

Spis treści

Autorzy	23
Przedmowa	29
<i>Gynvael Coldwind</i>	
Wstęp	35
<i>Michał Sajdak</i>	
Dlaczego aplikacje webowe?	37
Co można powiedzieć o ich bezpieczeństwie?	37
Największy problem z bezpieczeństwem aplikacji WWW?	37
Czy istnieją metody sprawdzenia poziomu bezpieczeństwa naszych aplikacji?	38
Podziękowania	39
Prawne aspekty ofensywnego bezpieczeństwa IT	41
<i>Bohdan Widła</i>	
Wstęp	43
Prawo, czyli właściwie co?	44
Ochrona informacji i dostępu do systemów informatycznych	45
Przełamywanie lub omijanie zabezpieczeń	46
Dostęp do systemu bez omijania zabezpieczeń	48
Podśluch komputerowy	49
Ingerencja w dane lub w pracę systemu informatycznego	49
Naruszanie integralności danych	49
Zakłócanie przetwarzania danych lub pracy systemu	51
Sabotaż	52
Narzędzia	52
Próba ograniczenia karalności, czyli lex bug bounty	54
Stan wyższej konieczności	56
Kilka uwag końcowych	57
Podstawy protokołu HTTP	61
<i>Michał Sajdak</i>	
Wstęp	63
Podstawowa komunikacja HTTP	63
Metody HTTP	66
URL czy URI?	69
Nagłówki HTTP	71
Czy nagłówki są absolutnie wymagane?	74
Wartości przekazywane do aplikacji protokołem HTTP	75
Nagłówki żądania HTTP	75
URL	76
POST: application/x-www-form-urlencoded	78

POST: multipart/form-data	82
Ciasteczka	84
Podsumowanie	85

Burp Suite Community Edition – wprowadzenie do obsługi proxy HTTP 89

Marcin Piosek

Wstęp	91
Wyznaczamy cel	91
Czym jest Burp Suite?	91
Alternatywy i dlaczego Burp?	92
Pobranie i uruchomienie Burp Suite Community Edition	92
Konfiguracja środowiska pracy	94
Przystępujemy do pracy	96
Modyfikowanie zapytań HTTP	99
Przechwytywanie odpowiedzi	103
Repeater – powtórz to jeszcze raz	104
Intruder – automatyzacja i oszczędność czasu	106
Comparer – wskaż różnice	111
Decoder – radzimy sobie z „dziwnym” ciągiem znaków	112
Sequencer – analiza entropii i nie tylko	114
Połączenie nie jest bezpieczne – HTTPS	117
Dopasuj i zamień	119
Socksproxy – proxy w proxy	122
Rozwiązywanie nazw i brak uprawnień	123
Skróty klawiszowe	124
Wtyczki – jeszcze więcej możliwości	125
Gdy coś przebiega niezgodnie z planem	126
Podsumowanie	127

Protokół HTTP/2 – czyli szybciej, ale czy również bezpieczniej? 129

Michał Sajdak

Wstęp	131
Porównanie komunikacji z HTTP/1.1	132
Wykorzystanie protokołu TCP	132
Podstawy komunikacji HTTP/2	135
Bezpieczeństwo	138
Obowiązkowy TLS?	138
Złożoność protokołu	138
Znane podatności raz jeszcze	139
WAF/IDS	140
Co dalej?	140
Narzędzia	141
Podsumowanie	144

