

Prise d'empreinte de système d'exploitation via analyse passive du trafic réseau

1) Partenaires

- ANSSI : Johan Mazel, Pierre Chifflier, Clément Parssegny;
- Inria : Michel Hurfin, Equipe PIRAT¹);
- IMT Atlantique Rennes : Gilles Guette, Equipe Sotern;

2) Contexte

La prise d'empreinte passive de systèmes d'exploitation via l'analyse du trafic réseau est un composant important de la supervision d'un réseau car elle permet l'identification des systèmes d'exploitation des hôtes présents sur ce réseau. Ceci améliore la connaissance qu'une organisation a de son réseau et est particulièrement utile lors des missions d'audit ou de réponses à incidents. L'identification des systèmes d'exploitation présent sur un système d'information supervisé est aussi une condition nécessaire pour la configuration de certains outils de supervision (ex Suricata [1], [2]). Si l'empreinte est suffisamment précise, elle permet aussi d'identifier les équipements vulnérables. Ces deux informations permettent ainsi à une organisation de mettre en place des actions qui améliorent sa posture de sécurité. Ce document identifie les systèmes d'exploitation comme la cible de la prise d'empreinte. Le composant logiciel qui est en fait identifié est la pile réseau TCP/IP¹. Cette subtilité n'est pas rappelée à chaque fois que le terme "prise d'empreinte de systèmes d'exploitation" est mentionnée mais elle doit rester à l'esprit du lecteur.

Nous identifions deux principaux challenges pour l'amélioration des techniques de prise d'empreinte :

- l'utilisation de données suffisamment diverses et la génération reproductible de ces données ;
- améliorer la granularité de la classification en utilisant des labels hiérarchique qui permettent une sémantique de classification plus riche (ex. identifier un système d'exploitation vs identifier une version spécifique qui est vulnérable).

Voici des explications un peu plus détaillées sur ces deux aspects.

La première limitation est liée aux données utilisées dans les travaux existants. Ces données sont en effet peu diverses. Elles ne contiennent bien souvent que des systèmes d'exploitation de postes de travail ou serveurs (ex Windows, Linux, BSD, OSX), et de terminaux mobiles (ex. Android et iOS). Le nombre des systèmes d'exploitation est également limité, 4 systèmes d'exploitation dans [3], 13 dans [4] qui utilise CIC IDS 2017 [5], ou 7 pour [6]. Il nous semble très important de construire des données qui intègrent d'autres types d'équipements et leurs systèmes d'exploitation tels que les systèmes embarqués (ex piles lwip[7] ou uIP[8]) et les systèmes industriels (ex automates). De plus, ces jeux de données ne sont bien souvent pas disponibles [3] ou d'un intérêt limité dans la mesure où les attributs ont déjà été pré-extraits et les données brutes ne sont pas disponibles [6]. Une approche possible pour obtenir la diversité souhaitée est de générer des données annotées. Cette génération doit s'appuyer sur des environnements contrôlés (par exemple via virtualisation) et reproductibles. Cela demande cependant un effort considérable de mise en place et de configuration. La seule approche existante de ce type [9] déploie cependant des systèmes d'exploitation qui sont au repos (i.e. ni client ni serveur lors d'un échange protocolaire). Il convient donc d'adapter ou étendre cette approche afin de collecter des données dans lesquelles les machines testées se comportent comme des clients *et* des serveurs.

¹Il existe aussi des piles réseaux indépendantes (ex lwip ou uIP) qui sont déployées sur des systèmes d'exploitation distincts.

Enfin, la prise d’empreinte de systèmes d’exploitation est une tâche dont le résultat peut avoir différente granularité (ex. famille de système d’exploitation, ou famille de système d’exploitation et version/sous-version). Ce résultat est en fait une hiérarchie qui peut se représenter via un arbre sur lequel les familles d’OS sont le noeud le plus proche de la racine, et les versions/sous-versions sont situées sur les noeuds qui s’éloignent de cette racine. Or, les métriques de performance utilisées dans les travaux existants sont celles des problèmes de classification ce qui n’est pas adapté. Par exemple, si l’on souhaite identifier une version, le label contient la famille et la version du système d’exploitation, une erreur de version avec une famille correcte sera considérée comme fausse alors que le résultat est seulement partiellement erroné. Il nous semble donc pertinent d’utiliser des métriques de performance de classification hiérarchique [10], [11] pour améliorer l’évaluation des méthodes de prise d’empreinte.

