tests d’intrusion - Python

m2iformation.fr

Sommaire

- Introduction aux tests d'intrusion
- Types de tests d'intrusion: Tests internes vs externes, tests en boîte noire, en boîte blanche et en boîte grise.
- Processus de tests d'intrusion: Phases typiques d'un test d'intrusion.
- Utilisation de Python pour la découverte de réseau:
  - Écriture de scripts pour scanner les ports avec socket.
  - Utilisation de Scapy pour les tâches de reconnaissance de réseau.
- Utilisation de Python pour l'automatisation de Nmap.
  - Analyse des résultats et identification des cibles potentielles.
  - Introduction aux techniques de contournement des pare-feux et IDS.
- Développement de scripts d'exploitation en Python.
- Utilisation de frameworks d'exploitation existants (ex. Metasploit).
- Exercices pratiques d'exploitation de vulnérabilités courantes (ex. buffer overflow, SQL injection).
- Techniques d'escalade de privilèges.
- Automatisation des attaques avec Python.

Sommaire

- Collecte d'informations système supplémentaires.
- Maintien de l'accès avec des backdoors Python.
- Nettoyage des traces d'attaque.
- Importance du rapport et communication des résultats.

Introduction aux tests d’intrusion: Définitions, importance, et éthique

Les tests d'intrusion, également connus sous le nom de **penetration testing** ou pentesting, sont des simulations d'attaques informatiques sur un système pour identifier des failles de sécurité. Le but est de découvrir ces vulnérabilités avant qu'un attaquant malveillant ne puisse les exploiter.

Les tests d'intrusion sont cruciaux pour la sécurité informatique car ils permettent de :

- **Détecter et corriger les vulnérabilités** avant qu'elles ne soient exploitées.
- **Évaluer l'efficacité des mécanismes de défense** en place.
- **Assurer la conformité** avec les normes de sécurité internationales et les réglementations légales.
- **Protéger les actifs de l'entreprise** et maintenir la confiance des clients et des partenaires.

L'éthique dans les tests d'intrusion est fondamentale. Les pentesters doivent toujours :

- **Obtenir une autorisation explicite** avant de commencer les tests.
- **Respecter la confidentialité** des informations découvertes.
- **Agir de manière responsable** en évitant de causer des dommages ou des interruptions dans le réseau ou le système cible.
- **Rapporter toutes les vulnérabilités découvertes** aux parties responsables.

Introduction aux tests d'intrusion: Définitions, importance, et éthique

- **Tests internes** : Effectués depuis l'intérieur du réseau de l'entreprise. Ils simulent une attaque par un utilisateur malveillant avec des accès internes ou par un employé ayant des intentions malicieuses.
- **Tests externes** : Réalisés depuis l'extérieur du réseau de l'entreprise. Ils visent à identifier ce qu'un attaquant externe pourrait exploiter sans accès préalable.
- **Boîte noire** : Le pentester a peu ou pas d'information préalable sur le système. Cela simule une attaque à l'aveugle où le pentester doit découvrir le système en temps réel.
- **Boîte blanche** : Le pentester a accès à des informations complètes sur l'infrastructure, y compris les schémas de réseau, le code source, les configurations d'API, etc. Cela permet de réaliser une analyse exhaustive.
- **Boîte grise** : Une approche intermédiaire où le pentester a quelques informations sur le système, mélangeant les éléments des tests en boîte noire et en boîte blanche.

Introduction aux tests d'intrusion: Définitions, importance, et éthique

Phases typiques

1. **Planification et reconnaissance** :
   - Définir le périmètre et les objectifs du test.
   - Collecter des informations sur le système cible pour identifier les points d'entrée potentiels.
2. **Analyse et identification des vulnérabilités** :
   - Scanner le système pour détecter les vulnérabilités existantes à l'aide d'outils automatisés ou de techniques manuelles.
3. **Exploitation** :
   - Tenter d'exploiter les vulnérabilités découvertes pour évaluer l'impact potentiel sur le système.
4. **Post-exploitation** :
   - Déterminer ce qui peut être réalisé une fois l'accès obtenu, tel que l'accès à des données sensibles, l'escalade des privilèges, etc.
5. **Rapport et recommandations** :
   - Rédiger un rapport détaillé des vulnérabilités découvertes, des données exploitables obtenues, et fournir des recommandations pour la mitigation des risques identifiés.
6. **Réévaluation** :
   - Après que les correctifs ont été appliqués, réaliser un nouveau cycle de tests pour s'assurer que les vulnérabilités ont été effectivement résolues.

