Automatisation Metasploit

Auteur : Mathis GOGAN

Maître de stage : Paul MALÉCOT

Tuteur pédagogique : Morgan ROGERS

2023-2024

Remerciement

Premièrement je tiens à remercier ma mère, qui m'a mis en contact avec le frère d'un de ses collègue Marc DELAMARRE. Ce qui m'a permis de trouver mon stage. Je remercie mes maitres de stage Paul MALÉCOT et Amine LARABI, ainsi qu'à l'équipe qui m'a accompagné, Romain FRANCOIS, Nicolas KORN. Je remercie également toute la R&D, pour l'aide qu'ils m'ont apporté.

Abstract

The project aimed at automating Metasploit to generate packets containing an attack, and to test the detection quality of the TD SDK IDS and IxEngine DPI.

My internship at ENEA lasted eleven weeks. During this period, I developed a python script that automates network capturing (pcap) of attacks using Metasploit. The result of this script is a folder containing the pcap files and a text file describing the attack and the target.

After the pcap which were generated, I tested the quality of the detection of the TD SDK and the IxEngine. Attack detection is better on OS Windows than on OS Linux because the rules target more Windows.

Metasploit automation was a success, with the script generating almost 1 GB of network traffic. The TD SDK and IDS IxEngine detection quality test is quite good.

Table des matières

Introduction

Passionné dans le domaine de la cybersécurité, j'ai eu l'opportunité d'effectuer un stage de onze semaines chez ENEA, une entreprise internationale, spécialisée dans les réseaux et les télécoms. Au cours de ce stage, j'ai occupé le poste de stagiaire au sein du département Recherche et Développement (R&D). Ma mission principale consistait à développer un projet d'automatisation de Metasploit.

Ce stage a été pour moi une expérience enrichissante qui m'a permis d'approfondir mes connaissances dans le trafic réseau ainsi que la programmation. Dans ce rapport, je présenterai mon expérience au sein de ENEA. J'aborderai d'abord la présentation de l'entreprise, je décrirai ma mission de stage, avec comme problématique le manque de captures réseau (pcap) contenant des attaques. Ensuite je présenterai le déroulé du stage et les premiers obstacles rencontrés. Enfin je présenterai les résultats obtenus à la fin du stage, et les compétences que j'ai acquis grâce au stage.

1. Déroulé du stage

Dans cette première partie nous allons introduire l'entreprise ainsi que la problématique.

1.1. ENEA

ENEA [1] est fondée en 1968 par Rune Engman, le premier produit développé par l'entreprise est une solution de stockage de données dans un système d'exploitation pour le contrôle aérien pour l'armée de l'air Suédoise.

En 1985 ENEA OSE est lancé, l'OSE est un système d'exploitation temps réel pour les applications critiques.

Dans les années 2000, ENEA consolide sa place de leader mondial de solutions pour les produits à forte intensité de communication. Une acquisition en Roumanie augmente la capacité d'externalisation de l'entreprise.

En 2010, ENEA cède son organisation de service suédoise, ce qui permet à l'entreprise de se concentrer sur les télécommunications et les réseaux. L'entreprise investie dans des projets de collaborations open source (Linux Fondation, Linaro, Yocto Project et OPNFV) et acquière Centered Logic, Openwave Mobility.

L'ixEngine a été créé par Qosmos dans les années 2000. IxEngine, est un moteur DPI* de classification et d'inspection des paquets, il est capable de reconnaitre plus de 4500 protocoles et applications. Qosmos a été racheté par ENEA en 2016.

Aujourd'hui ENEA développe un SDK* ENEA Qosmos Threat Detection (TD SDK), qui a pour but d'aider les systèmes de détections d'intrusions (IDS), les IDS* sont des éléments importants de sécurité des réseaux. Le TD SDK d'ENEA Qosmos se base sur l'IDS le plus connu du marché Suricata et est relié au moteur d'analyse du trafic ENEA Qosmos ixEngine.

1.2. Problématique

Depuis quelques années, ENEA s'oriente de plus en plus vers la cybersécurité, notamment avec leur produit TD SDK. Le problème rencontré par l'entreprise a été le manque de pcap comportant des traces d'attaques. Les traces d'attaques dans les pcap*, permettent de tester les règles de détections d'attaques ou de tentatives d'attaques. Lorsque qu'un pcap possédant une trace d'attaque est traité par le TD SDK, une anomalie doit être remontée. Si nous traitons un pcap contenant une attaque et qu'aucune anomalie est remontée, alors nous devons comprendre comment fonctionne l'attaque avant de créer une nouvelle règle, afin qu'une anomalie soit remontée lorsqu'un pcap comprenant des traces de cette attaque est analysé.

Le projet qui nous est demandé, c'est de créer une automatisation de Metasploit afin d'obtenir des pcap pour tester le niveau de sécurité du TD SDK et l'améliorer.

1.1. État de l'art sur l'automatisation

La première tâche à effectuer est de savoir si un projet d'automatisation de Metasploit a déjà été réalisé, ou s'il est en cours de réalisation. Afin de savoir si l'automatisation est possible, il a fallu rechercher qu'elles sont les méthodes qui ont été testées, celles qui ont fonctionnées et celles qui n'ont pas fonctionnées. A la suite de nos recherches nous avons trouvé plusieurs projets qui font de l'automatisation, mais ces automatisations ne visent qu'un seul type d'attaque. Tel que « Automating reconnaissance and brute force attacks [2] » réalisé par r00t-3xp10it, un pentester, ou « Ruby script to automate Metasploit scanning, exploitation, and post-exploitation » [3] réalisé par paulosgf, un analyste sécurité linux. Pour conclure sur nos recherches, nous avons appris qu'il existe des projets qui font de l'automatisation de Metasploit, mais qu'ils n'exploitent pas tous les exploits comme nous le souhaitons. Nous avons aussi pris connaissance des différents types de langages utilisés pour les scripts d'automatisation (Ruby [4], Python [5]).

1.2. Recherche des logiciels à utiliser

Pour réaliser ce projet, un réseau dédié aux attaques a été créé, nommé « SecurityLab», Dans ce réseau on va retrouver la VM (Virtual Machine) attaquante et les VMs cibles. L'outils proposé pour faire les attaques est Metasploit [6]. Metasploit est l'outil contenant le plus d'attaques sur Internet. Pour utiliser Metasploit nous avons décidé d'utiliser une VM Kali [7] qui est une VM spécialisée dans le pentest. Pour les machines cibles nous avons plusieurs idées, Metasploitable2 [8] [9] qui est une VM qui possède des failles Linux intentionnelles. Cette VM a été créée pour s'entrainer au pentest et à la sécurisation. Pour avoir une diversité d'attaque, nous ajoutons au réseau une VM Windows Active Directory [10], une VM Windows10 [11] et une VM Ubuntu 16-04 [12].

2. Dispositif expérimental

Dans cette partie nous allons premièrement montrer comment s'est déroulé le stage. Puis dans un second temps nous allons voir comment on a réussi à automatiser Metasploit. Et pour finir nous verrons comment avoir une grande surface d'attaque.