Nagłówki HTTP w kontekście bezpieczeństwa	147
<i>Artur Czyż</i>	
Wstęp	149
Jak możemy sprawdzić aktualne nagłówki dla konkretnej strony?	150
Wybrane nagłówki HTTP a ich wpływ na bezpieczeństwo	151
HTTP Strict-Transport-Security (HSTS)	151
Wdrożenie	153
Referrer-Policy	155
Wdrożenie	156
X-Content-Type-Options	157
Wdrożenie	157
Feature-Policy	159
Wdrożenie	160
X-Frame-Options	161
Wdrożenie	161
Nagłówki w służbie omijania zabezpieczeń i filtrów (m.in. WAF)	162
Podsumowanie	164
Chrome DevTools w służbie bezpieczeństwa aplikacji webowych	167
<i>Rafał 'bl4de' Janicki</i>	
Wstęp	169
Narzędzia	169
Analiza kodu HTML	170
Analiza mechanizmów przechowywania danych (Cookies, Storage)	176
Statyczna analiza kodu JavaScript	177
Debugger JavaScript	177
Wykonywanie kodu JavaScript z użyciem snippetów (Snippets)	181
Punkty wejścia i wykonania kodu (sources oraz execution sinks)	183
Podsumowanie	184
Bezpieczeństwo hasel statycznych	187
<i>Adrian 'vizzdoom' Michalczyk</i>	
Wstęp	189
Metody uwierzytelniania	189
Hasła statyczne	191
Hasła jednorazowe	191
Protokół wyzwanie-odpowiedź	193
Przechowywanie hasel statycznych	194
Funkcje skrótu i ich właściwości	195
Kryptograficzna funkcja skrótu Message Digest	197
Kryptograficzna funkcja skrótu SHA	198
Problem szybkości	199
Sól i pieprz	199
Key stretching	201
Wszystko w jednym – funkcje PBKDF	201
Studium przypadku – Battlefield Heroes	205
Studium przypadku – Dropbox	207

Ataki zdalne (online)	208
Ataki lokalne (offline)	209
Kompromis czasowo-pamięciowy i tęczowe tablice	212
Receptury	216
Polecane zasoby w sieci	218
Rekonesans aplikacji webowych (poszukiwanie celów)	221
<i>Michał Sajdak</i>	
Wstęp	223
Cel inwentaryzacji	224
Lokalizacja serwerów webowych na zadanym zakresie adresów IP	224
Poszukiwanie aktywne	224
nmap	224
Poszukiwanie pasywne	228
VirusTotal/PassiveTotal	228
Censys/Zoomeye/Shodan	229
Inne narzędzia	231
Rekonesans poddomen	232
Aktywne zapytania do DNS	232
Amass – metoda słownikowa i pozyskiwanie danych z zewnętrznych źródeł	232
DNS zone transfer	234
DNSSEC	235
Pasywne pozyskiwanie informacji o poddomenach	237
PassiveTotal	237
VirusTotal	238
Google/Bing/Yandex	239
Certificate Transparency logs	241
Projekt Sonar – historyczne wpisy Forward DNS oraz Reverse DNS	243
SecurityTrails	245
CSP	246
Źródła stron HTML/JS/CSS	246
Aplikacje mobilne	249
Domeny wirtualne	249
Automatyzacja rekonesansu	251
Domeny powiązane z bazowymi	254
Builtwith	254
ViewDNS.info	255
Podsumowanie	255
Ukryte katalogi i pliki jako źródło informacji o aplikacjach internetowych	257
<i>Rafał 'bl4de' Janicki</i>	
Wstęp	259
Systemy kontroli wersji	259
Podstawowe informacje o obiektach Git	259
Plik .gitignore	264
Subversion (SVN)	264
Katalogi i pliki konfiguracyjne środowisk programistycznych	267
JetBrains IDE – PhpStorm, WebStorm, PyCharm, IntelliJ IDEA	267

