Analyse de Trafic Chiffré : Application au Web

Pierre-Olivier Brissaud*†, Jérôme François*, Isabelle Chrisment*‡, Thibault Cholez*‡ et Olivier Bettan†
*Inria Nancy Grand Est, France, prénom.nom@inria.fr
†Thales Service, Palaiseau, France, prénom.nom@thalesgroup.com
‡TELECOM Nancy, Université de Lorraine, France

***Abstract*—Le chiffrement des communications est devenu un enjeu majeur pour tous les fournisseurs de services web. En effet, suite à de nombreux scandales, la population a pris conscience de l'importance de protéger ses données et regarde comment elles transitent et sont utilisées. En revanche, face à l'augmentation du chiffrement des communications, la supervision du trafic, étant encore nécessaire, doit maintenant s'adapter et évoluer pour rester efficace. Dans un environnement d'entreprise par exemple, il est légitime de superviser et caractériser les communications afin de détecter, ou même bloquer le cas échéant, les contenus inappropriés ou illégaux.**

L'utilisation d'un proxy de déchiffrement ou la mise en place de blocage de services ne correspondent pas à nos besoins. En effet, le premier ne respecte pas la vie privée des utilisateurs et le deuxième est devenu obsolète avec le développement de quelques très grands acteurs qui centralisent toutes sortes de services difficilement différentiables.

Nous proposons ici une solution passive et transparente pour l'utilisateur, qui se base sur la taille des objets des pages chargées. Cela permet de mettre en place un filtrage plus fin pour certains services et d'en différencier les différents usages.

I. Introduction

Aujourd'hui, de plus en plus de services sont rendus accessibles via le web et demandent aux utilisateurs de renseigner des données personnelles. En même temps, de nombreux scandales publics sur la surveillance de masse, ont permis une prise de conscience des utilisateurs et leurs attentes face à la sécurisation de leurs communications ont augmentées. Ainsi, beaucoup de services utilisent du chiffrement par défaut pour protéger les données de leurs utilisateurs. Cela se traduit par le développement important du protocole HTTPS [1]. D'après l'Electronic Frontier Foundation [1] depuis février 2017, plus de la moitié du trafic internet était protégée par HTTPS.

Bien que la défense de la vie privée des utilisateurs soit une priorité, la surveillance et la détection de comportements inappropriés ou interdits est également légitime, du point de vue d'une entreprise par exemple. Les techniques basiques consistent à effectuer du filtrage via les ports ou à l'aide d'inspecteurs de paquets mais sont totalement inefficaces face au chiffrement. L'usage de proxy de déchiffrement n'étant évidement pas une solution envisageable si on veut assurer un minimum de confidentialité pour les utilisateurs, il faut trouver une autre méthode.

Comme autre solution, nous nous sommes tournés vers l'analyse de trafic chiffré. Cette technique passive et non intrusive pour les clients tente d'extraire de l'information sans effectuer de déchiffrement. On trouve différentes applications à cette technique : on notera par exemple la reconnaissance de pages web malgré l'utilisation de VPN ou de réseaux d'anonymisation [2] [3] [4] [5], la reconnaissance de mots ou d'expressions sur la VoIP [6] [7] ou sur des vidéos telles que le propose Netflix [8]. À un autre niveau, nous pouvons identifier le type de trafic d'après les travaux de Velan [9] ou à un niveau plus haut, reconnaître un service [10].

Dans le cadre d'une entreprise, la surveillance d'un service HTTPS a besoin de ce type de solutions car il peut actuellement soit tout accepter sans contrôle, soit tout bloquer, bien que la majorité de son usage soit légitime. C'est dans ce cadre que notre contribution s'inscrit. La solution que nous présentons ici permet de détecter une liste prédéfinie de mots-clefs (blackliste établie par l'administrateur) lors d'une requête sur un moteur de recherche d'images en écoutant uniquement du flux réseau.

Nous expliquerons tout d'abord la méthodologie que nous avons suivie puis nous présenterons les résultats obtenus avec notre solution appliquée au moteur Google Images.