2.1. Calendrier

Le stage est réparti sur 11 semaines, la première semaine est consacrée à la prise en main de Metasploit, de la VM Kali, pour cela nous effectuons des attaques à la main sur les différentes cibles (Debian et Windows Active Directory), cette semaine a aussi servis à la découverte de l'entreprise. La deuxième semaine a été le début des recherches sur comment automatiser les attaques. La troisième semaine a été la réalisation d'un premier script qui se connecte à Metasploit, avec une revue des bases de python. La quatrième semaine a été la création d'un premier script d'automatisation. La cinquième semaine a été l'ajout de tcpdump dans le script pour pouvoir capturer les paquets lors des attaques et générer des pcap avec des traces d'attaques. La sixième semaine a été l'analyse des premiers pcap, à ce stade du projet les exploits qui étaient lancé par le script étaient prédéfini dans le script avant le lancement de celui-ci. Le script ne pouvait pas avoir plus d'un exploit en même temps. La septième et huitième semaine nous a servis à améliorer le script, pour qu'il soit capable d'exécuter plusieurs exploits en même temps. A ce stade du projet le script prend en argument le ou les exploits à lancer, il est même capable de tous les exécuter. La neuvième semaine nous a servis à augmenter la surface d'attaque en créant un script python qui créé et lance des VMs, nous avons également créé un script python qui ouvre et reste en écoute sur tous les ports qui sont fermés et non utilisés. La dixième et onzième semaine nous a servis à la réalisation du compte rendu, du PowerPoint et à l'analyse complète des résultats obtenus.

2.2. Framework d'attaque

Il existe plusieurs façons d'automatiser Metasploit, la première méthode consiste à utiliser Metasploit Pro qui est la version payante et qui permet d'obtenir de nombreux avantages par rapport à la version gratuite tel que le montre le tableau ci-dessous réalisé par « Red Team Security Blog [13]».

Fonctionnalité	Metasploit Pro	Métasploit gratuit
Base de données d'exploits intégrée	Oui	Oui
Interface utilisateur Web	Oui	Non
Rapports	Oui	Non
Planification	Oui	Non
Automatisation	Oui	Non
Récolte des informations d'identification	Oui	Oui
Multi-numérisation	Oui	Non
Analyse de vulnérabilité	Oui	Non
Tests de pénétration	Oui	Oui
Audit de mot de passe	Oui	Oui
Ingénierie sociale	Oui	Oui
Ingénierie inverse	Oui	Oui

Tableau 1 Comparaison de Metasploit et Metasploit Pro

On constate que Metasploit Pro peut être utilisé pour automatiser des attaques. Mais il existe également un autre moyen pour faire de l'automatisation avec Metasploit, sans devoir utiliser la version Pro mais celle ouverte à tous, qui est d'utiliser Metasploit RPC server (Remote Process Call). Avec cette méthode, le module se connecte au serveur Metasploit RPC et utilise la procédure console.write, ce qui permet d'écrire dans la console msfconsole. Mais alors comment automatiser Metasploit avec RPC serveur ?

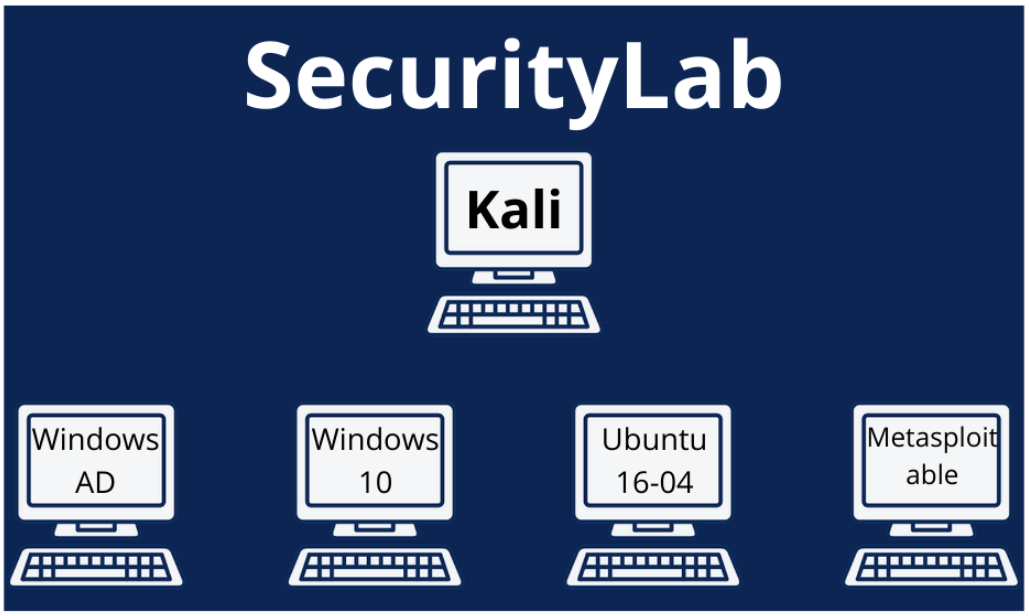
Il y a plusieurs langages de programmation pour écrire le script d'automatisation, le premier langage est Ruby. Ruby est un langage open source, qui est très connu et beaucoup utilisé pour faire de l'automatisation de Metasploit. Malheureusement, nous ne connaissons pas ce langage de programmation et il est complexe à apprendre, avec la courte durée du stage utilisé, Ruby n'est pas envisageable. Ce qui nous mène à la deuxième solution, Python. Python est aussi un langage open source, il est connu et très polyvalent grâce à ces nombreuses bibliothèques. Et c'est un langage de programmation que nous connaissons. Nous allons donc utiliser Python comme langage de programmation pour le script d'automatisation. Metasploit possède une bibliothèque pour Python, pymetasploit3.msfrpc, avec laquelle nous allons pouvoir nous connecter au serveur RPC et lui envoyer des commandes à exécuter.

2.3. Surface d'attaque

Pour augmenter les chances de réussites, il faut un maximum de failles. Donc pour cela, nous avons multiplié les cibles en augmentant le nombre de machines dans le réseau. Nous avons d'abord récupéré des versions anciennes d'OS présentes dans le stock de VM du LAB de l'entreprise. Puis, nous avons choisi une VM externe d'entrainement : Metasploitable. Pour le déploiement de Metasploitable, nous du utiliser KVM car le noyau Linux de Metasploitable n'avait pas de driver pour les cartes réseaux de VMWare.

Le réseaux SecurityLab possède désormais :

- Windows Active Directory
- Windows 10
- Metasploitable
- Ubuntu16-04



Réseau 1 Réseau SecurityLab

La deuxième étape pour augmenter la surface d'attaque a été de désactiver tous les antivirus et tous les pares-feux des machines, afin de rendre accessible un maximum de failles pour les exploits. Une fois fait, nous avons installé et activé un maximum de serveur possiblement vulnérable. Sur Windows nous avons installé un serveur web, installé différentes options de Windows (IIS, SQL Serveur). Sur Linux nous avons installé SMTP, POP, IMAP, SSH, MySQL, PostgreSQL, FTP et Telnet.