Utilisation de Python pour la découverte de réseau

Le module `socket` en Python permet de créer des connexions réseau, qui peuvent être utilisées pour tester la disponibilité des ports sur une machine distante. Un script de scan de ports essaie de se connecter à une série de ports d'une machine spécifique pour identifier quels ports sont ouverts et écoutent pour des connexions, ce qui est un indicateur de quels services peuvent être accessibles depuis l'extérieur.

```
import socket

ip = "192.168.1.1"  # Remplacez par l'IP de la cible
port_list = [22, 80, 443]  # Liste des ports à scanner

for port in port_list:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    result = sock.connect_ex((ip, port))
    if result == 0:
        print(f"Port {port} is open on {ip}.")
    else:
        print(f"Port {port} is closed on {ip}.")
    sock.close()
```

Intérêt

- **Détection rapide des services exposés** : Identifier les ports ouverts permet de comprendre quels services sont accessibles, ce qui est crucial pour évaluer la surface d'attaque d'un système.
- **Personnalisation** : Les scripts peuvent être adaptés aux besoins spécifiques, comme scanner des plages de ports ou intégrer des fonctionnalités plus complexes.

Utilisation de Python pour la découverte de réseau

`Scapy` est une bibliothèque Python puissante utilisée pour la manipulation de paquets réseau. Elle permet non seulement de créer et d'envoyer ses propres paquets, mais aussi d'intercepter et d'analyser les paquets envoyés par d'autres dispositifs sur le réseau.

```python
from scapy.all import ARP, Ether, srp
target_ip = "192.168.1.1/24"  # Spécifiez la plage IP
# Crée un paquet ARP
arp = ARP(pdst=target_ip)
ether = Ether(dst="ff:ff:ff:ff:ff:ff")
packet = ether/arp
# Envoie le paquet et attend les réponses
result = srp(packet, timeout=3, verbose=0)[0]
# Affiche les IP et MAC des hôtes découverts
for sent, received in result:
    print(f"IP: {received.psrc}, MAC: {received.hwsrc}")
```

Intérêt

- **Reconnaissance avancée** : `Scapy` permet une reconnaissance détaillée du réseau, y compris la découverte d'hôtes, l'identification des systèmes d'exploitation et la cartographie des topologies réseau.
- **Flexibilité et puissance** : Il peut être utilisé pour une large gamme de tâches, de la simple écoute à des attaques complexes comme le spoofing ou les attaques par déni de service.

Exercice 1 - Scanner de Ports avec socket

Objectif : Écrire un script Python qui scanne une gamme de ports sur une machine cible pour déterminer lesquels sont ouverts.

Description : Les participants créeront un script en Python utilisant le module socket pour tenter de se connecter à une série de ports sur une adresse IP spécifiée. L'objectif est d'identifier les ports ouverts sur un serveur de test pré-configuré.

Exercice 2: Découverte de Réseau avec Scapy

Objectif : Utiliser Scapy pour découvrir les appareils actifs sur un réseau local.

Description :

Les participants utiliseront Scapy pour écrire un script qui envoie des requêtes ARP (Address Resolution Protocol) sur un réseau local et capture les réponses pour identifier les appareils présents.

Sujet de TP : Détection et Analyse des Attaques ARP Spoofing sur un Réseau Local

Dans un environnement réseau local, les attaques ARP Spoofing sont une menace courante qui peut permettre à un attaquant de réaliser des attaques de type “man-in-the-middle”, interceptant ou altérant le trafic entre deux machines sans être détecté. Ces attaques exploitent le protocole ARP, qui est utilisé pour associer les adresses IP aux adresses MAC sur un réseau local. Le TP proposé permettra aux étudiants d’utiliser Scapy pour détecter la présence de telles attaques sur un réseau simulé et d’analyser les paquets pour identifier l’attaquant.

Objectif du TP :

- Comprendre le fonctionnement du protocole ARP et les mécanismes d’une attaque ARP Spoofing.
- Utiliser Scapy pour capturer et analyser les paquets ARP afin de détecter des anomalies indiquant une attaque.
- Développer un script Python avec Scapy qui alerte en temps réel lorsqu’une attaque ARP Spoofing est détectée.

Description du TP :

Les étudiants devront écrire un script Python en utilisant Scapy pour écouter les paquets ARP sur le réseau. Le script doit analyser ces paquets pour détecter si plusieurs réponses ARP (ARP replies) contiennent la même adresse IP mais des adresses MAC différentes, ce qui est un indicateur potentiel d’une attaque ARP Spoofing.

Utilisation de Python pour l'automatisation de Nmap

Nmap (Network Mapper) est un outil de scan de réseau puissant et polyvalent utilisé pour découvrir des informations sur les réseaux informatiques.

1. Découverte des Hôtes: Nmap permet d'identifier quels hôtes sont actifs sur un réseau donné. Cela inclut non seulement les ordinateurs et serveurs, mais aussi d'autres dispositifs réseau comme les routeurs, les commutateurs, les imprimantes, etc. En envoyant différents types de paquets IP et en analysant les réponses, Nmap peut lister les appareils qui répondent sur le réseau.

2. Scan des Ports: L'une des utilisations principales de Nmap est de scanner les ports de réseau des hôtes pour déterminer quels ports TCP ou UDP sont ouverts. Un port ouvert sur un hôte indique généralement qu'un service réseau (comme un serveur web, un serveur de fichiers, etc.) est en écoute et accessible via ce port. Cela aide à comprendre la surface d'attaque potentielle d'un système.