3) Objectifs

3.a) Construire un jeu de données de référence pour la prise d’empreinte de système d’exploitation

Pour pouvoir évaluer les performances des méthodes existantes de prise d’empreinte, un jeu de données contenant le trafic émis par les systèmes d’exploitation précédemment étudiées sera constitué. Pour toutes les configurations précédemment identifiées, le trafic réseau de l’implémentation en mode client et serveur sera capturé. Ce travail s’appuiera sur des déploiements en environnement virtualisé (Vagrant et, qemu/libvirt ou Virtualbox).

3.b) Concevoir une approche de prise d’empreinte qui limite l’impact des altérations de configuration par défaut

Les objectifs précédents auront permis 1) de connaître la façon sur laquelle les systèmes d’exploitation sont configurables et configurés et 2) d’obtenir du trafic réseau généré en environnement contrôlé qui illustre la diversité des systèmes d’exploitation et de leurs configurations possibles.

En partant de ces données, une nouvelle méthode de prise d’empreinte sera conçue. Elle devra être capable d’identifier les système d’exploitation de manière fiable dans le cas où ceux-ci sont déployés avec une configuration par défaut et avec des paramètres altérés. Cette méthode devra ainsi être capable de trouver des invariants protocolaires qui ne sont pas affectés par l’utilisation d’une configuration spécifique. Dans le cas où ces invariants n’existeraient pas, cette méthode devra également être capable de construire des scores de vraisemblance qui permettent de maintenir de bonnes performances. Il est envisagé d’utiliser ici des techniques par signature [12], par IA [4], ou une hybridation des deux. Il est en particulier envisagé de s’appuyer sur l’analyse des déploiements effectuée précédemment pour orienter la décision de la méthode proposée.

4) Résultats attendus & perspectives

Résumant les principaux objectifs du stage, les résultats attendus des travaux de recherche sont décrits ci-dessous.

- Construction d’un jeu de données annotés qui capture cette diversité.
- Conception d’une méthode de prise d’empreinte qui résiste aux altérations de configuration ou de *middlebox*.

5) Compétences requises

Le candidat, présentant des compétences solides en informatique, doit maîtriser au moins un langage de programmation (idéalement C ou Python ou Rust) et avoir des connaissances raisonnables sur le fonctionnement des réseaux TCP/IP.

Une spécialisation en sécurité ou en réseaux informatiques serait appréciée.

Des compétences en apprentissage automatique/IA ou virtualisation (Vagrant/VirtualBox) seraient un plus.

6) Logistique

Le stage se déroulera à l'IMT Atlantique campus de Rennes.

- Durée du stage : 5/6 mois.
- Rémunération : gratification de stage.
- Niveau : M2, troisième année d'école d'ingénieurs ou équivalent.
- Candidature par mail à gilles.guette@imt-atlantique.fr avec CV et lettre de motivation.

Bibliography

- [1] OISF, "Suricata." [Online]. Available: <https://suricata-ids.org/>
- [2] OISF, "Suricata Documentation on Host-os-policy." [Online]. Available: <https://suricata.readthedocs.io/en/suricata-4.0.2/configuration/suricata-yaml.html#host-os-policy>
- [3] B. Anderson and D. McGrew, "OS fingerprinting: New techniques and a study of information gain and obfuscation," in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017, pp. 1–9.
- [4] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3366–3383.
- [5] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, and others, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, no. 2018, pp. 108–116, 2018.
- [6] M. Laštovička, M. Husák, P. Velan, T. Jirsík, and P. Čeleda, "Passive operating system fingerprinting revisited: Evaluation and current challenges," *Computer Networks*, vol. 229, p. 109782, 2023.
- [7] L. W. Adam Dunkels, "lwip." [Online]. Available: https://www.nongnu.org/lwip/2_1_x/index.html
- [8] A. Dunkels, "uip." [Online]. Available: <https://github.com/adamdunkels/uip>
- [9] V. Bartoš, M. Novotná, and M. Hulák, "CESNET Idle OS Traffic: A Dataset of Idle Traffic of Various Operating Systems," in *2025 21st International Conference on Network and Service Management (CNSM)*, 2025, pp. 1–6.
- [10] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.
- [11] F. M. Miranda, N. Köhnecke, and B. Y. Renard, "Hiclass: a python library for local hierarchical classification compatible with scikit-learn," *Journal of Machine Learning Research*, vol. 24, no. 29, pp. 1–17, 2023.
- [12] M. Zalewski, "p0f." [Online]. Available: <https://lcamtuf.coredump.cx/p0f3/>