La dernière étape a été de créer un fichier Python qui se met en écoute sur tous les ports fermés non utilisés par les machines, afin que les exploits qui utilisent des ports spécifiques puissent lancer leur attaque.

L'objectif derrière ce script est de pouvoir capturer une trace d'attaque sur un port qui devrait être fermé, en espérant que l'attaque se fasse sur le premier paquet. Car il n'y a pas de serveur sur ces ports pour répondre, ce qui fait, que toute attaque devait échouer à coup sûr.

```python
import socket

def find_open_ports(start=0, end=65535):
    open_ports = []
    for port in range(start, end + 1):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.bind(('10.5.2.108', port))
            s.listen(1)
            open_ports.append(port)
            print("Listening on port", port)
        except Exception as e:
            print("Port", port," is not available:", e)
        finally:
            s.close()
    return open_ports

if __name__ == "__main__":
    find_open_ports()
```

Script 1 Ouvre et reste en écoute sur les ports fermés

3. Résultats

Dans cette partie, nous allons voir en quatre étapes les résultats du projet. Dans la première étape, nous allons voir la réalisation du script. Dans la deuxième étape, nous verrons les statistiques générales. Puis dans la troisième partie, nous verrons les statistiques de la VM WindowsAD. Et pour finir nous verrons les détails d'une attaque.

3.1. Résultats de l'automatisation

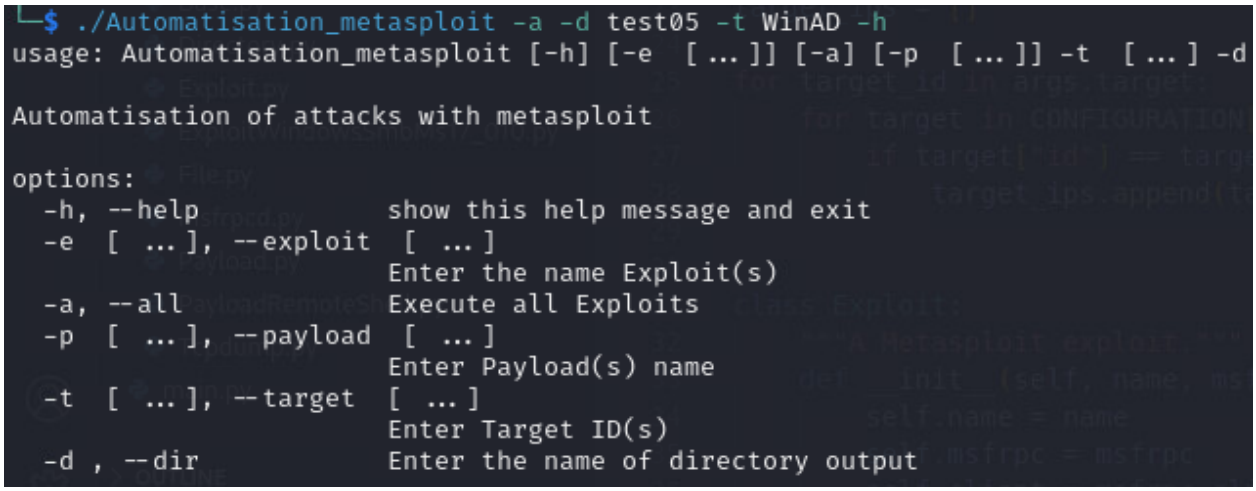
Dans cette partie nous allons voir le fonctionnement du script que nous avons réalisé, puis nous verrons les résultats obtenus à la fin de l'exécution du script.

3.1.1. Script de l'automatisation

Pour lancer le programme il faut entrer la commande suivante :

```
./Automatisation_metasploit -a -d test05 -t WinAD
```

L'argument « -a » correspond à l'utilisation de tous les exploits qui sont dans Metasploit. L'argument « -d » correspond au répertoire de sortie, dans lequel se trouveront les pcap des attaques effectuées, ainsi qu'un fichier .txt dans lequel se trouve le nom de l'exploit utilisé, la description de l'exploit, le nom de la payload utilisé, une description de la machine cible avec son IP, son OS, son port attaqué, et un message de réussite ou d'échec selon le résultat de l'attaque. L'argument « -t » correspond à la cible ou aux cibles, il est possible d'en cibler plusieurs à la fois. Il existe d'autres arguments comme « -h » qui affiche toutes les options possibles :



Script 2 Options du script d'automatisation de Metasploit

Le script fonctionne grâce à 3 boucles imbriquées, la première boucle, la plus externe est sur les cibles, la deuxième boucle, celle du milieu est sur les exploits et la dernière boucle est sur les payloads. Par exemple je lance mon script avec deux cibles Win10 et WinAD, et avec l'option « -a », le script va commencer à attaquer Win10 avec le premier exploit, et toute les payloads associés à l'exploit, une fois toutes les payloads effectuées, il va passer à l'exploit suivant et ses payloads, ainsi de suite jusqu'à finir tous les exploits. Quand il aura fini avec la première cible il va passer à la deuxième, et répéter exactement le même procédé.

Nous avons fait tourner trois fois le script, à chaque fois sur les différents OS. A la fin de l'exécution du script nous obtenons trois répertoires :

- Metasploitable
- All_Debian
- WindowsAD

Chaque répertoire contient tous les pcap des attaques qu'ils ont subits ainsi que leur fichier de description .txt.