II. Méthodologie

A. Vue d'ensemble

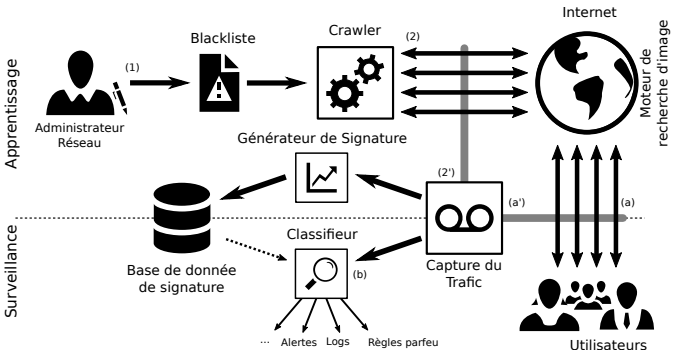

Fig. 1: Architecture globale pour la détection dans un service HTTPS

Comme indiqué dans la Figure 1, notre solution se décompose en deux parties : l'apprentissage et l'analyse du trafic. La première permet la composition d'une base de connaissances qui va créer et enregistrer une signature pour chaque mot-clef de la blackliste définie par l'administrateur (1). Pour ce faire, l'outil de capture va enregistrer les traces réseaux (2') du chargement des pages correspondantes aux

[1] https://www.eff.org/deeplinks/2017/02/were-halfway-encrypting-entire-web

requêtes des mots-clefs sur le moteur de recherche d'images (2) et construire une signature. Ensuite, lorsqu'un utilisateur va chercher un mot-clef sur le moteur d'images, notre système va capturer le trafic, l'analyser, en déduire une signature et la comparer à notre base de connaissances. Le moteur de classification prendra alors une décision à partir des scores obtenus. Il pourra ainsi détecter s'il s'agit d'un mot-clef de la base de données et de savoir lequel ou de ne rien faire avec des mot-clefs légitimes.

B. Génération des signatures

Lors du chargement d'une page sur un moteur de recherche d'images, le navigateur récupère un nombre conséquent de miniatures d'images qui constituent une grande partie du contenu de la page. Chaque image ayant une taille différente dans un intervalle presque uniforme de 6000 valeurs et le nombre de miniatures étant de 50 en moyenne (informations déterminées empiriquement sur notre dataset), cela suffit à caractériser un mot clef de manière efficace comme nous allons le voir. Les signatures reflètent ainsi les tailles des images contenues dans la page correspondant à la recherche d'un mot-clef sur un moteur d'images lors des différentes captures.

Pour construire notre signature, nous utilisons KDE (Kernel Density Estimation [11]). Cela permet d'approximer via la fonction de densité, les tailles d'images pour l'ensemble des captures d'un mot-clef et de lisser les petites variations sur la taille des objets. Ces variations pouvant être dues aux variations sur les métadonnées, elles sont contenues dans un intervalle de plus ou moins 16 octets (déterminé empiriquement). À la fin de cette étape, nous avons une base de données qui lie chaque mot-clef de la blackliste à une signature qui se présente sous la forme d'une fonction.

C. Algorithme de classification

L'objectif du moteur de classification est de déterminer si une trace correspondant au chargement d'une page sur un moteur d'images correspond à un mot-clef interdit. Pour ce faire, la classification se passe en trois étapes :

- Extraction des tailles des miniatures
- Calcul du score pour chaque signature
- Filtrage du résultat

À partir des traces réseau, nous extrayons les tailles des objets HTTP qui sont envoyés puis nous gardons uniquement les tailles correspondantes à des images. Pour réaliser ce tri, nous nous basons sur la taille des requêtes envoyées par le navigateur. En effet, la requête pour une image est stable et facilement différentiable car comprise dans une plage de valeurs caractéristiques. Ainsi, en repérant les requêtes pour les images, il est possible de filtrer les tailles d'objets et de garder uniquement celles des images.