3. Détection des Services et Versions: Au-delà de simplement identifier les ports ouverts, Nmap peut également tenter de déterminer quels services s'exécutent sur ces ports, ainsi que leurs versions. Par exemple, il peut détecter si un serveur web Apache ou un serveur FTP est en cours d'exécution, et quelle version de ces logiciels est utilisée. Cette information est cruciale pour identifier les logiciels potentiellement vulnérables.

Utilisation de Python pour l'automatisation de Nmap

4. Détection des Systèmes d'Exploitation: Nmap peut utiliser des techniques de fingerprinting pour deviner le système d'exploitation (OS) d'un hôte distant, basé sur les caractéristiques de son trafic réseau. Connaître le système d'exploitation peut aider à planifier des tests de sécurité ou des attaques, car différents OS ont différentes vulnérabilités.

5. Audit de Sécurité: Les administrateurs réseau et les auditeurs de sécurité utilisent Nmap pour vérifier l'état de la sécurité de leurs réseaux. En scannant régulièrement leur réseau, ils peuvent découvrir de nouveaux dispositifs non autorisés, vérifier que les politiques de sécurité du réseau sont bien appliquées et identifier les services inutiles qui devraient être désactivés.

Exercice

Vous êtes un analyste SOC pour une entreprise avec un réseau interne qui contient plusieurs services critiques, y compris un serveur web, un serveur de bases de données, et des systèmes de gestion interne. Votre tâche est de scanner régulièrement le réseau pour détecter toute activité suspecte ou non autorisée et d'analyser les résultats pour identifier les risques potentiels

TP : Analyse de Sécurité en Boîte Noire sur DVWA et Juice Shop

Objectif du TP: Ce travail pratique vise à familiariser les stagiaires avec les techniques d'analyse de sécurité en boîte noire à travers l'utilisation de scripts Python, Nmap pour le scanning de réseau, et une base de données de vulnérabilités pour identifier et proposer des correctifs aux failles détectées dans deux applications web intentionnellement vulnérables : DVWA (Damn Vulnerable Web Application) et OWASP Juice Shop.
Les stagiaires seront chargés de développer des scripts en Python qui utilisent Nmap pour scanner les applications DVWA et Juice Shop déployées localement ou sur un réseau de test.

1. **Installation et Configuration**
   - Assurez-vous que Python, Nmap et les bibliothèques Python nécessaires sont installés.
2. **Développement du Script de Scanning avec Nmap**
   - Créer un script Python qui utilise pour scanner les adresses IP où DVWA et Juice Shop sont hébergés.
   - Le script doit identifier les ports ouverts et les services associés à ces ports.
3. **Interrogation d'une Base de Données de Vulnérabilités**
   - Utiliser une API de base de données de vulnérabilités pour rechercher des informations sur les vulnérabilités des services découverts.
   - Intégrer cette fonctionnalité dans le script Python pour automatiser le processus de correspondance entre les services et les vulnérabilités.

TP : Analyse de Sécurité en Boîte Noire sur DVWA et Juice Shop

4. **Analyse des Résultats et Rédaction d'un Rapport**
    - Analyser les données recueillies par le script pour identifier les vulnérabilités potentielles.
    - Rédiger un rapport qui présente chaque vulnérabilité, explique l'impact potentiel et propose des correctifs ou des mesures d'atténuation appropriées.

Introduction aux techniques de contournement des pare-feux et IDS.

1. Comprendre Pare-feux et IDS

- **Pare-feux** : Un pare-feu est un dispositif de sécurité réseau qui surveille et contrôle le trafic entrant et sortant selon des règles prédéfinies. Il agit généralement comme une barrière entre un réseau de confiance interne et un réseau externe non fiable.
- **IDS** : Un système de détection d'intrusions surveille le trafic réseau et système pour détecter des activités anormales ou malveillantes. Les IDS peuvent être basés sur des signatures (détectant des modèles connus d'attaques) ou sur des anomalies (détectant des déviations par rapport à un modèle de comportement normal).

2. Techniques de Contournement des Pare-feux

- **Changement de port** : Utiliser des ports non standards pour les communications, car certains pare-feux bloquent uniquement les ports connus utilisés pour des applications spécifiques.
- **Tunneling** : Encapsuler le trafic illégitime dans des protocoles légitimes. Par exemple, encapsuler du trafic SSH (port 22) dans du trafic HTTP (port 80) pour échapper à la détection.
- **VPN et TOR** : Utiliser des réseaux privés virtuels (VPN) ou The Onion Router (TOR) pour masquer l'origine et la nature du trafic.

Introduction aux techniques de contournement des pare-feux et IDS.