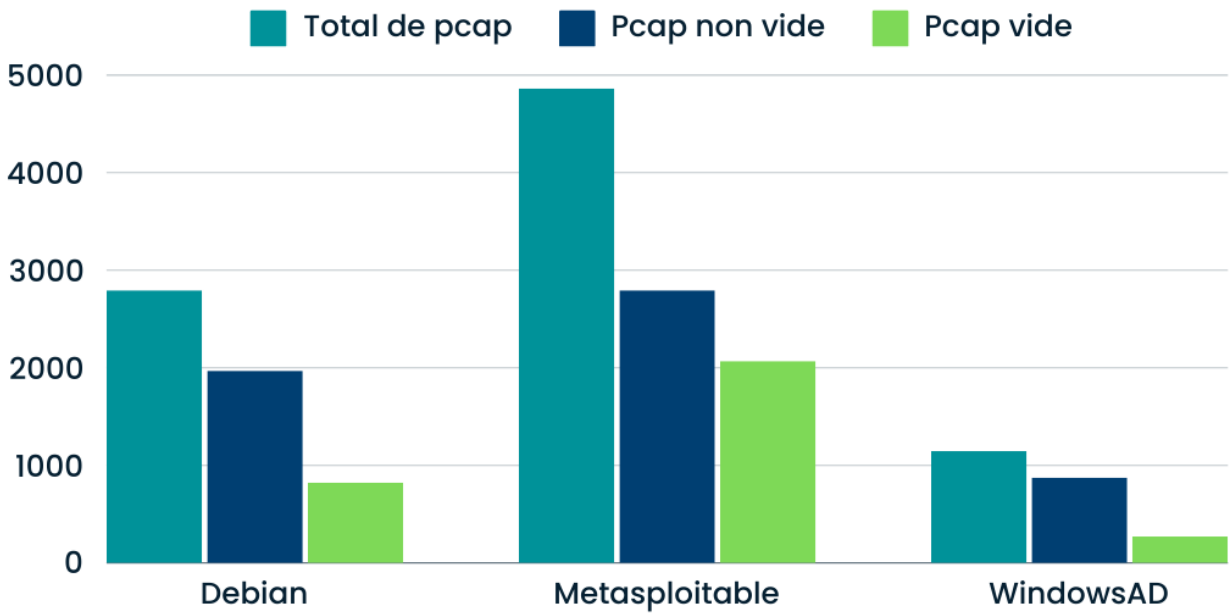
3.1.2. Statistiques

Le temps d'exécution du script lancé depuis Kali est de 18h, pour les OS Linux. Pour l'OS Windows 4h a suffi à faire tous les exploits le concernant. Le nombre de pcap récupéré dans les répertoires est de 8803, en sachant que les exploits qui ont été utilisés, ont été exécutés deux fois avec différentes payloads. Nous considérons que le nombre de pcap sorti est suffisant.

2397 exploits sont présents dans Metasploit, nous avons appliqué un filtre sur les exploits pour n'avoir que ce qui nous intéressait. Nous cherchons tous les exploits qui attaquent en passant par le réseau, nous avons donc retiré tous les exploits qui comportent le mot local dans leur nom. Ainsi que tous les exploits qui doivent générer des fichiers pour effectuer leur attaque, et les exploits qui doivent se connecter à l'extérieur du réseau.

Pour la machine Debian nous avons généré 2794 pcap, dont 1970 ne sont pas vides. Pour la machine Metasploitable nous avons généré 4863 pcap, dont 2794 ne sont pas vides. Et pour la machine WindowsAD nous avons généré 1146 pcap, dont 875 ne sont pas vides.

	Debian	Metasploitable	WindowsAD
Nombre de pcap	2794	4863	1146
Pcap non vide	1970	2794	875
Volume de fichiers	301MO	200MO	550MO
Attaques réussies selon Metasploit	1688	2855	826

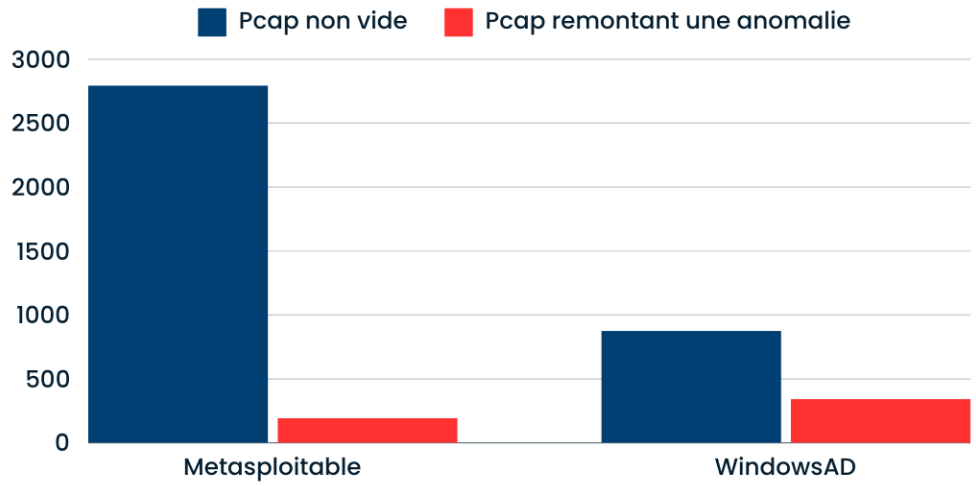


Graphique 1 Nombre de pcap généré par OS

Les pcap vides correspondent aux attaques qui n'ont pas fonctionné et qui ont laissé aucune trace sur le réseau.

Nous avons passé les pcap contenus dans les répertoires de Metasploitable, et WindowsAD dans l'ixEngine afin de constater le nombre de pcap sur lequel une anomalie est remontée. L'ixEngine donne les anomalies détectées lors de son analyse du paquet. Ces anomalies ne sont rendues publiques pour le moment que pour quelques protocoles.

Sur le graphique ci-dessous nous retrouvons le résultat des anomalies détectées par l'ixEngine. Pour Metasploitable sur les 2794 pcap, il a détecté 193 anomalies. Pour WindowsAD sur les 875 pcap, il a détecté 342 anomalies.



Graphique 2 Anomalie détecté par l'ixEngine

3.2. Analyse d'une attaque

Dans cette partie, nous vérifierons qualitativement l'une des attaques produites, en l'occurrence EternalBlue.

3.2.1. Description EternalBlue

EternalBlue est l'un des exploits les plus tristement connu, il a été conçu par le NSA dans le but d'espionnage, et était notamment utilisé dans la lutte contre le terrorisme. Mais la NSA a été piratée par un célèbre groupe de hackers Shadow Brokers. Ils ont décidé de rendre publique ce qu'ils ont volé à la NSA. Le 14 avril 2017, ils ont dévoilé via un lien sur Twitter l'outils de piratage. L'exploite fonctionne sur toutes les versions de Windows antérieurs à Windows 8, il exploite les vulnérabilités du protocole SMBv1 pour insérer des paquets de données malveillants et diffuser des malwares sur le réseau.

L'attaque EternalBlue, est une attaque qui a été corrigée par Microsoft en 2017, mais elle représente toujours une menace. Car de nombreux systèmes d'exploitation Windows ne sont toujours pas à jour dans leur mise à jour.

3.2.2. Analyse du fichier texte

Comme Metasploit possède l'exploit EternalBlue, notre script a pu lancer une attaque avec l'exploit. Nous allons analyser le fichier .txt qui a été créé par le script à la fin de l'attaque :

```
--------------------------------------------------------------------------------
 - Exploit name: windows/smb/ms17_010_eternalblue

 - Exploit description:
 This module is a port of the Equation Group ETERNALBLUE exploit, part of the
FuzzBunch toolkit released by Shadow Brokers. There is a buffer overflow memmove
operation in Srv!SrvOs2FeaToNt. The size is calculated in
Srv!SrvOs2FeaListSizeToNt, with mathematical error where a DWORD is subtracted
into a WORD. The kernel pool is groomed so that overflow is well laid-out to
overwrite an SMBv1 buffer. Actual RIP hijack is later completed in
srvnet!SrvNetWskReceiveComplete. This exploit, like the original may not trigger
100% of the time, and should be run continuously until triggered. It seems like
the pool will get hot streaks and need a cool down period before the shells rain
in again. The module will attempt to use Anonymous login, by default, to
authenticate to perform the exploit. If the user supplies credentials in the
SMBUser, SMBPass, and SMBDomain options it will use those instead. On some
systems, this module may cause system instability and crashes, such as a BSOD or
a reboot. This may be more likely with some payloads.

 - Payload name: windows/meterpreter/reverse_tcp

 - About Target:
Target OS: windows
Target Id: WinAD
Target IP: 10.5.2.116
The target port: 445

Result of attack: Successfull
```

Figure 1 EternalBlue.txt

Dans les fichiers de sortie .txt nous retrouvons le nom de l'exploit, sa description, le nom de la payload utilisé, des informations sur la cible et le résultat de l'attaque. Dans notre cas, on voit que Metasploit considère que l'attaque a fonctionné, nous allons donc analyser le pcap qui a été généré.