Pour déterminer si une nouvelle trace correspond à un des mots-clefs blacklistés, l'algorithme de classification évalue la fonction signature de chaque mot-clef en chacun des points correspondant à la taille d'une image contenue dans la trace à tester. Ensuite, ces valeurs sont sommées pour définir un score global. En effet, si une trace correspond à un mot-clef, l'évaluation de sa signature aux points correspondant à une taille d'image devrait donner des valeurs non nulles et le score total sera plus grand si une trace correspond au mot-clef car les tailles des images extraites seront similaires.

Enfin, un filtrage du score est réalisé avant de donner un résultat. Ce filtrage permet de décider si la trace considérée correspond ou non à un mot-clef de la blackliste. Cela permet ainsi d'éliminer une trace légitime. Pour ce faire, la signature ayant obtenu le score le plus élevé est comparée aux autres suivant l'équation 1, avec s la trace à classifier, k le mot-clef correspondant au meilleur score et L_s la liste de tous les scores pour la trace s. Si le $score_k(s)$ vérifie cette inégalité, k sera le mot-clef retenu, sinon il retournera l, ce qui correspond à une requête légitime, l'idée étant de détecter les score largement supérieurs aux autres, qui correspondraient ainsi à des mots-clefs connus.

$$score_k(s) > moyenne(L_s) + \alpha \times std(L_s) \tag{1}$$

III. Evaluation

Notre évaluation se décomposera en deux parties, la première définira les métriques utilisées et présentera rapidement le dataset et la deuxième donnera les résultats de notre expérimentation.

A. Métriques et jeu de données

Les métriques utilisées pour évaluer notre jeu de données sont les suivantes :

- **TPR** (True Positive Rate) La probabilité qu'une trace d'un mot-clef blacklisté soit classée avec le bon mot-clef blacklisté.
- **FPR** (False Positive Rate) La probabilité qu'une trace légitime soit classée avec un mot-clef blacklisté à tord.
- **WCR** (Wrong Classification Rate) La probabilité qu'une trace d'un mot-clef blacklisté soit classée comme un autre mot-clef blacklisté.
- **Acc** (Accuracy) Le ratio de traces bien classées indépendamment de leur nature.

Le TPR et le FPR nous permettent de tracer une courbe ROC (Receiver Operating Characterictic). Plus la courbe est proche du point (0,1), plus la précision est bonne.

Les traces utilisées pour évaluer notre approche ont été collectées par notre outil de capture pour des recherches de mots-clefs sur *Google Images* entre juin et juillet 2017. Les conditions de capture étaient les suivantes :

- Côté client : Firefox 54.0.1 (64-bit) sur debian 8,
- Paramètres Firefox : plein écran avec résolution de 1920×1080 sans HTTP/2 et sans cache,
- Version anglaise de *Google Images* (google.com [2]).

[2]Pour ne pas être redirigé suivant la localisation d'une adresse IP, il faut aller à l'adresse suivant : https://google.com/ncr

TABLE I: Résultats : 10,500 blacklistés / 105,000 légitimes

α \ h	1	4	16	50	1	4	16	50	1	4	16	50	1	4	16	50
	TPR				FPR				Accuracy				WCR			
3.5	0.820	0.979	0.986	0.927	0.1042	0.0908	0.0811	0.0739	0.8889	0.9207	0.9250	0.9262	0.017	0.001	0.001	0.001
3.9	0.814	0.973	0.980	0.790	0.0219	0.0201	0.0162	0.013	0.9631	0.9788	0.9834	0.9691	0.004	0	0	0
4.1	0.805	0.965	**0.966**	0.557	0.0052	0.0069	**0.0056**	0.0026	0.9775	0.9884	**0.9918**	0.9574	0.001	0	**0**	0
4.25	0.780	0.932	0.871	0.170	0.0009	0.0038	0.0026	0.0003	0.9792	0.9857	0.9858	0.9242	0			
4.35	0.448	0.253	0.009	0	0	0.0004	0	0	0.9498	0.8762	0.9099	0.9091	0			

Notre jeu de données est séparé en deux avec $data_1$ qui correspond à des captures collectées plusieurs fois pour l'entraînement et $data_2$ qui rassemble plus de mots-clefs différents mais en un seul exemplaire pour simuler des traces légitimes. Le tableau II donne une vue schématique de ces jeux de données. Il n'y a pas de collision de mots-clefs entre les deux datasets. Les données sont disponibles publiquement sur http://betternet.lhs.inria.fr/datasets/googleimg/ pour supporter la recherche reproductible.