3. Techniques de Contournement des IDS

- **Fragmentation des paquets** : Diviser les données malveillantes en plusieurs petits paquets. Certains IDS peuvent ne pas reconstituer correctement ces paquets pour analyser le contenu complet.
- **Obfuscation des charges utiles** : Modifier la présentation des charges utiles malveillantes pour éviter la détection basée sur les signatures. Cela peut inclure l'encodage, le chiffrement, ou la modification des formats de données.
- **Utilisation de trafic chiffré** : Utiliser HTTPS ou d'autres formes de chiffrement pour masquer le contenu du trafic. Les IDS ont du mal à inspecter le contenu chiffré sans des capacités de décryptage configurées.
- **Génération de faux positifs** : Flooder l'IDS avec un grand nombre d'alertes pour le submerger, réduisant ainsi son efficacité à détecter des attaques réelles.

4. Exemples de Contournement en Pratique

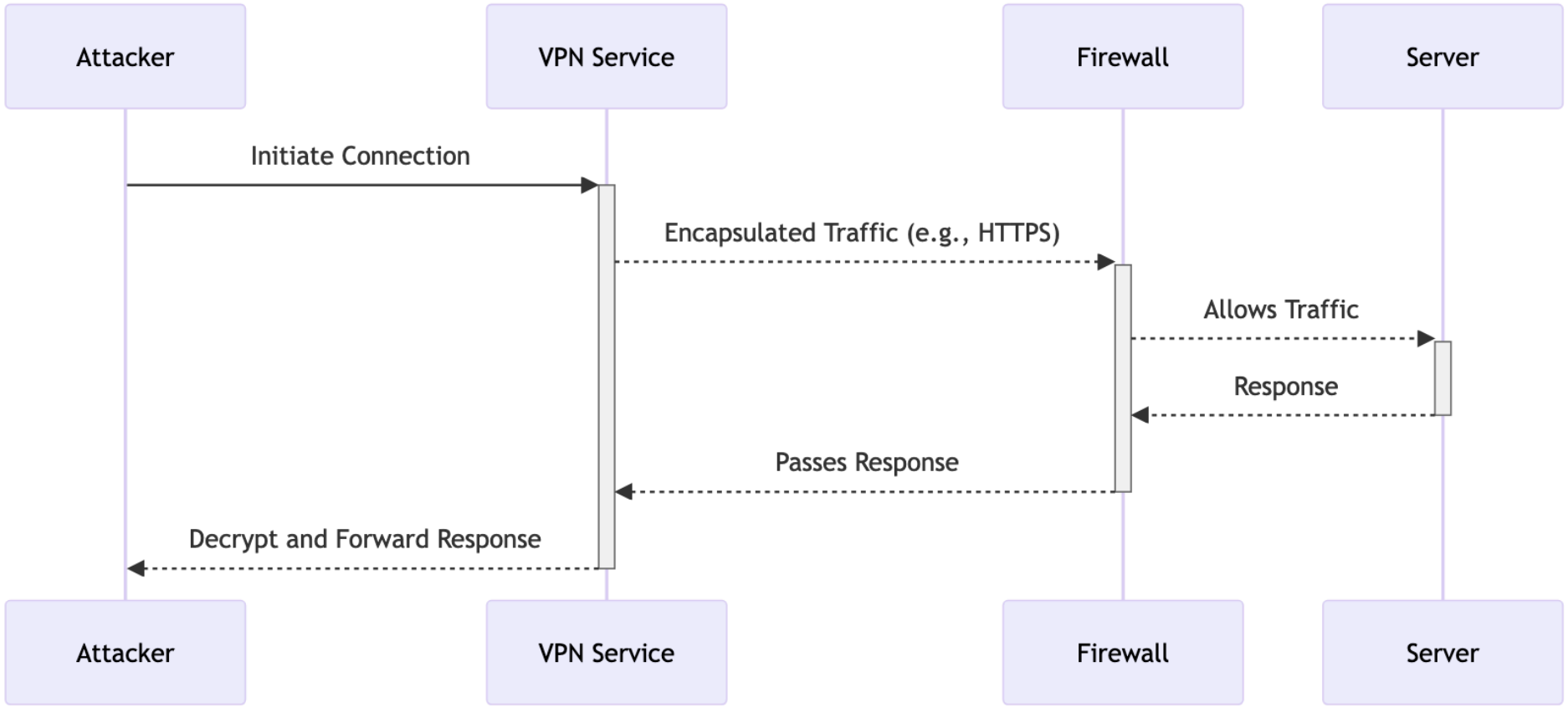
- **Attaques par tunneling** : Un attaquant encapsule le trafic de commande et de contrôle (C&C) à l'intérieur du trafic Web pour échapper à un pare-feu qui ne permet que le trafic HTTP/HTTPS.
- **Fragmentation IP pour échapper aux IDS** : En divisant une requête HTTP malveillante en plusieurs segments IP, un attaquant pourrait éviter que l'IDS reconnaisse la nature malveillante de la requête complète.
- **Chiffrement SSL/TLS** : En utilisant le chiffrement SSL/TLS pour masquer les commandes de botnet ou les exfiltrations de données, les attaquants peuvent éviter la détection par les IDS qui ne peuvent pas inspecter le contenu chiffré.

Introduction aux techniques de contournement des pare-feux et IDS.