3.2.3. Analyse du pcapng

Le handshake entre les deux machines consiste en une négociation de protocole et une configuration de session, le ransomware se connecte au partage IPC$ de la machine cible Windows. Le logiciel malveillant est codé pour se connecter à une IP locale codée en dur, nous retrouvons l'IP de la cible lorsque que l'on regarde le pcap sur Wireshark, comme sur la figure ci-dessous.

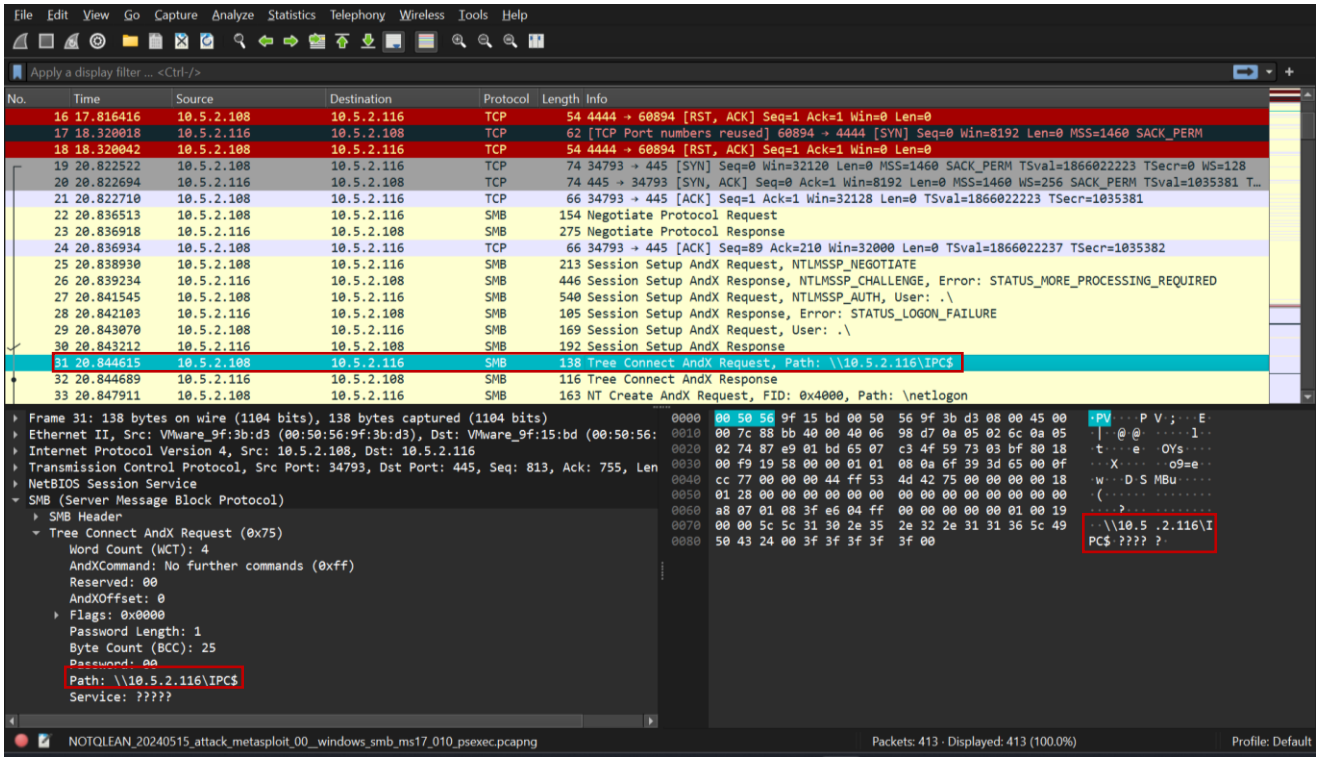


Figure 2 Connection au serveur SMB

L'attaquant envoie ensuite, une requête NT Trans, qui représente une charge utile énorme, qui est inhabituelle et nous laisse penser que l'attaque a commencé.

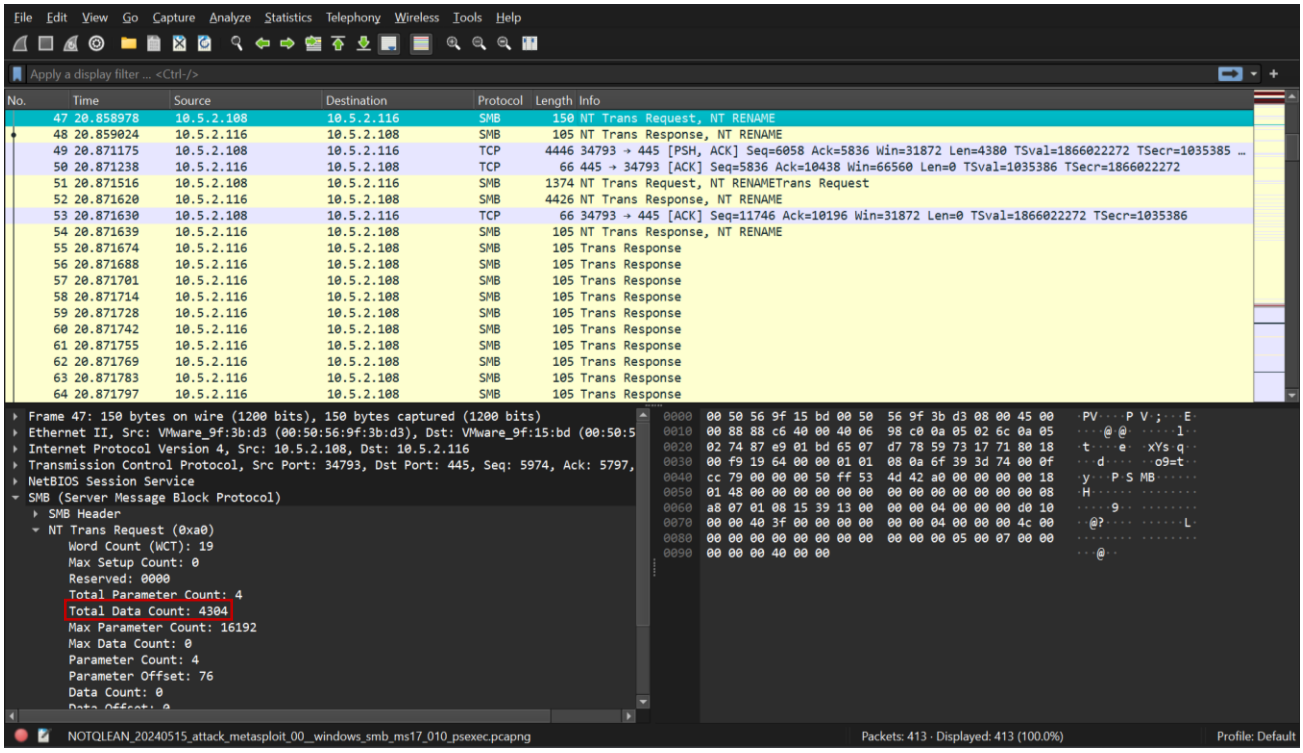


Figure 3 Anomalie sur la taille de la charge utile

Les requêtes NT RENAMETrans, contienne de grande taille de donnée dont une partie binaire. Nous supposons que la payload y est.