TABLE II: Origine du Dataset

ID	Origine	Nombre de mots-clefs	Captures par mot-clef	Nombre moyen de paquets par trace	Durée moyenne par trace
$data_1$	Dictionnaire (en)	10,500	5	1,155 pkt/trace	2.94 s/trace
$data_2$	Titre de page Wikipedia (en)	105,000	1		

B. Résultats

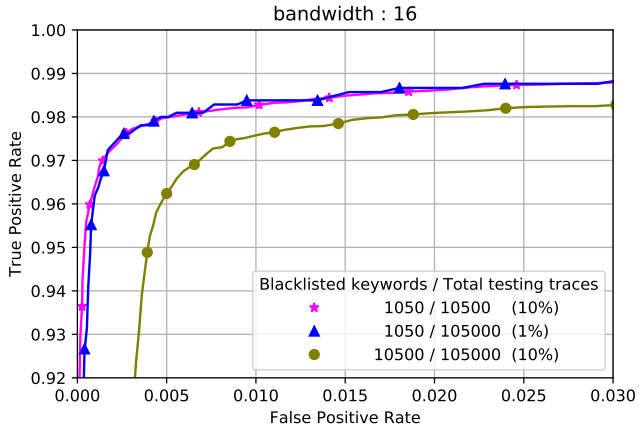

Fig. 2: Courbe ROC : en fonction du nombre d'éléments de la blacklist ou le nombre de requêtes légitimes.

La première partie des résultats que nous avons obtenus nous a servi à fixer les paramètres en nous basant uniquement sur les traces de $data_1$. Nous ne détaillerons pas cette partie ici mais cela nous a permis de fixer les paramètres α (paramètre de filtrage) et h (largeur de bande pour KDE).

Après cela, en utilisant $data_1$ (4 traces d'entraînement et une trace de test par mot-clef) et $data_2$ (traces légitimes pour le test) pour évaluer notre classifieur, nous avons obtenu les résultats consignés dans le tableau I. Dans le meilleur cas, nous constatons une précision de 0.9918 avec $TPR = 0.966$ et $FPR = 0.0056$.

La figure 2 nous permet d'évaluer l'impact du nombre de signatures et de requêtes légitimes sur les résultats de la classification. On constate que le nombre de requêtes légitimes ne modifie pas les taux pour le TPR et le FPR. En revanche une augmentation du nombre de mots blacklistés diminue les performances.

IV. Conclusion

Ce papier introduit un framework de filtrage HTTPS destiné aux moteurs de recherche d'images. Cette solution a été conçue pour être utilisée dans un réseau d'entreprise pour surveiller et filtrer une liste de mots-clefs. La méthode repose sur l'observation des tailles individuelles des images chargées lors d'une requête. En utilisant KDE pour former une signature, nous avons montré l'efficacité de cette solution sur Google Images. L'évaluation sur 115 000 traces donne une précision de plus de 99% et moins de 1% de faux positifs. Notre solution permet une surveillance des utilisateurs respectueuse de la vie privée car elle ne détecte que les comportements non légitimes. Les prochaines perspectives sont de faire fonctionner notre méthode sur du trafic HTTP2 et de pouvoir dériver les signatures pour d'autres configurations automatiquement.

References

[1] E. Rescorla, "Http over tls," RFC 2818, 2000.

[2] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 103–114.

[3] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 605–616.

[4] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting." in *USENIX Security Symposium*, 2014, pp. 143–157.

[5] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique." in *USENIX Security Symposium*, 2016, pp. 1187–1203.

[6] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *Security and Privacy*. IEEE, 2008.

[7] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob?" in *USENIX Security Symposium*, vol. 3, 2007, pp. 43–54.

[8] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. ACM, 2017, pp. 361–368.

[9] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[10] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify https services," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2016.

[11] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.