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.

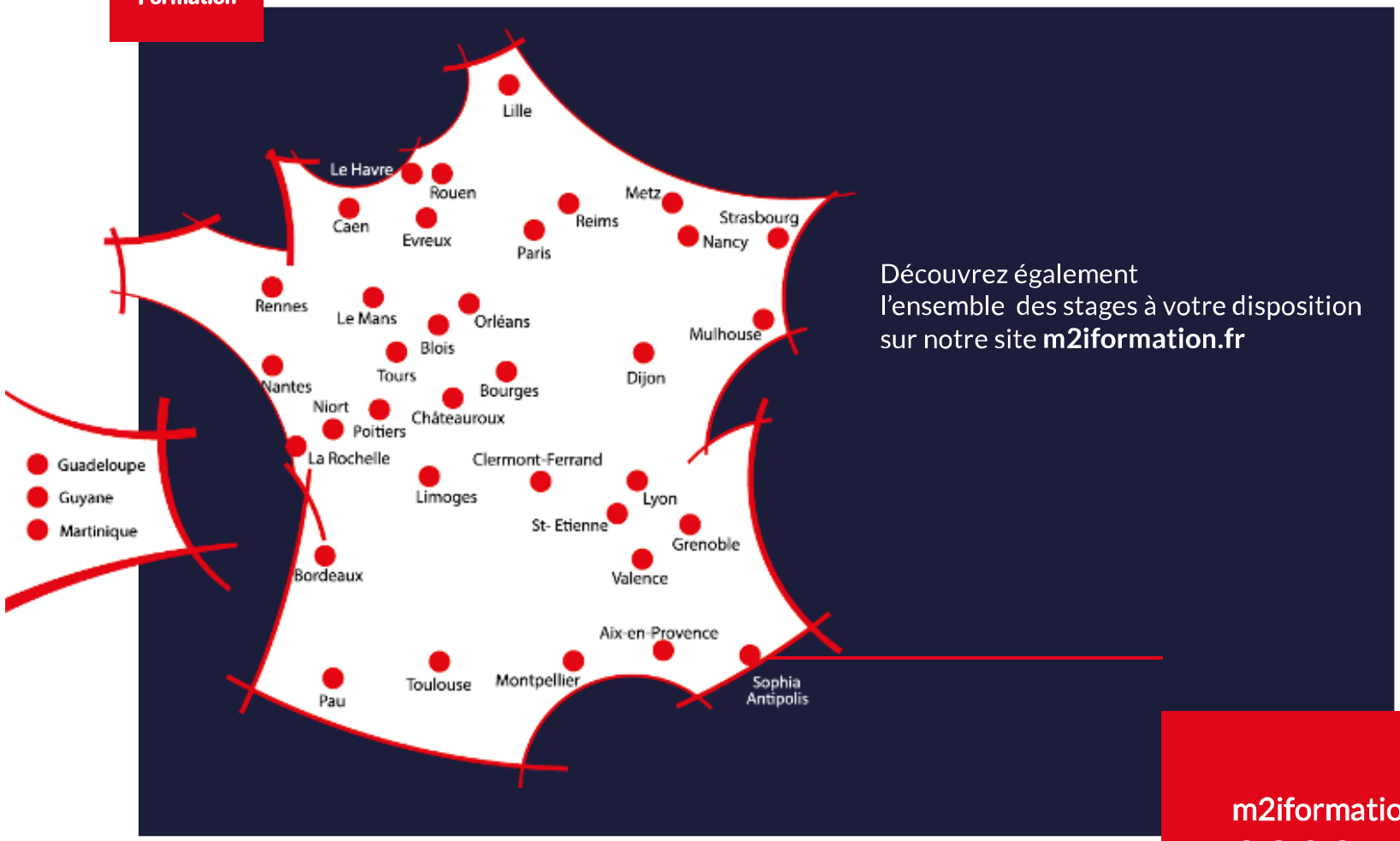
Description du TP :

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.

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