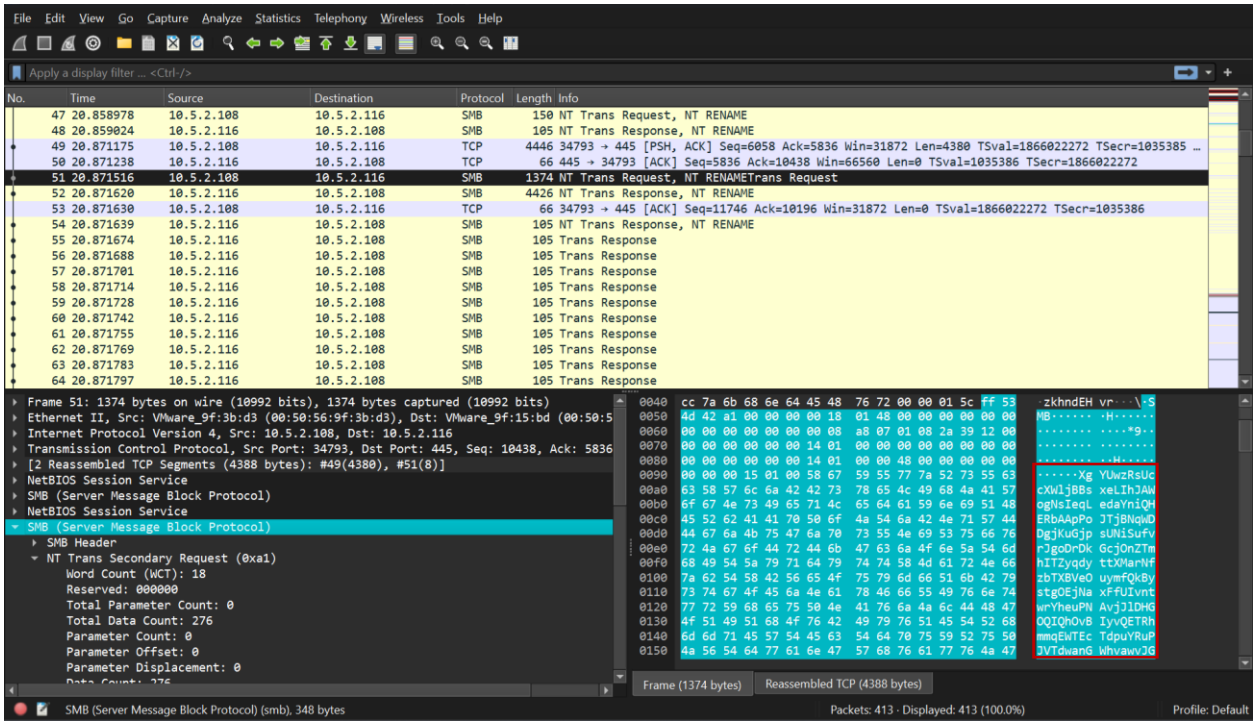


Figure 4 Charge de donnée crypté suspecte

Une fois connectée à la machine cible l'attaque est réussi, la payload est envoyée pour ouvrir un reverse shell. L'attaquant peut donc accéder à tous les dossiers présents sur la machine cible, il peut lui introduire des malwares ou se servir d'elle pour lancer d'autres attaques cachées derrière la machine qu'il vient de pirater.

Sur la capture ci-dessous nous retrouvons le process qui est exécuté sur la machine cible.

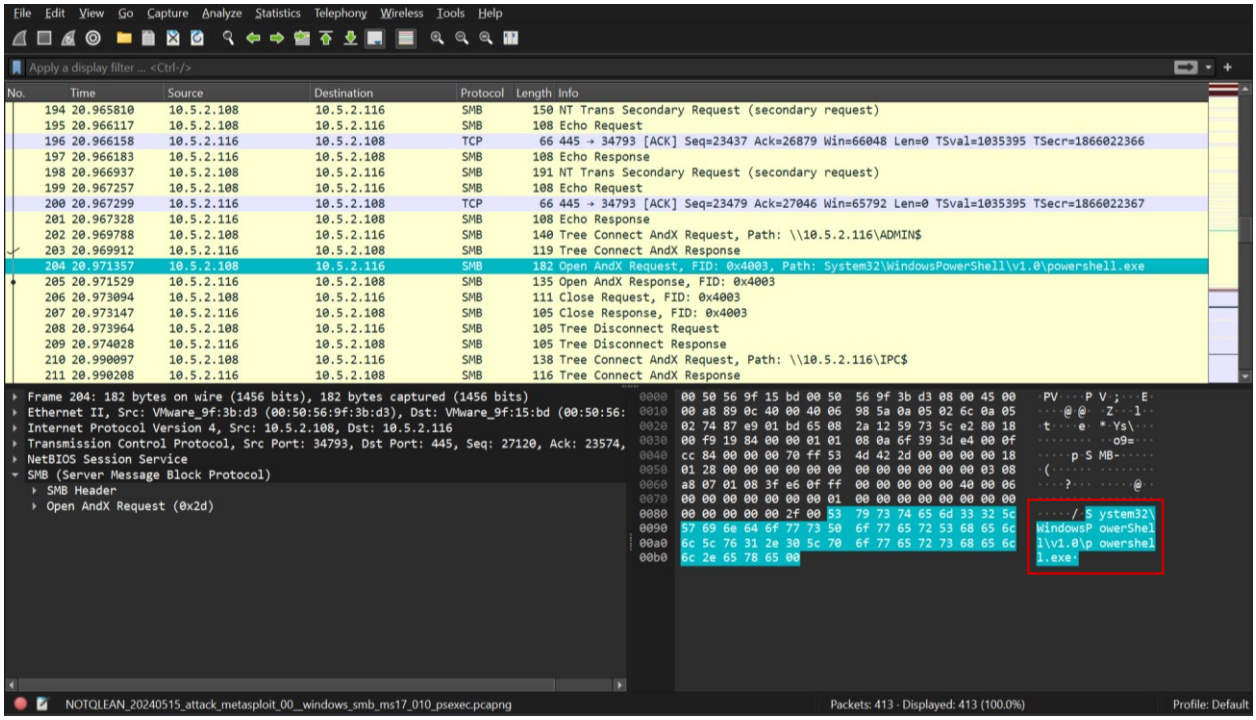


Figure 5 Exécution de PowerShell

3.3. Test du pcap sur les IDS

La détection d'attaques est la partie la plus intéressante pour nous, maintenant que nous avons vérifié le bon fonctionnement de notre automatisation. Il nous reste à savoir si les attaques sont détectables par notre produit et, ou par Suricata. Les résultats qui sortiront de notre produit permettront de nous assurer qu'il est capable de détecter les attaques, et s'il ne détecte pas d'attaques alors nous ouvrirons un ticket pour améliorer notre produit afin qu'il puisse détecter ces attaques.