NetBeans IDE	271
Pliki konfiguracyjne narzędzi deweloperskich	271
Pliki konfiguracyjne specyficzne dla Node.js czy JavaScript: bower.json oraz package.json	271
Plik konfiguracyjny .gitlab-ci.yml (GitLab CI/CD)	274
Plik Ruby on Rails: database.yml	274
Plik macOS .DS_Store	275
Odkrywaj ukryte foldery oraz pliki z gotowym słownikiem dla swojego ulubionego narzędzia	276
Podsumowanie	276
Podatność Cross-Site Scripting (XSS)	279
<i>Michał Bentkowski</i>	
Wstęp	281
Czym jest XSS oraz typy podatności XSS	281
Skutki XSS	284
Konteksty XSS	287
Konteksty DOM XSS	291
Funkcje typu eval	291
Funkcje przyjmujące kod HTML	292
Funkcje przyjmujące adres URL	292
XSS nie tylko w HTML	294
XSS a dopuszczanie fragmentów HTML	295
Ochrona przed XSS	296
Enkodowanie danych	296
Systemy szablonów z automatycznym enkodowaniem	297
Ochrona przed DOM XSS	298
Filtrowanie HTML	298
Upload plików	299
Content-Security-Policy	300
Filtry anti-XSS w przeglądarkach	300
XSS a popularne biblioteki JS	301
Dynamiczne budowanie szablonów	302
Bezpośrednie podstawianie HTML	303
Pułapka kontekstu URL-owego	303
Frameworki a DOM XSS	304
Podsumowanie	304
Content Security Policy (CSP)	307
<i>Michał Bentkowski</i>	
Czym jest CSP i przed czym chroni	309
Składnia CSP	310
Dyrektywy CSP	311
Dyrektywy *-src	311
Dyrektywa script-src	313
Dyrektywa base-uri	318
Dyrektywy block-all-mixed-content i upgrade-insecure-requests	319
Dyrektywa form-action	320

Dyrektywa frame-ancestors	321
Dyrektywa plugin-types	322
Dyrektywa report-uri	322
Dyrektywa sandbox	323
Raportowanie	323
Sposoby obejścia CSP	325
Obejście przez JSONP	325
Obejście przez frameworki JS	326
Kiedy warto, a kiedy nie warto stosować CSP?	328
Przykładowe polityki CSP	329
Podsumowanie	330

Same-Origin Policy i Cross-Origin Resource Sharing (CORS)

Mateusz Niezabitowski

Wstęp	335
Same-Origin Policy (SOP)	335
Przykład 1	336
Przykład 2	336
Przykład 3	337
Cross-Origin Resource Sharing (CORS)	338
Przykład podobny do trzeciego ze wstępu	339
Przykład podobny do drugiego ze wstępu	339
Obiekty XMLHttpRequest2	340
Model pierwszy – zapytania proste (Simple Requests)	341
Model drugi – zapytania nie-takie-proste (Not-So-Simple Requests)	345
Przesyłanie danych uwierzytelniających w CORS	350
Implementacja mechanizmu CORS po stronie serwera	351
Wady CORS	352
Alternatywy dla CORS	353
JSONP	353
postMessage	354
Serwer proxy	355
WebSockets	356
Flash i crossdomain.xml	356
Sposoby obejścia Same-Origin Policy	357
Obejścia CORS	357
Zbyt szerokie uprawnienia: * (gwiazdka) w odpowiedzi	357
Zbyt szerokie uprawnienia: „odbijanie” originu	358
Błędy implementacji	359
„null” origin	360
Nadmierne zaufanie do stron trzecich	361
CORS i Cache Poisoning	361
Inne przykłady obejścia SOP	363
Przykład 1	363
Przykład 2	363
Przykład 3	363
Przykład 4	364
Przykład 5	365
Obejścia dla deweloperów	367

