Wirefish

Antoine SANSON - M1 Cybersécurité
Benjamin DELSOL - M1 Cybersécurité

04/05/2022

Yday 2021 - 2022 : projet fil rouge
Laboratoire sécurité des systèmes d'information

Table des matières

Introduction

Cas d’utilisation

État de l’art

Fonctionnement

Modules implémentés

Choix des technologies

Difficultés rencontrées

Axes d’amélioration

Démonstration

Conclusion

Introduction

Cas d’utilisation

Expérimentations dans un cadre personnel

Test d’intrusion

Man In The Middle

État de l’art

Wireshark

tcpdump

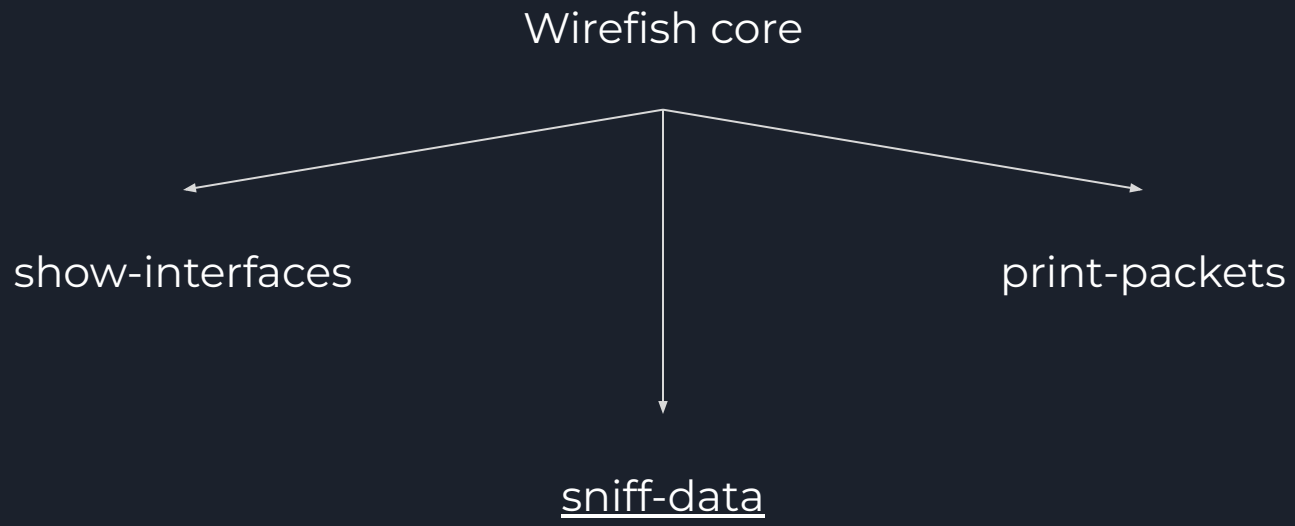
Affiche et enregistre le trafic

NetworkMiner

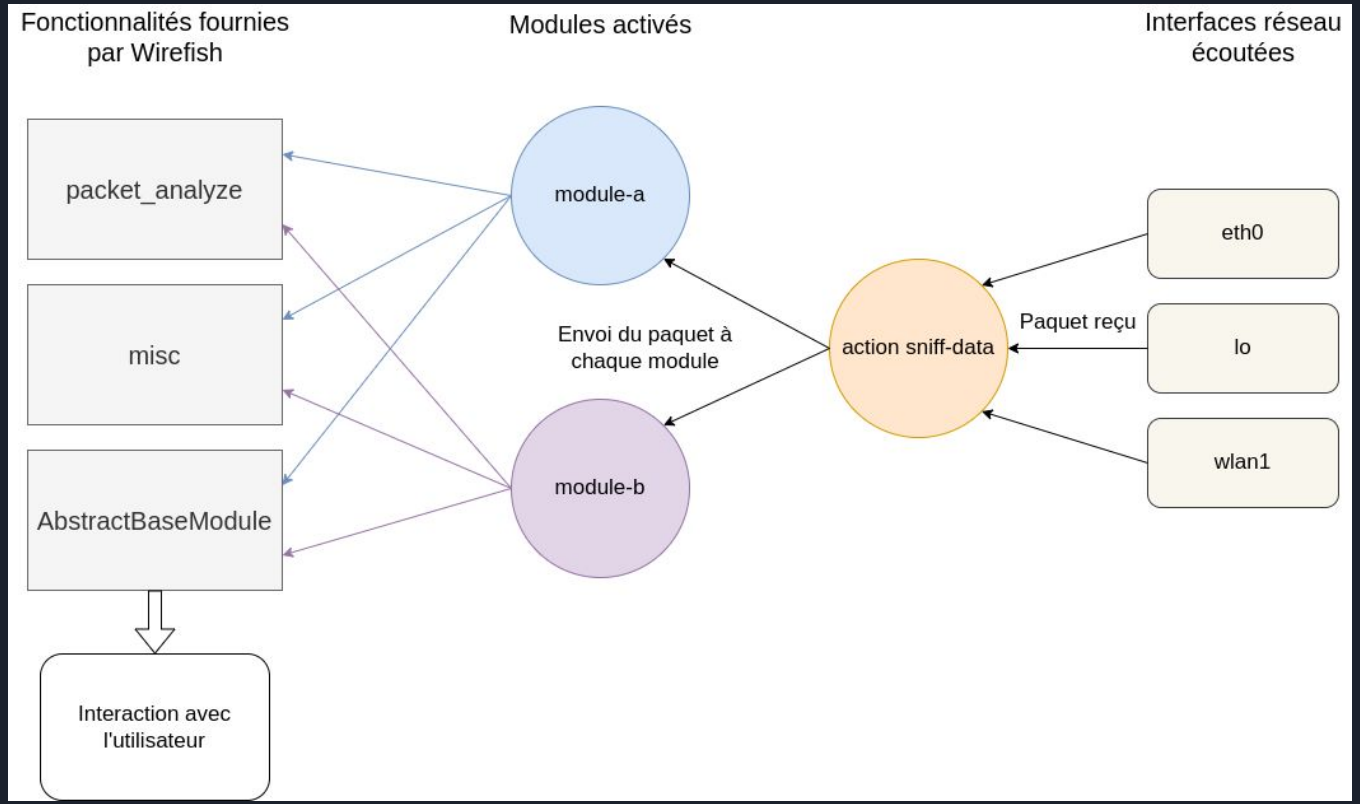
HTTP, FTP, SMB, SMTP

Fonctionnement

Fonctionnement : actions



Fonctionnement : sniff-data



Qu’est-ce qu’un module ?

Classe *Module*

on_receive_packet(packet) -> None

héritage

Classe abstraite *AbstractBaseModule*

_module_name

on_receive_packet(packet) -> None
set_module_name()
log(), log_message(), log_error()
write_binary_file()
to_str_safe()