3.3.1. IxEngine

dpi_nreg est un outil de test interne du SDK Qosmos ixEngine, si celui-ci détecte des anomalies dans le formatage des trames réseaux. Alors une anomalie est remontée.

```
[matgog@FRPARWLN1120 suricata]$ ~/dpi_nreg_p_1 -h .:procesing_anomaly_event -h .:processing_anomaly_attr
-h .:processing_anomaly_type WindowsAD/NOTQLEAN_20240613_attack_metasploit__wf01__windows_smb_ms17_010_et
ernalblue.pcapng
Opening WindowsAD/NOTQLEAN_20240613_attack_metasploit__wf01__windows_smb_ms17_010_eternalblue.pcapng
```

Figure 6 Détection d'anomalie avec IxEngine

Nous constatons que dans le pcap contenant l'attaque, dpi_nreg ne remonte aucune anomalie. Une anomalie sera ajoutée.

3.3.2. Suricata

Suricata est l'IDS (Système de Detection d'Intrusion) le plus connu du marché et le plus utilisé. Les règles de Suricata se basent sur ET Pro (Emerging Threat Pro Rules) [14] de Proofpoint, qui est un des leaders du marché. Tester notre pcap sur les règles de Suricata nous permettra de tester son niveau d'efficacité, et de connaitre les différences entre notre produit et celui de Suricata.

Suricata remonte trois alertes, nous allons pouvoir regarder avec plus de détails les alertes qui sont remontées.

```
17/6/2024 -- 11:19:49 - <Perf> - AutoFP - Total flow handler queues - 8
17/6/2024 -- 11:19:49 - <Info> - Alerts: 3
17/6/2024 -- 11:19:49 - <Perf> - ippair memory usage: 414144 bytes, maximum: 16777216
17/6/2024 -- 11:19:49 - <Perf> - host memory usage: 398144 bytes, maximum: 33554432
17/6/2024 -- 11:19:49 - <Info> - cleaning up signature grouping structure... complete
```

Figure 7 Alerte remontée par Suricata

Suricata nous donne les détails sur l'attaque qui s'est produite. Il commence par nous donner la date précise à laquelle l'attaque a été lancée. Il nous donne ensuite l'IP source et le port source, ainsi que l'IP de destination et le port de destination. Le protocole utilisé est TCP. La signature de l'alerte du pcap indique que c'est un exploit, et qu'il est possible que ce soit EternalBlue. Suricata donne toutes les versions de Windows pouvant être affectées par EternalBlue. Il indique que le niveau d'alerte est majeur, ce qui indique que l'attaque est très dangereuse. Pour finir il nous donne le tag de EternalBlue et Metasploit.

Pour conclure nous constatons que Suricata détecte l'attaque, et qu'il est capable de l'identifier en nous communiquant des informations sur l'exploit qui a été utilisé avec un niveau d'alerte.

```
{
  "timestamp": "2024-06-13T00:26:08.224783+0200",
  "flow_id": 45881533405264,
  "pcap_cnt": 17,
  "event_type": "alert",
  "src_ip": "10.5.2.108",
  "src_port": 44253,
  "dest_ip": "10.5.2.116",
  "dest_port": 445,
  "proto": "TCP",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2025649,
    "rev": 3,
    "signature": "ET EXPLOIT Possible ETERNALBLUE Probe MS17-010 (MSF style)",
    "category": "Unknown Classtype",
    "severity": 3,
    "metadata": {
      "affected_product": [
        "Windows_XP_Vista_7_8_10_Server_32_64_Bit"
      ],
      "attack_target": [
        "Client_Endpoint"
      ],
      "created_at": [
        "2018_07_11"
      ],
      "deployment": [
        "Internal"
      ],
      "former_category": [
        "EXPLOIT"
      ],
      "signature_severity": [
        "Major"
      ],
      "tag": [
        "ETERNALBLUE",
        "Metasploit"
      ],
      "updated_at": [
        "2019_09_28"
      ]
    }
  },
```


Figure 8 Détection d'anomalie avec Suricata

3.3.3. TD SDK

Le TD SDK est le produit qui est développé par ENEA, qui associe l'IxEngine et l'IDS Suricata. Il est exécuté avec les même règles que Suricata, et sur le même pcap.

Le TD SDK a remonté 4 anomalies sur le pcap qu'il a analysé, soit une anomalie de plus que Suricata. Nous allons regarder les détails, les règles, qui ont remontées une anomalie.

La première règle qui est remontée est de class ET INFO, cette règle nous prévient que le pcap contient possiblement un protocole SMBv1 non sûre. La deuxième anomalie remontée est de class ET EXPLOIT, la règle a détecté une possibilité que l'exploit EternalBlue soit utilisé avec probablement MS17-010 (Generique Flags). La troisième anomalie est de class ET EXPLOIT, la règle a détecté une possibilité que l'exploit EternalBlue soit utilisé avec probablement MS17-010 (MSF style). La dernière anomalie remontée est de class ET EXPLOIT, la règle a détectée une possibilité que l'exploit EternalBlue soit utilisé avec probablement un système de réponse vulnérable de MS17-010. L'attaque est donc répétée trois fois.

```
1/1/0 9 f(1) Rule SID 2023997 (ET INFO Potentially unsafe SMBv1 protocol in use) Matched
        Class: not-suspicious: Unknown Classtype
        Reference: undefinedwww.us-cert.gov/ncas/current-activity/2017/01/16/SMB-Security-Best-Practices
        Reference: undefinedblogs.technet.microsoft.com/filecab/2016/09/16/stop-using-smb1/
1/1/0 17 f(1) Rule SID 2025992 (ET EXPLOIT Possible ETERNALBLUE Probe MS17-010 (Generic Flags)) Matched
        Class: trojan-activity: Unknown Classtype
        Reference: undefinedgithub.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/smb/smb_ms17_010.rb
1/1/0 17 f(1) Rule SID 2025649 (ET EXPLOIT Possible ETERNALBLUE Probe MS17-010 (MSF style)) Matched
        Class: trojan-activity: Unknown Classtype
        Reference: undefinedgithub.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/smb/smb_ms17_010.rb
1/1/0 18 f(1) Rule SID 2025650 (ET EXPLOIT ETERNALBLUE Probe Vulnerable System Response MS17-010) Matched
        Class: trojan-activity: Unknown Classtype
        Reference: undefinedgithub.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/smb/smb_ms17_010.rb
1/1/0 0 f(2)/no ids metadata, nothing to match
```

Figure 9 Détection d'anomalie avec le TD SDK

Chaque règle possède une Rule SID qui permet de voir comment est construite la règle, et de savoir ce qui la fait matcher dans le pcap. Par exemple la première règle vérifie que la version de SMB soit la première version, si c'est la première version de SMB qui est détectée alors la règle remonte l'anomalie.