Podsumowanie	367
Polecane zasoby w sieci	368
Podatność Cross-Site Request Forgery (CSRF)	371
<i>Michał Sajdak</i>	
Wstęp	373
Przykład 1. CSRF realizowany w tej samej domenie.	
Nieautoryzowane utworzenie nowego konta administracyjnego, metoda GET	374
Przykład 2. CSRF realizowany pomiędzy różnymi domenami.	
Nieautoryzowane usunięcie konta administratora, metoda GET	375
CSRF a Same-Origin Policy	375
Przykład 3. CSRF realizowany pomiędzy różnymi domenami.	
Bankowość elektroniczna, metoda POST	376
CSRF a inne niż GET/POST metody HTTP	377
Przykład 4. CSRF w połączeniu z innymi podatnościami – urządzenia sieciowe	378
Przykład 5. Podatność wieloetapowa – przejście dostępu do systemu WordPress	379
Metody ochrony przed CSRF	380
Losowe tokeny	380
SameSite	382
Ekran logowania	382
Nowe podatności wprowadzone przez ochronę przeciwko CSRF	383
Podsumowanie	383
Podatność Server-Side Template Injection (SSTI)	385
<i>Mateusz Niezabitowski</i>	
Wstęp	387
Silniki szablonów	387
Server-Side Template Injections – Velocity	390
Teoria, metodyka, narzędzia	397
Identyfikacja podatności	397
Identyfikacja silnika	402
Eksploracja	403
Narzędzia i przykład zastosowania – Freemarker	404
Zapobieganie i obrona	417
Rezygnacja z szablonów (przynajmniej częściowo)	417
Użycie bezpiecznych silników	418
Sandboxing	418
Hardening	425
Podsumowanie	425
Polecane zasoby w sieci	425
Podatność Server-Side Request Forgery (SSRF)	427
<i>Michał Sajdak</i>	
Wstęp	429
Przykłady	430
Możliwe skutki wykorzystania podatności	431

Częste miejsca występowania podatności SSRF	432
Podstawy	432
Pliki XML	432
XXE	432
Document type definition (DTD)	433
XInclude	433
SVG/XLink	433
XSLT	434
Formaty pakietów biurowych	434
Inne formaty plików	435
Dowolne formaty	435
MP4	436
Biblioteki	436
Mechanizm uploadu	437
Inne miejsca	437
Protokoły inne niż HTTP wykorzystywane w SSRF	438
Wstęp	438
HTTPS	439
PHAR	440
Gopher	441
Inne protokoły	442
Częste błędy w filtrach anti-SSRF	442
Filtry blacklist	442
Filtry whitelist	443
Metody ochrony	445
Podsumowanie	446

Podatność SQL Injection

Michał Bentkowski

Wstęp	451
Czym jest SQL Injection	451
Sposoby wykorzystania SQL Injection	453
UNION-based	453
ERROR-based	457
BLIND (content based)	458
BLIND (time based)	461
Stacked queries	462
Skutki wykorzystania SQL Injection	462
Wydobycie dowolnych danych z bazy	462
Omijanie ekranu logowania	463
Modyfikacja/usuwanie danych	464
Przykład 1. Zmiana hasła administratora: UPDATE	464
Przykład 2. Ścieżka dostępu	464
Przykład 3. Wykonanie kodu PHP	465
Odczyt plików z dysku	465
Zapis plików na dysku	466
Wykonywanie poleceń systemu operacyjnego	466
Jak szukać SQL Injection?	468
Second-order SQL Injection	472

Metody ochrony przed SQL Injection	473
Zapytania parametryzowane	473
Walidacja typów danych	474
Stosowanie systemów klasy ORM	474
Hardening bazy danych	475
Podsumowanie	475
 Podatność Path Traversal	477
<i>Marcin Piosek</i>	
Wstęp	479
Logika podatności	479
Zagrożenia	479
Ujawnienie nadmiarowych informacji	480
Ujawnienie plików konfiguracyjnych	480
Zdalne wykonanie kodu	480
Szersze spojrzenie na problem	480
Przykłady podatności	481
GlassFish Server	481
ColoradoFTP 1.3	482
Testowanie	482
Automatyzacja	482
Omijanie filtrów	484
Ochrona	484
Blokowanie ataku poprzez podmianę tekstu	484
Modelowanie zagrożeń	485
Podsumowanie	485
 Code Injection i Command Injection –	
przegląd wektorów ataku w aplikacjach webowych	487
<i>Marcin Piosek</i>	
Wstęp	489
Czym jest Code Injection oraz Command Injection?	489
Wektory ataków	490
Formalność – Eval	490
Klasyka: Local File Inclusion i Remote File Inclusion	491
Podatności w mechanizmie wgrzywania plików	493
Uruchamianie zewnętrznego oprogramowania – Command Injection	494
Panele administracyjne	496
Cross-Site Scripting a Code Injection	499
Tryb debug a serwer produkcyjny	499
Od SQL Injection do RCE	500
XSLT	501
WebDAV	502
Podatności w bibliotekach	503
Przegląd podatności	503
Drobne błędy	506
Czy to już wszystko?	506
Skutki	507

Wykradanie danych	507
Eskalacja	507
Jakby tego było mało – webshelle i tylne furtki	508
Podsumowanie	508

Uwierzytelnianie, zarządzanie sesją, autoryzacja 511

Marcin Piosek

Wstęp	513
Teoria i podstawowe pojęcia	513
Błędne pojęcia	513
Identyfikacja	513
Uwierzytelnianie	514
Zarządzanie sesją	514
Autoryzacja	514
Błędy bezpieczeństwa	515
Rodzaje mechanizmów uwierzytelniania	515
Sposób klasyczny (zapytanie oraz ciasteczka HTTP)	515
HTTP Basic Authentication	516
OpenID Connect	516
Klucze API	517
Uwierzytelnianie certyfikatem	517
Kerberos oraz NTLM	517
Uwierzytelnianie	518
Nie ma HTTPS, nie ma poświadczeń	518
Brak uwierzytelnienia	521
Po co wywierać, skoro można obejść – omijanie uwierzytelnienia	522
Poświadczenia podane na tacy	523
Praca po stronie serwera	526
Enumeracja użytkowników	527
Automatyzacja ataku, czyli ataki brute-force	528
Zabezpieczenia tak dobre, że aż złe – jak skutecznie bronić się przed atakami siłowymi	531
Jak nie drzwiami, to oknem	532
Reset hasła	533
Niebezpieczne pytania bezpieczeństwa	536
Podaj hasło jeszcze raz – krytyczne akcje	536
Higiena przechowywania haseł	537
Polityka bezpieczeństwa haseł	538
Logowanie zdarzeń	538
Zarządzanie sesją	539
Wymyślanie koła na nowo	540
Regenerowanie sesji	541
Session Fixation	541
Obsługa równoległych sesji	542
Strzec jak oka w głowie	544
Odpowiednia złożoność	546
Odpowiedni czas życia	547
Unieważnianie sesji	547
ID sesji jako fingerprint	548
ID sesji jako zagrożenie	548

Kłopotliwa funkcja „zapamiętaj mnie”	548
Autoryzacja	550
Brak autoryzacji oraz autoryzacja na warstwie interfejsu	551
Podjmowanie decyzji i eskalacja uprawnień	552
Problem globalnych identyfikatorów	553
Centralizacja	555
Rozliczalność oraz niezaprzeczalność	555
Krytyczne operacje	556
Wybór modelu autoryzacji	556
Zasada najmniejszego uprzywilejowania	557
Co można zrobić lepiej	558
Uwierzytelnianie dwuskładnikowe	558
Poszerzanie wiedzy	559
Podsumowanie	560
Polecane zasoby w sieci	560
Pułapki w przetwarzaniu plików XML	563
<i>Michał Bentkowski</i>	
Wstęp	565
Podstawy XML – encje i encje parametryczne	565
Billion laughs	567
Quadratic blowup	569
XXE (XML eXternal Entity)	570
Inne podatności XML	573
Podsumowanie	574
Bezpieczeństwo API REST	577
<i>Michał Sajdak</i>	
Wstęp	579
Czym jest API REST?	579
Metody HTTP	579
Brak ścisłej formalizacji użycia metod	579
Nadpisywanie metod	580
Rekonesans API	583
Przykład	583
Dokumentacja	584
Wykonanie metody API na wiele różnych sposobów	587
Frameworki	587
Struts REST Plugin	588
Spring Data REST	588
Spring OAuth	589
RESTEasy	589
Jackson Databind	589
Tryb debug	590
Jakie formaty danych akceptuje nasze API?	591
JSON	591
XML	593
YAML	594

Bezpośrednia deserializacja	595
Jakiego formatu danych oczekuję w odpowiedzi?	595
Problemy z kluczami API	597
Bezpieczeństwo webhooks	599
Uwierzytelnienie i autoryzacja	600
Problem z uwierzytelnieniem, a następnie z autoryzacją	600
Reset hasła	601
Dostęp do panelu administracyjnego	601
Kradzież środków z jednego z największych banków w Indiach	602
Dostęp do wewnętrznego API	602
Podsumowanie	603

Niebezpieczeństwa JSON Web Token (JWT)

Michał Sajdak

Wstęp	609
Definicja	609
Kolejny przykład JWT i pierwsze problemy bezpieczeństwa	610
Kolec pierwszy: nadmierna komplikacja	612
Kolec drugi: none	613
Kolec trzeci: łamanie hasła do HMAC	614
Kolec czwarty: gdzie algorytmów sześć, tam nie ma bezpieczeństwa	616
Kolec piąty: choć może należałoby mu się pierwszeństwo	617
Kolec szósty: czy szyfrowanie JWT może w ogóle działać?	617
Kolec siódmy: dekodowanie/weryfikacja – co za różnica?	618
Kolec ósmy: przechwycenie dowolnego tokena = przejęcie dostępu do API?	618
Kolec dziewiąty: replay JWT	619
Kolec dziesiąty: ataki czasowe na podpis	620
Kolec jedenasty: mnogość bibliotek	620
Alternatywa do JWT?	620
Czy JWT może być bezpieczne?	621
Podsumowanie	622
Polecane zasoby w sieci	623

Zalety i wady OAuth 2.0 z perspektywy bezpieczeństwa

Marcin Piosek

Wstęp	627
Czym jest OAuth 2.0?	627
Wykorzystywana terminologia	628
Zasada działania OAuth	629
Korzyści wynikające ze stosowania OAuth	632
Krok w tył – czym OAuth nie jest	632
Pozostałe metody pozyskiwania tokena	633
Implicit Grant	633
Client Credentials	634
Resource Owner Credentials	635
Refresh token	635
Jaką metodę pozyskiwania tokena wybrać?	636
Co może pójść nie tak	636

Brak szyfrowanego kanału komunikacji	637
Serwer autoryzujący	637
Problematyczne przekierowania	638
Stary znajomy – CSRF	640
Granulacja uprawnień	641
Klient	643
Tokeny oraz kody dostępu	644
Consent screen	646
OAuth i phishing	647
Biblioteki i gotowe rozwiązania	647
Aplikacje mobilne	648
Cookies	648
Brak izolacji	648
Krok w tył	648
Złe nawyki	648
Własne URI	649
PKCE	649
Alternatywa dla WebViews oraz Custom URI	650
Aplikacje natywne	650
Różnice w stosunku do OAuth 1.0 oraz kontrowersje	650
Modelowanie zagrożeń	651
Podsumowanie	652
Bezpieczeństwo protokołu WebSocket	655
<i>Marcin Piosek</i>	
Wstęp	657
Co to jest i jak działa protokół WebSocket?	657
Prosty klient	660
Zagrożenia	663
Same-Origin Policy	663
Niepoprawne zarządzanie uwierzytelnieniem oraz sesją	664
Ominięcie autoryzacji	664
Wstrzyknięcia i niepoprawna obsługa danych	664
Wyczerpanie zasobów serwera	664
Tunelowanie ruchu	665
Szyfrowany kanał komunikacji	665
Gotowe rozwiązania	666
Testowanie	666
Modelowanie zagrożeń	668
Podsumowanie	668
Polecane zasoby w sieci	668
Flaga SameSite – jak działa i przed czym zapewnia ochronę?	671
<i>Marcin Piosek</i>	
Wstęp	673
Podstawy – przeglądarki, ciasteczka i sesja	673
Nadużycie	674
SameSite na ratunek	674

Polityka Lax – nawigacja „top-level” oraz bezpieczne metody HTTP	679
Bilans zysków i strat	681
Podsumowanie: lek na całe zło?	682
Niebezpieczeństwa deserializacji w PHP	685
<i>Michał Bentkowski</i>	
Wstęp	687
Jak wygląda serializacja w PHP?	687
Podatność PHP Object Injection	689
Trening – wykorzystanie Object Injection w Guzzle	691
Praktyczne wykorzystanie Object Injection	695
Ochrona przed podatnością	697
Podsumowanie	698
Niebezpieczeństwa deserializacji w Pythonie (moduł pickle)	701
<i>Michał Bentkowski</i>	
Wstęp	703
Jak działa moduł pickle?	703
Złośliwe użycie pickle	705
Sposoby ochrony	706
Podsumowanie	708
Niebezpieczeństwa deserializacji w .NET	711
<i>Grzegorz Trawiński</i>	
Wstęp	713
Środowisko programistyczne	714
Json.Net	715
XmlSerializer	726
Case Studies	732
Breeze (CVE-2017-9424)	732
DotNetNuke Platform (CVE-2017-9822)	732
Microsoft SharePoint (CVE-2019-0604)	734
Niebezpieczeństwa deserializacji w Javie	737
<i>Mateusz Niezabitowski</i>	
Wstęp	739
Podstawy – teoria	739
Mechanizm serializacji/deserializacji	740
Niebezpieczna deserializacja	740
Automatyczne wykonanie metod	740
Klasy „interesujące” i „dostępne”	740
Program deserializuje dane od użytkownika	740
Podatność deserializacji w języku Java	740
Apache commons-collections gadget chain	741
Analiza	743
Podatność deserializacji w języku Java – praktyka	748

Case Study: prosta aplikacja	749
Natywna serializacja a DoS	751
Rekurencyjne zbiory (java.util.Set)	752
Analiza	752
Modyfikacja zserializowanych danych i błąd przepełnienia pamięci	753
Analiza	754
Inne formaty serializacji	755
Biblioteka XStream	755
Jak się zabezpieczyć przed deserializacją niezaufanych danych?	760
Rozwiązanie #0: obfuskacja	761
Rozwiązanie #1: brak serializacji	761
Eliminacja serializacji natywnej	762
Rozwiązanie #2: blokowanie gadżetów	764
Blacklisting i whitelisting	764
Java Serialization Filter	765
Rozwiązanie #3: kryptografia	766
Analiza	769
Rozwiązanie bonusowe: monitoring	771
Monitorowanie przesyłania zserializowanych obiektów	771
Monitorowanie wyjątków	772
Podsumowanie	772
Wprowadzenie do programów bug bounty	775
<i>Jarosław Kamiński</i>	
Wstęp	777
Programy bug bounty	777
Rys historyczny	778
Jak się do tego zabrać?	780
Budowanie warsztatu	780
Udział w konkursach i programy bughunterskie	781
W jaki sposób zgłosić błąd?	784
Dokumentacja programu	784
Zasady punktacji	786
Zgłoszenie znalezionej podatności	786
Modelowy raport	787
Cykl życia błędu, czyli co się dzieje po przesłaniu raportu	788
Jak wybrać program?	789
Na miarę możliwości	789
Rozpoznana technologia	789
Wnikliwa analiza zasad obowiązujących w programie	790
Zakres testu	792
Prawa i obowiązki programów	793
Profesjonalizacja	794
Podsumowanie	795