software. Automated software engineering **21**(1), 65–105 (2014)
17. tesseract ocr: tesseract. https://github.com/tesseract-ocr/tesseract (2019)
18. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
19. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
20. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019)
21. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection (2016)
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks (2016)
23. Rueda, U., Vos, T.E., Almenar, F., Martínez, M., Esparcia-Alcázar, A.I.: Testar: from academic prototype towards an industry-ready tool for automated testing at the user interface level. Actas de las XX Jornadas de Ingenierıa del Software y Bases de Datos (JISBD 2015) pp. 236–245 (2015)
24. Singhera, Z., Horowitz, E., Shah, A.: A graphical user interface (gui) testing methodology. International Journal of Information Technology and Web Engineering (IJITWE) **3**(2), 1–18 (2008)
25. Vashisht, S.O., Singh, A.: Turing Test in Reverse: New Sandbox-Evasion Techniques Seek Human Interaction (2014), https://www.fireeye.com/blog/threat-research/2014/06/turing-test-in-reverse-new-sandbox-evasion-techniques-seek-human-interaction.html
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
27. Yeh, T., Chang, T.H., Miller, R.C.: Sikuli: using gui screenshots for search and automation. In: Proceedings of the 22nd annual ACM symposium on User interface software and technology. pp. 183–192 (2009)