Conclusion

Pour conclure, l'objectif du stage était de créer une automatisation de Metasploit, afin d'obtenir des pcap avec des traces d'attaques, pour pouvoir tester le niveau de qualité de détection du TD SDK.

Nous avons pu démontrer que Metasploit peut être automatisé pour lancer des attaques. Nous avons réalisé cette automatisation, avec le langage de programmation Python. Le script sort un répertoire dans lequel il y a, les pcap, accompagnés de leur fichier .txt, récapitulant l'exploit utilisé, la payload utilisé et les informations de la cible qui a été visée.

Avec les pcap obtenu, nous avons testé la qualité de détection de l'IDS, de l'IxEngine et du TD SDK. Nous constatons qu'ils ont remontés peu d'anomalies ciblant la machine Metasploitable. Pour la machine Windows, ils ont remonté plus d'anomalies car les règles sont plus tournées vers Windows. A la suite des tests nous ouvrirons des tickets, sur les pcap qui n'ont pas remonté d'anomalie et pour lesquelles nous constatons une anomalie, pour améliorer notre produit.

Ce stage m'a permis d'approfondir mes compétences, notamment de programmation grâce à l'élaboration des scripts Python, et d'approfondir mes compétences d'analyse réseau grâce aux analyses que j'ai effectuées sur les pcap.

Bibliography

[1] «Enea,» [En ligne]. Available: https://www.enea.com/.

[2] r00y-3xp10it, «Automating reconnaissance and brute force attacks,» 2019. [En ligne]. Available: https://github.com/r00t-3xp10it/resource_files.

[3] paulosgf, «Ruby script to automate metasploit scanning, exploitation, and post-exploitation,» 2020. [En ligne]. Available: https://github.com/paulosgf/autoMetasploit.

[4] R. community, «Ruby,» [En ligne]. Available: https://www.ruby-lang.org/en/.

[5] «Python,» 2024. [En ligne]. Available: https://www.python.org/.

[6] «Metasploit,» 2024. [En ligne]. Available: https://www.metasploit.com/.

[7] «Kali,» 2024. [En ligne]. Available: https://www.kali.org/.

[8] «Metasploitable2,» 2024. [En ligne]. Available: https://sourceforge.net/projects/metasploitable/files/Metasploitable2/.

[9] «Metasploitable 2,» 2024. [En ligne]. Available: https://docs.rapid7.com/metasploit/metasploitable-2/.

[10] «Windows Active Directory 2012,» 2024. [En ligne]. Available: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

[11] «Windows10,» 2024. [En ligne]. Available: https://www.bing.com/search?pglt=41&q=windows10.

[12] «Ubuntu,» 2024. [En ligne]. Available: https://ubuntu.com/download.

[13] «Red Team Security Blog,» 2024. [En ligne]. Available: https://blog.parrot-pentest.com/.

[14] «Et Pro Ruleset,» Proofpoint, [En ligne]. Available: https://www.proofpoint.com/us/resources/data-sheets/et-pro-ruleset. [Accès le 06 2024].

4. Annexe

4.1. Taille des pcap de WindowsAD

Répartition de la taille des captures :

	Nombre
Fichiers vides (s=0)	271
s <1kO	563
1kO ≤ s <10kO	238
10kO ≤ s <100kO	54
100kO ≤ s <1MO	18
1MO ≤ s <340MO	2
Nombre de fichiers	1146

4.2. Ports les plus utilisés

Les ports les plus utilisé par les exploits sont :

Port TCP	Nombre de connections	Application
80	4856	HTTP
4444	1802	Kerberos
443	630	HTTPS
25	448	SMTP
8080	237	HTTP
445	182	SMB
22	154	SSH
139	150	Netbios
23	102	Telnet
143	100	IMAP
5555	71	
21	68	FTP
2381	67	
3050	60	
8000	52	
9001	45	
6667	37	
515	36	Printer
8443	36	HTTP
3000	35	
10000	32	

4.3. Analyse avec les IDS

Répartition des alertes remontées par les IDS par catégories. Il y a un total de 50 catégories pour les règles ET Pro. Uniquement 10 catégories sont utilisées :

	Suricata	TD SDK
DELETED	11	11
DNS	2	0
EXPLOIT	14	14
FTP	2	2
HUNTING	18	14
INFO	35	35
NETBIOS	2	2
POLICY	1716	1725
RPC	2	0
TFTP	2	2

4.4. Protocoles classifiés par l'ixEngine

Exemples de classifications (protocoles remontés) par l'IxEngine pour Windows et Debian :

Windows AD 2012	Debian
base.arp base.ip6.icmp6 base.ip.icmp base.ip.tcp.http base.ip.tcp.http.owa base.ip.tcp.http.soap base.ip.tcp.rpc.portmap base.ip.tcp.smb base.ip.tcp.smb.dcerpc base.ip.tcp.smb.dcerpc.msrpc base.ip.tcp.ssh base.ip.tcp.ssl base.ip.tcp.ssl.https base.ip.tcp.ssl.https.cloudflare base.ip.tcp.ssl.https.google_gen.dns base.ip.tcp.ssl.https.microsoft base.ip.tcp.ssl.https.windows_azure base.ip.udp.dhcp base.ip.udp.dns base.ip.udp.nbns base.ip.udp.netbios.smb.mailslot base.ip.udp.ntp base.ip.udp.quic.ssl.http3.google_gen.dns	base.arp base.ip6.icmp6 base.ip.icmp base.ip.tcp base.ip.tcp.http base.ip.tcp.http.soap base.ip.tcp.http.ssh base.ip.tcp.java_rmi base.ip.tcp.ssh base.ip.tcp.ssl base.ip.tcp.ssl.https base.ip.tcp.ssl.https.cloudflare base.ip.tcp.ssl.https.google_gen.dns base.ip.tcp.tds base.ip.tcp.telnet base.ip.tcp.tns base.ip.udp base.ip.udp.dhcp base.ip.udp.dns base.ip.udp.nbns base.ip.udp.netbios.smb.mailslot base.ip.udp.ntp

base.ip.udp.sip base.ip.udp.tftp	base.ip.udp.quic.ssl.http3.google_gen.dns base.ip.udp.rpc.portmap base.ip.udp.sip base.ip.udp.snmp base.ip.udp.ssdp base.ip.udp.ssdp.upnp

Index

Lexique

DPI : Deep Packet Inspection est une méthode avancée pour examiner le trafic réseau. Ce mécanisme évalue l'en-tête et la charge utile d'un paquet transmis par un point d'inspection.

SDK : Software Development Kit (SDK), est un ensemble d'outils et de bibliothèques fournis aux développeurs.

IDS : Intrusion Detection System (IDS), est un système de sécurité passif qui surveille le trafic réseau ou les activités du système.

pcap : pcap est un format de fichier pour les captures réseau.

Table des illustrations