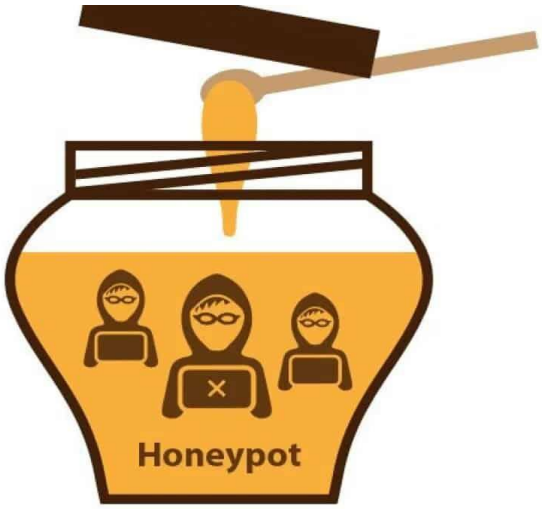
Guide d'installation d'un Honeypot.

11/05/2023



Étape 1 : Mise à jour des packages et installation des prérequis

Avant d'installer Apache2, nous devons nous assurer que les packages système sont à jour et que les prérequis sont installés. Exécutez les commandes suivantes dans votre terminal :

```
sudo apt install -y iptables-persistent tcpdump nmap iputils-ping python3 python-pip python3-psycopg2 lsof psmisc dnsutils
```

Étape 2 : Installation des dépendances Python

Apache2 peut avoir besoin de certaines dépendances Python pour fonctionner correctement. Exécutez la commande suivante pour installer les dépendances requises :

```
pip install scapy==2.4.4 netifaces==0.10.9 pyftpdlib==1.5.6 sqlalchemy==1.3.23 pyyaml==5.4.1 paramiko==2.7.1 impacket==0.9.22 twisted==20.3.0 psutil==5.8.0 requests==2.25.1 redis==3.5.3 mysql-connector-python==8.0.23 pygments==2.5.2
```

Étape 3 : Installation des packages supplémentaires

Certaines fonctionnalités d'Apache2 nécessitent des packages supplémentaires. Exécutez la commande suivante pour installer le package "requests[socks]":

```
pip install -U requests[socks]
```

De plus, assurez-vous d'installer la version spécifique de "rsa" en utilisant la commande suivante :

pip install -Iv rsa==4.0

Enfin, installez la bibliothèque "rdpy" en utilisant la commande suivante :

pip install rdpy==1.3.2

⚠ En **cas de conflit de dépendance :** ⚠
pip install --upgrade impacket==0.9.24 pip install --upgrade netifaces==0.11.0 pip install --upgrade psutil==5.9.0 pip install --upgrade requests==2.28.2 pip install --upgrade requests[socks]==2.28.2 pip install --upgrade scapy==2.4.5 pip install --upgrade twisted==21.7.0

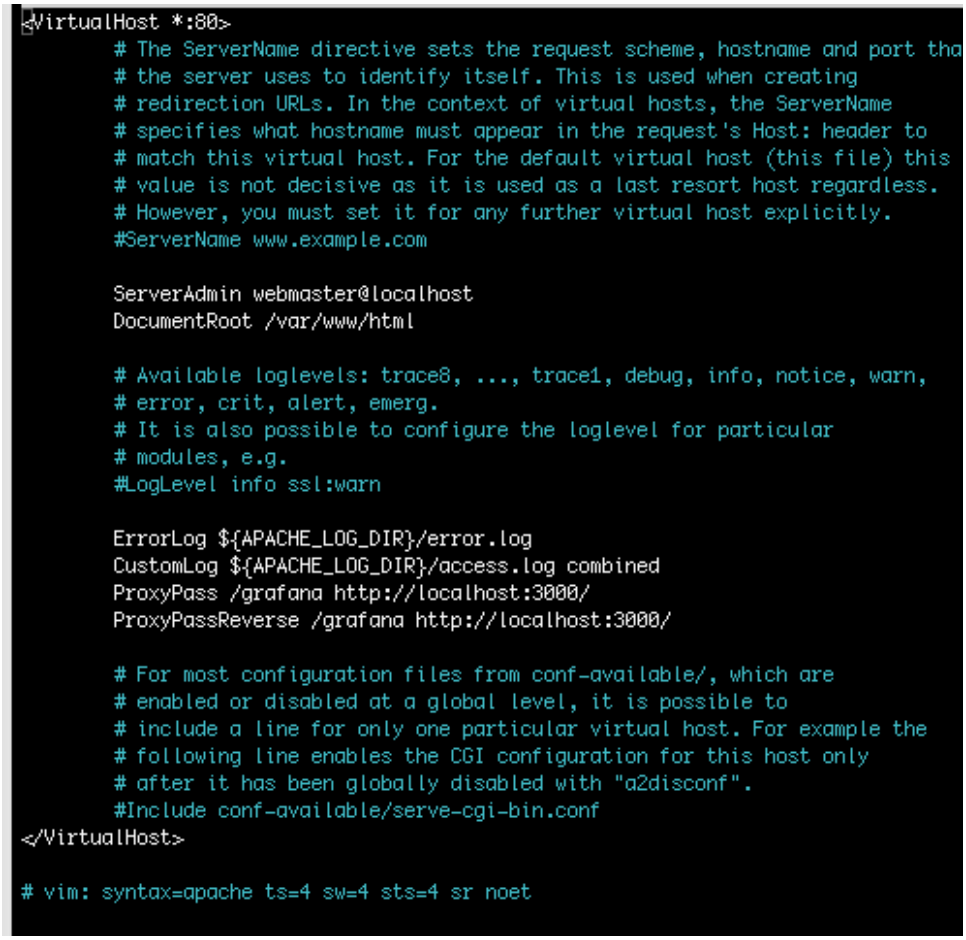
L’installation de Chameleon

Voici une présentation étape par étape de l'installation de Chameleon et de son lancement dans le répertoire /home/utilisateur, avec une brève explication pour chaque étape :