Modules implémentés

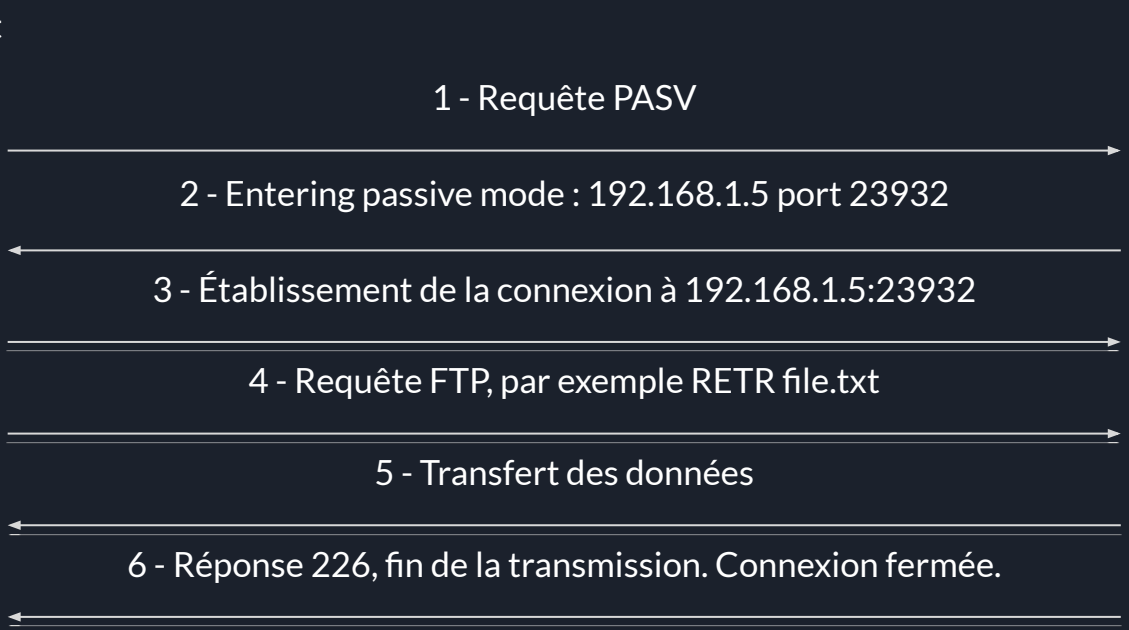
ftp.credentials

```
class Module(AbstractBaseModule):
    def on_receive_packet(self, packet: Packet) -> None:
        if packet.haslayer(TCP) and packet.haslayer(Raw):
            ftp_packet_parser = FTPPacketParser(packet=packet)
            if ftp_packet_parser.is_command("USER"):
                self.log_message("FTP username : %s " % (
                    self.to_str_safe(ftp_packet_parser.get_parameters())
                ))
            elif ftp_packet_parser.is_command("PASS"):
                self.log_message("FTP password : %s " % (
                    self.to_str_safe(ftp_packet_parser.get_parameters())
                ))
```

ftp.transfer_files

- Compréhension plus poussée du protocole FTP
- Interprétation des commandes FTP
- Mode passif ou actif pour le transfert de données

ftp.transfer_files



http.post_credentials

- Même principe que ftp.credentials
- Analyse de paquet isolé
- Récupération des tokens d’autorisation, des cookies, de données de formulaire

http.post_credentials

```
def handle_http_request(self, original_packet: Packet, request: HTTPRequest) -> None:
    request_url = self.to_str_safe(request.Host) + self.to_str_safe(request.Path)
    request_method = self.to_str_safe(request.Method)

    self.log_message("%s %s, Authorization header : %s, Cookie header : %s" % (
        request_method,
        request_url,
        self.to_str_safe(request.Authorization),
        self.to_str_safe(request.Cookie)
    ))
    if request_method == "POST" and original_packet.haslayer(Raw):
        self.log_message(original_packet[Raw].load)
```

http.download_files

- Copie de certains fichiers accédés, en fonction de leur type
- Corps des réponses HTTP
- Problème lorsque fichiers fragmentés

Choix des technologies

Python3

- Langage interprété
- Syntaxe claire et concise
- Écosystème
- Utilisé en cybersécurité

Scapy

- Traitement de paquet en Python
- Syntaxe simple
- Flexibilité avec fonctions manipulations de paquets
- À jour

Difficultés rencontrées

Scapy : problèmes de performance

➡ Résultats incohérents ou perte de données

<u>Solution partielle:</u>

Utilisation de **libpcap**

Contrainte supplémentaire dans l'environnement

Difficultés rencontrées

Perte, ou non-capture de paquets

Méthodologie

Limites d’un module

Axes d’amélioration

Stocker logs

Fichier de configuration

Optimisation

Lecture de fichiers
de capture réseau

Démonstration

Conclusion

Merci !