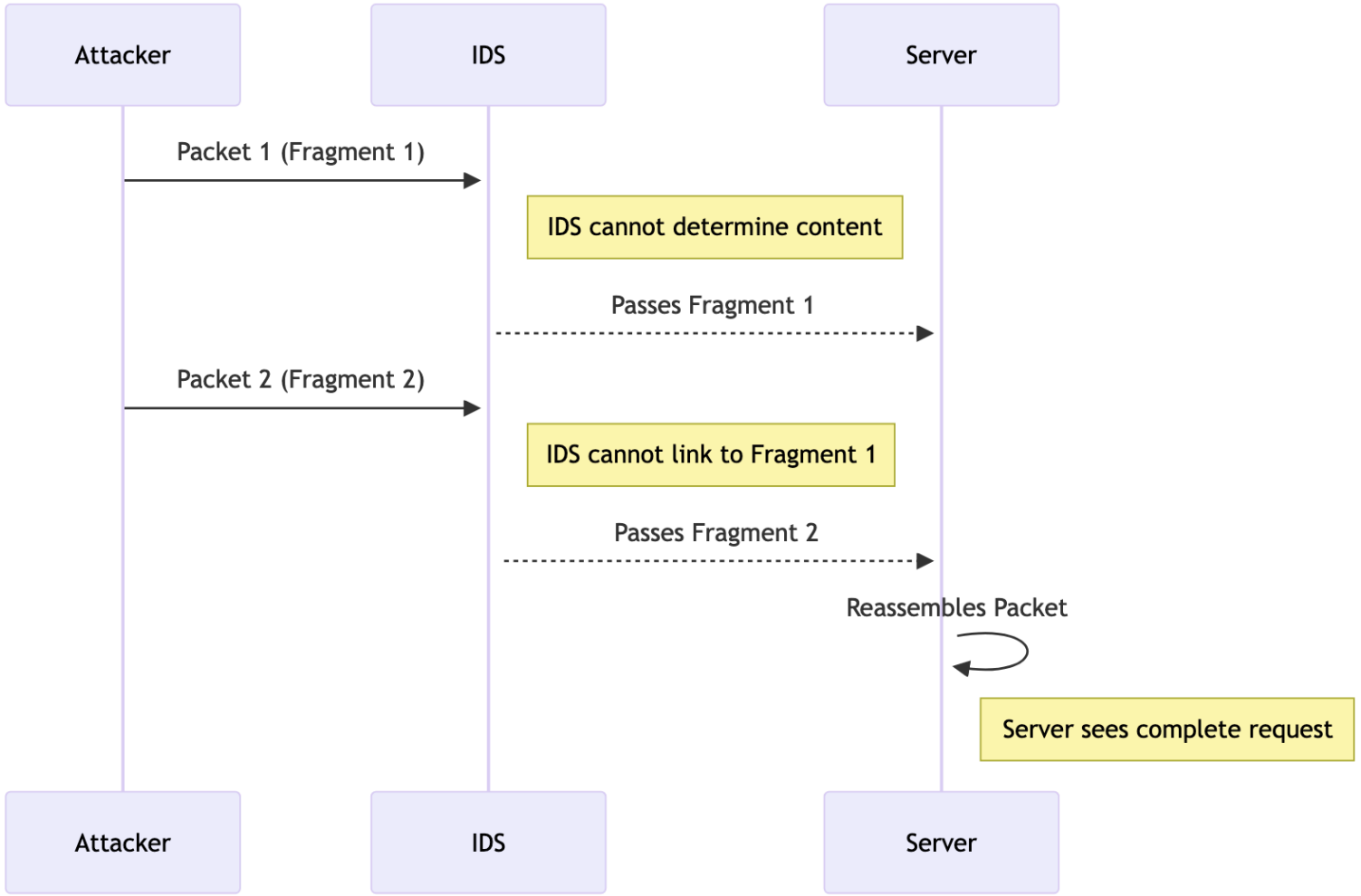
5. Défenses et Contre-Mesures

- **Inspection approfondie des paquets** : Utiliser des dispositifs capables d'effectuer une inspection en profondeur des paquets (DPI) pour analyser le contenu même sur les ports non standards ou le trafic chiffré.
- **Renforcement des règles IDS et pare-feu** : Maintenir à jour les signatures IDS et les règles de pare-feu basées sur les nouvelles vulnérabilités et techniques de contournement découvertes.
- **Surveillance et analyse comportementale** : Combiner les IDS basés sur les signatures avec des systèmes de détection basés sur des anomalies pour détecter les activités suspectes qui pourraient autrement passer inaperçues.