Étape	Commande	Explication
1	Ouvrez un terminal	Ouvrez l'application du terminal sur votre système d'exploitation.
2	cd /home/utilisateur	Naviguez vers le répertoire /home/utilisateur en utilisant la commande cd. Assurez-vous de remplacer "utilisateur" par votre nom.
3	git clone https://github.com/qeeqbox/chameleon.git	Clonez le dépôt Chameleon depuis GitHub pour obtenir les fichiers du projet.
4	cd chameleon	Accédez au répertoire "chameleon" créé après le clonage.
5	sudo chmod +x ./run.sh	Rendez le script run.sh exécutable en utilisant la commande chmod.
6	sudo ./run.sh test	Lancez Chameleon en mode test en exécutant le script run.sh avec la commande sudo. Il est possible de le lancer en mode déploiement en remplaçant “test” par “deploy”

Dans un autre shell Installation d'apache pour permettre de faire tourner Grafana

Vous rentre dans **sudo nano /etc/apache2/sites-enabled/000-default.conf**



Modifier le fichier

<VirtualHost *:80>

ServerName localhost

ProxyPass / http://localhost:3000/

ProxyPassReverse / http://localhost:3000/

Autres directives de configuration...

</VirtualHost>

Activer le proxy

sudo a2enmod proxy

sudo a2enmod proxy_http

sudo service apache2 restart

Ensuite dans un navigateur taper l'adresse ip de la machine virtuelle.

Ensuite cliquer sur la loupe à gauche du navigateur puis sélectionné chameleon maintenant vous prouvez voir l'interface qui va permettre d'afficher les logs de connexion.

Pour pouvoir afficher c'est connexion il nous faudra utiliser PostgreSQL sui est une base de données

sudo apt install postgresql

Création d'un nouvel utilisateur et une base de données postgresql

Se connecté en super utilisateur

sudo -u postgres psql

Créer la base de données

create database info_atack;

Créer l'utilisateur et l'c'est privilège

CREATE USER information WITH PASSWORD 'information_mdp';

GRANT ALL PRIVILEGES ON DATABASE info_atack TO information;

Création de table

Se déplacer dans la base de données info_atack

\c info_atack

Creation des tables

```
CREATE TABLE informations (
    protocole text,
```

```
    adresse_ip inet,
    utilisateur text,
    mot_de_passe text,
    pays text
);
```

Mémoire ssh honeypot

```
from warnings import filterwarnings
filterwarnings(action='ignore', module='.*paramiko.*')
filterwarnings(action='ignore', module='.*socket.*')

from paramiko import RSAKey, ServerInterface, Transport, OPEN_SUCCEEDED,
OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED
from socket import socket, AF_INET, SOCK_STREAM, SOL_SOCKET, SO_REUSEADDR, getfqdn
from _thread import start_new_thread
from io import StringIO
from random import choice
from subprocess import Popen
from os import path, getenv
import psycopg2
from honeypots.helper import check_if_server_is_running, close_port_wrapper, get_free_port,
kill_server_wrapper, server_arguments, set_local_vars, setup_logger
from uuid import uuid4
from contextlib import suppress
from re import compile as rcompile
from time import time
from threading import Event
from binascii import hexlify
```

```
class QSSHServer():
    def __init__(self, **kwargs):
        self.auto_disabled = None
        self.mocking_server = choice(['OpenSSH 7.5', 'OpenSSH 7.3', 'Serv-U SSH Server 15.1.1.108',
'OpenSSH 6.4'])
        self.process = None
        self.uuid = 'honeypotslogger' + '_' + __class__.__name__ + '_' + str(uuid4())[:8]
        self.config = kwargs.get('config', '')
        if self.config:
            self.logs = setup_logger(__class__.__name__, self.uuid, self.config)
            set_local_vars(self, self.config)
        else:
            self.logs = setup_logger(__class__.__name__, self.uuid, None)
        self.ip = kwargs.get('ip', None) or (hasattr(self, 'ip') and self.ip) or '0.0.0.0'
        self.port = (kwargs.get('port', None) and int(kwargs.get('port', None))) or (hasattr(self, 'port')
and self.port) or 22
        self.username = kwargs.get('username', None) or (hasattr(self, 'username') and self.username)
or 'test'
        self.password = kwargs.get('password', None) or (hasattr(self, 'password') and self.password) or
'test'
        self.options = kwargs.get('options', '') or (hasattr(self, 'options') and self.options) or
getenv('HONEYPOTS_OPTIONS', '') or ''
        self.ansi = rcompile(r'(?:\x1B[@-_]|[\x80-\x9F])[0-?]*[ -/]*[@-~]')

        # Connexion à la base de données PostgreSQL
        self.db_host = 'localhost'
        self.db_port = 5432
        self.db_name = 'votre_base_de_donnees'
        self.db_user = 'votre_utilisateur'
        self.db_password = 'votre_mot_de_passe'

    def generate_pub_pri_keys(self):
        with suppress(Exception):
```

```
        key = RSAKey.generate(2048)
        string_io = StringIO()
        key.write_private_key(string_io)
        return key.get_base64(), string_io.getvalue()
    return None, None

def save_auth_data(self, username, password):
    try:
        connection = psycopg2.connect(
            host=self.db_host,
            port=self.db_port,
            dbname=self.db_name,
            user=self.db_user,
            password=self.db_password
        )

        cursor = connection
```

/logkalanda:logkalanda@localhost:5430/connexion