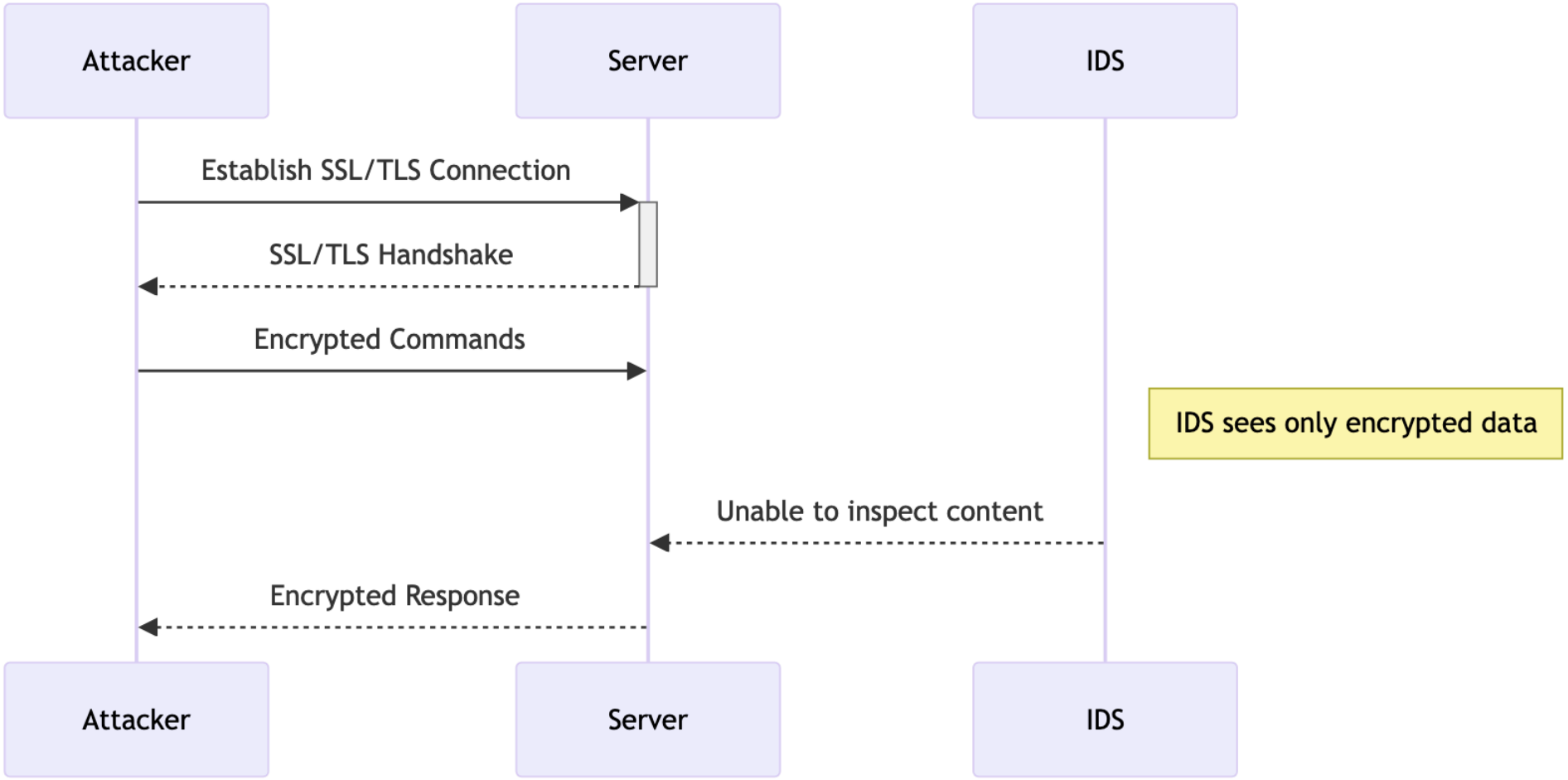
Introduction aux techniques de contournement des pare-feux et IDS.



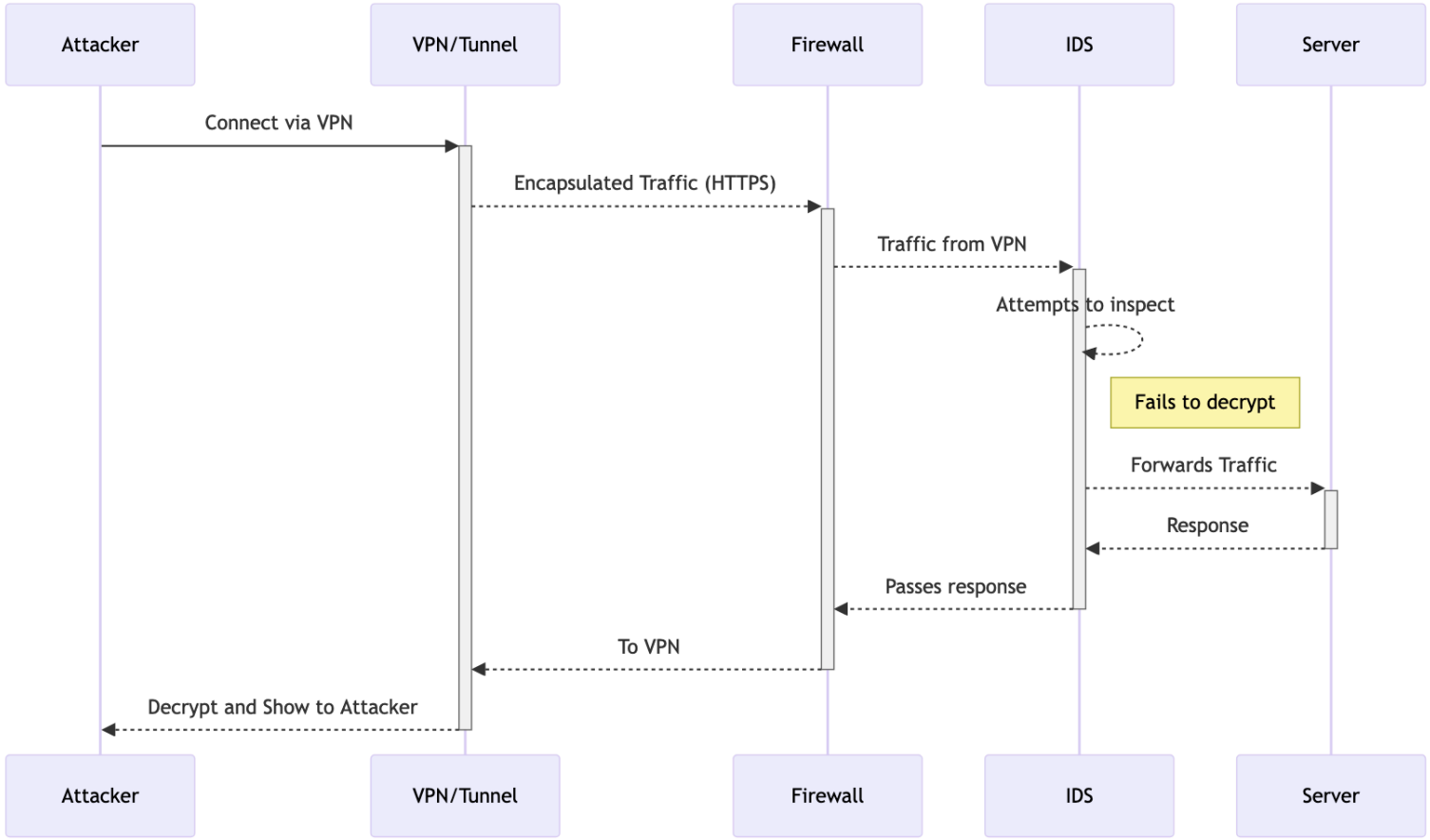
Introduction aux techniques de contournement des pare-feux et IDS.



Introduction aux techniques de contournement des pare-feux et IDS.



Introduction aux techniques de contournement des pare-feux et IDS.



Exercice : Simulation de Contournement d'un IDS avec Scapy

- Les analystes SOC doivent produire des scripts pour contourner un IDS (Suricata) en utilisant divers techniques sur une application vulnérable (JuiceShop).

1. Fragmentation des Paquets : Diviser les données malveillantes en plusieurs petits paquets.
2. Obfuscation des Charges Utiles : Modifier la présentation des charges utiles pour éviter la détection basée sur les signatures (par exemple, encodage Base64).
3. Utilisation de Trafic Chiffré : Utiliser HTTPS pour masquer le contenu du trafic.
4. Génération de Faux Positifs : Flooder l'IDS avec un grand nombre d'alertes pour le submerger.

Utilisation de Metasploit

- Metasploit est un framework de test de pénétration largement utilisé qui permet aux utilisateurs de découvrir, d'exploiter et de valider les vulnérabilités dans les systèmes. Il offre un ensemble riche en modules, y compris des exploits, des payloads, des post-exploitations, des encodeurs, et des nops. Bien que Metasploit soit écrit en Ruby, il peut être contrôlé à distance via une API RPC, ce qui permet son utilisation avec d'autres langages de programmation, notamment Python.
- L'API RPC (Remote Procedure Call) permet une interaction programmatique avec Metasploit, rendant possible l'automatisation et la personnalisation des tâches de test de pénétration.

1. Démarrage du service RPC :

Pour utiliser l'API RPC, vous devez d'abord démarrer le service RPC de Metasploit. Vous pouvez le faire en exécutant le service `msfrpcd` sur la machine où Metasploit est installé.

```
msfrpcd -P your_password -n -f -a 127.0.0.1
```

- `-P your_password` définit le mot de passe pour l'accès à l'API.
- `-n` désactive le serveur de base de données.
- `-f` force l'exécution en arrière-plan.
- `-a 127.0.0.1` lie le serveur RPC à l'adresse localhost pour des raisons de sécurité.

Utilisation de Metasploit

2. Installation du package Python :

Utilisez `pymetasploit3`, un wrapper Python pour l'API RPC de Metasploit.

```
pip install pymetasploit3
```

Utilisation de l'API avec Python

1. Connexion à l'API RPC :

Créez un script Python pour se connecter à l'API et réaliser des opérations de base.

```
from pymetasploit3.msfrpc import MsfRpcClient

# Connexion au client RPC
client = MsfRpcClient('your_password', server='127.0.0.1', port=55553)
```

2. Choisir et configurer un exploit :

Sélectionnez un module d'exploit et configurez-le avec les options nécessaires.

```
exploit = client.modules.use('exploit', 'exploit_name')  # Remplacez 'exploit_name' par le nom réel de l'exploit
exploit['RHOSTS'] = '192.168.1.101'  # IP cible
```

3. Définir un payload :

Choisissez un payload approprié pour l'exploit et configurez-le.

```
payload = client.modules.use('payload', 'payload_name')  # Remplacez 'payload_name' par le nom réel du payload
payload['LHOST'] = '192.168.1.100'  # Votre IP pour la connexion inverse
payload['LPORT'] = 4444  # Port sur lequel écouter la connexion inverse
```

4. Exécuter l'exploit :

```
output = exploit.execute(payload=payload)
print(output)
```

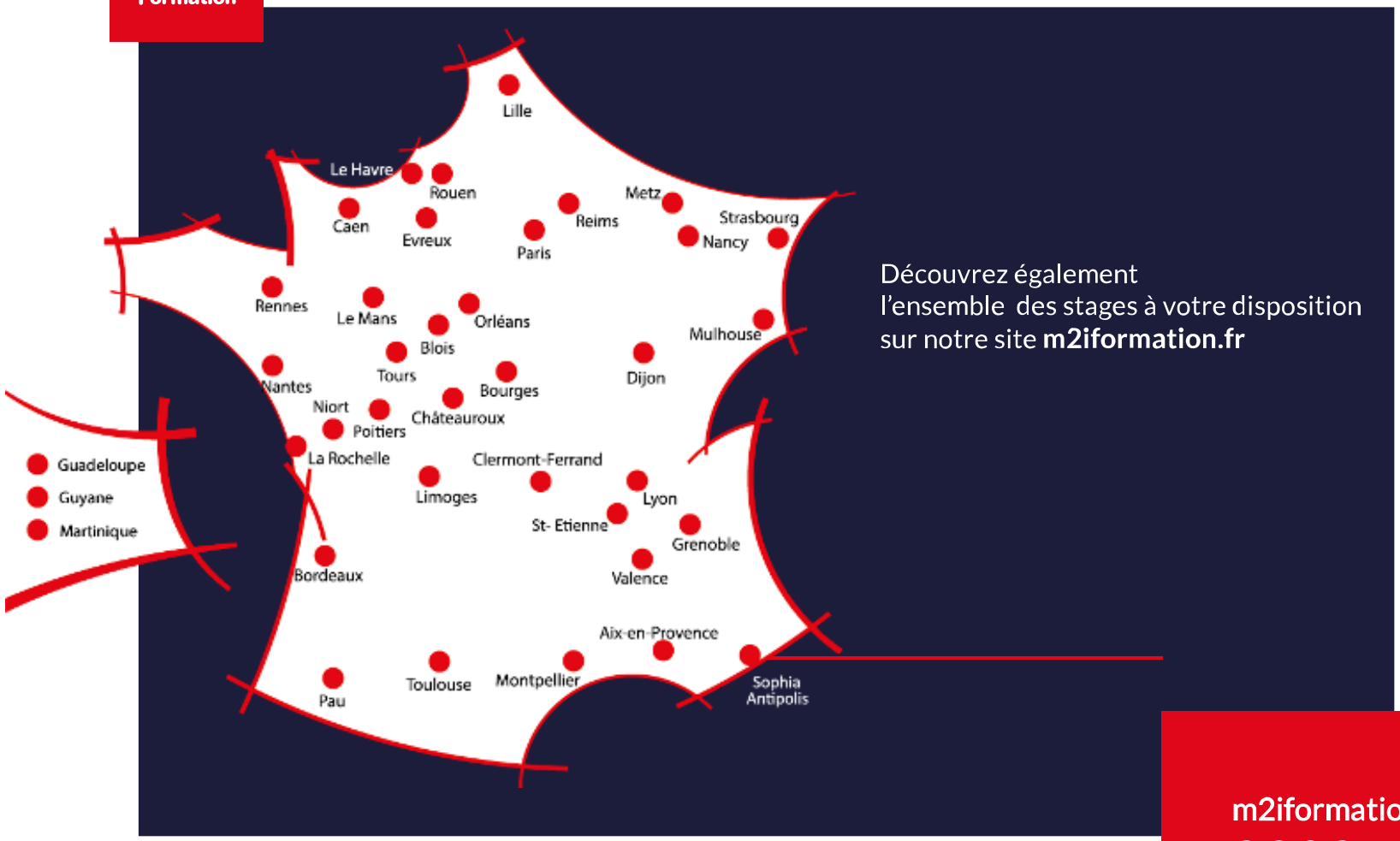
Exercice

- En utilisant metasploit et Python, essayez d'exploiter les vulnérabilités les plus dangereuse sur JuiceShop.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Merci pour votre attention

Des questions ?



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr