

# Spis treści

<b>Autorzy</b> .....	21
<b>Wstęp</b> .....	27
Wstęp .....	29
Dlaczego aplikacje webowe? .....	29
Co można powiedzieć o ich bezpieczeństwie? .....	29
Największy problem z bezpieczeństwem aplikacji WWW? .....	29
Czy istnieją metody sprawdzenia poziomu bezpieczeństwa naszych aplikacji? .....	30
<b>Przedmowa</b> .....	33
Przedmowa .....	35
<b>Prawne aspekty ofensywnego bezpieczeństwa IT</b> .....	39
Wstęp .....	41
Prawo, czyli właściwie co? .....	42
Ochrona informacji i dostępu do systemów informatycznych .....	43
Przełamywanie lub omijanie zabezpieczeń .....	44
Dostęp do systemu bez omijania zabezpieczeń .....	46
Podsłuch komputerowy .....	47
Ingerencja w dane lub w pracę systemu informatycznego .....	47
Naruszanie integralności danych .....	47
Zakłócanie przetwarzania danych lub pracy systemu .....	49
Sabotaż .....	50
Narzędzia .....	50
Próba ograniczenia karalności, czyli lex bug bounty .....	52
Stan wyższej konieczności .....	54
Kilka uwag końcowych .....	55
<b>Podstawy protokołu HTTP</b> .....	59
Wstęp .....	61
Podstawowa komunikacja HTTP .....	61
Metody HTTP .....	64
URL czy URI? .....	67
Nagłówki HTTP .....	69
Czy nagłówki są absolutnie wymagane? .....	72
Wartości przekazywane do aplikacji protokołem HTTP .....	73
Nagłówki żądania HTTP .....	73
URL .....	74
POST: application/x-www-form-urlencoded .....	76
POST: multipart/form-data .....	80
Ciasteczka .....	82
Podsumowanie .....	83

<b>Burp Suite Community Edition – wprowadzenie do obsługi proxy HTTP</b>	<b>87</b>
Wstęp	89
Wyznaczamy cel	89
Czym jest Burp Suite?	89
Alternatywy i dlaczego Burp?	90
Pobranie i uruchomienie Burp Suite Community Edition	90
Konfiguracja środowiska pracy	93
Przystępujemy do pracy	94
Modyfikowanie zapytań HTTP	97
Przechwytywanie odpowiedzi	101
Repeater – powtórz to jeszcze raz	102
Intruder – automatyzacja i oszczędność czasu	104
Comparer – wskaź różnice	109
Decoder – radzimy sobie z „dziwnym” ciągiem znaków	110
Sequencer – analiza entropii i nie tylko	112
Połączenie nie jest bezpieczne – HTTPS	115
Dopasuj i zamień	117
Socksproxy – proxy w proxy	120
Rozwiązywanie nazw i brak uprawnień	121
Skróty klawiszowe	122
Wtyczki – jeszcze więcej możliwości!	123
Gdy coś przebiega niezgodnie z planem	124
Podsumowanie	125
<b>Protokół HTTP/2 – czyli szybciej, ale czy również bezpieczniej?</b>	<b>127</b>
Wstęp	129
Porównanie komunikacji z HTTP/1.1	130
Wykorzystanie protokołu TCP	130
Podstawy komunikacji HTTP/2	133
Bezpieczeństwo	136
Obowiązkowy TLS?	136
Złożoność protokołu	136
Znane podatności raz jeszcze	137
WAF/IDS	138
Co dalej?	138
Narzędzia	139
Podsumowanie	142
<b>Nagłówki HTTP w kontekście bezpieczeństwa</b>	<b>145</b>
Wstęp	147
Jak możemy sprawdzić aktualne nagłówki dla konkretnej strony?	148
Wybrane nagłówki HTTP a ich wpływ na bezpieczeństwo	149
HTTP Strict-Transport-Security (HSTS)	149
Wdrożenie	151
Referrer-Policy	153
Wdrożenie	154
X-Content-Type-Options	155

Wdrożenie .....	155
Feature-Policy .....	157
Wdrożenie .....	158
X-Frame-Options .....	159
Wdrożenie .....	159
Nagłówki w służbie omijania zabezpieczeń i filtrów (m.in. WAF) .....	160
Podsumowanie .....	162
<b>Chrome DevTools w służbie bezpieczeństwa aplikacji webowych .....</b>	<b>165</b>
Wstęp .....	167
Narzędzia .....	167
Analiza kodu HTML .....	168
Analiza mechanizmów przechowywania danych (Cookies, Storage) .....	174
Statyczna analiza kodu JavaScript .....	175
Debugger JavaScript .....	175
Wykonywanie kodu JavaScript z użyciem snippetów (Snippets) .....	179
Punkty wejścia i wykonania kodu (sources oraz execution sinks) .....	181
Podsumowanie .....	182
<b>Bezpieczeństwo haseł statycznych .....</b>	<b>185</b>
Wstęp .....	187
Metody uwierzytelniania .....	187
Hasła statyczne .....	188
Hasła jednorazowe .....	189
Protokół wyzwanie-odpowiedź .....	191
Przechowywanie haseł statycznych .....	192
Funkcje skrótu i ich właściwości .....	193
Kryptograficzna funkcja skrótu Message Digest .....	194
Kryptograficzna funkcja skrótu SHA .....	195
Problem szybkości .....	197
Sól i pieprz .....	197
Key stretching .....	199
Wszystko w jednym – funkcje PBKDF .....	199
Studium przypadku – Battlefield Heroes .....	203
Studium przypadku – Dropbox .....	205
Ataki zdalne (online) .....	206
Ataki lokalne (offline) .....	207
Kompromis czasowo-pamięciowy i tęczowe tablice .....	210
Receptury .....	214
Polecane zasoby w sieci .....	216
<b>Rekonesans aplikacji webowych (poszukiwanie celów) .....</b>	<b>219</b>
Wstęp .....	221
Cel inwentaryzacji .....	222
Lokalizacja serwerów webowych na zadanym zakresie adresów IP .....	222
Poszukiwanie aktywne .....	222
nmap .....	222
Poszukiwanie pasywne .....	226

VirusTotal/PassiveTotal .....	226
Censys/Zoomeye/Shodan .....	227
Inne narzędzia .....	229
Rekonesans poddomen .....	230
Aktywne zapytania do DNS .....	230
Amass – metoda słownikowa i pozyskiwanie danych z zewnętrznych źródeł .....	230
DNS zone transfer .....	232
DNSSEC .....	233
Pasywne pozyskiwanie informacji o poddomenach .....	235
PassiveTotal .....	235
VirusTotal .....	236
Google/Bing/Yandex .....	237
Certificate Transparency logs .....	239
Projekt Sonar – historyczne wpisy Forward DNS oraz Reverse DNS .....	241
SecurityTrails .....	243
CSP .....	244
Źródła stron HTML/ JS/CSS .....	244
Aplikacje mobilne .....	247
Domeny wirtualne .....	247
Automatyzacja rekonesansu .....	249
Domeny powiązane z bazowymi .....	252
Builtwith .....	252
ViewDNS.info .....	253
Podsumowanie .....	253
<b><u>Ukryte katalogi i pliki jako źródło informacji o aplikacjach internetowych</u></b> .....	<b>255</b>
Wstęp .....	257
Systemy kontroli wersji .....	257
Podstawowe informacje o obiektach Git .....	257
Plik .gitignore .....	262
Subversion (SVN) .....	262
Katalogi i pliki konfiguracyjne środowisk programistycznych .....	265
JetBrains IDE – PHPStorm, WebStorm, PyCharm, IntelliJ IDEA .....	265
NetBeans IDE .....	269
Pliki konfiguracyjne narzędzi deweloperskich .....	269
Pliki konfiguracyjne specyficzne dla NodeJS czy JavaScript: bower.json oraz package.json .....	269
Plik konfiguracyjny .gitlab-ci.yml (GitLab CI/CD) .....	272
Plik Ruby on Rails: database.yml .....	272
Plik macOS .DS_Store .....	273
Odkrywaj ukryte foldery oraz pliki z gotowym słownikiem dla swojego ulubionego narzędzia .....	274
Podsumowanie .....	274
<b><u>Podatność Cross-Site Scripting (XSS)</u></b> .....	<b>277</b>
Wstęp .....	279
Czym jest XSS oraz typy XSS-ów .....	279
Skutki XSS-ów .....	282
Konteksty XSS .....	285

Konteksty DOM XSS .....	289
Funkcje typu eval .....	289
Funkcje przyjmujące kod HTML .....	290
Funkcje przyjmujące adres URL .....	290
XSS nie tylko w HTML-u .....	292
XSS a dopuszczanie fragmentów HTML-a .....	293
Ochrona przed XSS .....	294
Enkodowanie danych .....	294
Systemy szablonów z automatycznym enkodowaniem .....	295
Ochrona przed DOM XSS .....	296
Filtrowanie HTML .....	296
Upload plików .....	297
Content-Security-Policy .....	298
Filtry anti-XSS w przeglądarkach .....	298
XSS-y a popularne biblioteki JS .....	299
Dynamiczne budowanie szablonów .....	300
Bezpośrednie podstawianie HTML .....	301
Pułapka kontekstu URL-owego .....	301
Frameworki a DOM XSS .....	302
Podsumowanie .....	302
<b><u>Content Security Policy (CSP)</u></b> .....	<b>305</b>
Czym jest CSP i przed czym chroni .....	307
Składnia CSP .....	308
Dyrektywy CSP .....	309
Dyrektywy *-src .....	309
Dyrektywa script-src .....	311
Dyrektywa base-uri .....	316
Dyrektywy block-all-mixed-content i upgrade-insecure-requests .....	317
Dyrektywa form-action .....	318
Dyrektywa frame-ancestors .....	319
Dyrektywa plugin-types .....	320
Dyrektywa report-uri .....	320
Dyrektywa sandbox .....	321
Raportowanie .....	321
Sposoby obejścia CSP .....	323
Obejście przez JSONP .....	323
Obejście przez frameworki JS .....	324
Kiedy warto, a kiedy nie warto stosować CSP? .....	326
Przykładowe polityki CSP .....	326
Podsumowanie .....	327
<b><u>Same-Origin Policy i Cross-Origin Resource Sharing (CORS)</u></b> .....	<b>331</b>
Wstęp .....	333
Same-Origin Policy (SOP) .....	333
Przykład 1 .....	334
Przykład 2 .....	334
Przykład 3 .....	335
Cross-Origin Resource Sharing (CORS) .....	336

Przykład podobny do 3. ze wstępu .....	337
Przykład podobny do 2. ze wstępu .....	337
Obiekty XMLHttpRequest2 .....	338
Model pierwszy – zapytania proste (Simple Requests) .....	339
Model drugi – zapytania nie-takie-proste (Not-So-Simple Requests) .....	343
Przesyłanie danych uwierzytelniających w CORS .....	349
Implementacja mechanizmu CORS po stronie serwera .....	350
Wady CORS .....	351
Alternatywy dla CORS .....	351
JSONP .....	351
postMessage .....	352
Server Proxy .....	353
WebSockets .....	354
Flash i crossdomain.xml .....	354
Sposoby obejścia Same-Origin Policy .....	355
„Obejścia” CORS .....	355
Zbyt szerokie uprawnienia: * (gwiazdka) w odpowiedzi .....	355
Zbyt szerokie uprawnienia: „odbijanie” originu .....	356
Błędy implementacji .....	357
„null” origin .....	358
Nadmierne zaufanie do stron trzecich .....	359
CORS i Cache Poisoning .....	359
Inne przykłady obejścia SOP .....	361
Przykład 1 .....	361
Przykład 2 .....	361
Przykład 3 .....	362
Przykład 4 .....	363
Przykład 5 .....	363
Obejścia dla deweloperów .....	365
Podsumowanie .....	366
Polecane zasoby w sieci .....	366
<b>Podatność Cross-Site Request Forgery (CSRF) .....</b>	<b>369</b>
Wstęp .....	371
Przykład 1. CSRF realizowany w tej samej domenie .....	372
Nieautoryzowane utworzenie nowego konta administracyjnego, metoda GET .....	372
Przykład 2. CSRF realizowany pomiędzy różnymi domenami .....	373
Nieautoryzowane usunięcie konta administratora, metoda GET .....	373
CSRF a Same-Origin Policy .....	373
Przykład 3. CSRF realizowany pomiędzy różnymi domenami .....	374
Bankowość elektroniczna, metoda POST .....	374
CSRF vs inne niż GET / POST metody HTTP .....	375
Przykład 4. CSRF w połączeniu z innymi podatnościami – urządzenia sieciowe .....	376
Przykład 5. Podatność wieloetapowa – przejście dostępu do systemu Wordpress .....	377
Metody ochrony przed CSRF .....	378
Losowe tokeny .....	378
SameSite .....	380
Ekran logowania .....	380
Nowe podatności wprowadzone przez ochronę przeciwko CSRF .....	381

Podsumowanie .....	381
<b>Podatność Server-Side Template Injection (SSTI) .....</b>	<b>383</b>
Wstęp .....	385
Silniki szablonów .....	385
Server-Side Template Injections – Velocity .....	388
Teoria, metodyka, narzędzia .....	395
Identyfikacja podatności .....	395
Identyfikacja silnika .....	400
Eksploatacja .....	401
Narzędzia i przykład zastosowania – Freemarker .....	402
Zapobieganie i obrona .....	415
Rezygnacja z szablonów (przynajmniej częściowo) .....	415
Użycie bezpiecznych silników .....	416
Sandboxing .....	416
Hardening .....	423
Podsumowanie .....	423
Polecane zasoby w sieci .....	423
<b>Podatność Server-Side Request Forgery (SSRF) .....</b>	<b>425</b>
Wstęp .....	427
Przykłady .....	428
Możliwe skutki wykorzystania podatności .....	429
Częste miejsca występowania podatności SSRF .....	430
Podstawy .....	430
Pliki XML .....	430
XXE .....	430
Document type definition (DTD) .....	431
XInclude .....	431
SVG/XLink .....	431
XSLT .....	432
Formaty pakietów biurowych .....	432
Inne formaty plików .....	433
Dowolne formaty .....	433
MP4 .....	434
Biblioteki .....	434
Mechanizm uploadu .....	435
Inne miejsca .....	435
Inne niż HTTP protokoły, które mogą zostać wykorzystane w SSRF .....	436
Wstęp .....	436
HTTPS .....	437
PHAR .....	438
Gopher .....	439
Inne protokoły .....	440
Często występujące błędy w filtrach anti-SSRF .....	440
Filtry blacklist .....	440
Filtry whitelist .....	441
Metody ochrony .....	443
Podsumowanie .....	444

<b>Podatność SQL Injection</b>	447
Wstęp	449
Czym jest SQL Injection	449
Sposoby wykorzystania SQL Injection	451
UNION-based	451
ERROR-based	455
BLIND (content based)	456
BLIND (time based)	459
Stacked queries	460
Skutki wykorzystania SQL Injection	460
Wydobycie dowolnych danych z bazy	460
Omijanie ekranu logowania	461
Modyfikacja/usuwanie danych	462
Przykład 1. Zmiana hasła administratora: UPDATE	462
Przykład 2. Ścieżka dostępu	462
Przykład 3. Wykonanie kodu PHP	463
Odczyt plików z dysku	463
Zapis plików na dysku	464
Wykonywanie poleceń systemu operacyjnego	464
Jak szukać SQL Injection	466
Second-order SQL Injection	470
Metody ochrony przed SQL Injection	471
Zapytania parametryzowane	471
Walidacja typów danych	472
Stosowanie systemów klasy ORM	472
Hardening bazy danych	473
Podsumowanie	473
<b>Podatność Path Traversal</b>	475
Wstęp	477
Logika podatności	477
Zagrożenia	477
Ujawnienie nadmiarowych informacji	478
Ujawnienie plików konfiguracyjnych	478
Zdalne wykonanie kodu	478
Szersze spojrzenie na problem	478
Przykłady podatności	479
GlassFish Server	479
ColoradoFTP 1.3	480
Testowanie	480
Automatyzacja	480
Omijanie filtrów	482
Ochrona	482
Blokowanie ataku poprzez podmianę tekstu	482
Modelowanie zagrożeń	483
Podsumowanie	483
<b>Code Injection i Command Injection</b>	
– przegląd wektorów ataku w aplikacjach webowych	485



Wstęp .....	487
Czym jest Code Injection oraz Command Injection? .....	487
Wektory ataków .....	488
Formalność – Eval .....	488
Klasyka: Local File Inclusion i Remote File Inclusion .....	489
Podatności w mechanizmie wgrywania plików .....	491
Uruchamianie zewnętrznego oprogramowania – Command Injection .....	492
Panele administracyjne .....	494
Cross-Site Scripting a Code Injection .....	497
Tryb debug a serwer produkcyjny .....	497
Od SQL Injection do RCE .....	498
XSLT .....	499
WebDAV .....	500
Podatności w bibliotekach .....	501
Przegląd podatności .....	501
Drobne błędy .....	504
Czy to już wszystko? .....	504
Skutki .....	505
Wykradanie danych .....	505
Eskalacja .....	505
Jakby tego było mało – webshelle i tylne furtki .....	506
Podsumowanie .....	506
<b>Uwierzytelnianie, zarządzanie sesją, autoryzacja .....</b>	<b>509</b>
Wstęp .....	511
Teoria i podstawowe pojęcia .....	511
Błędne pojęcia .....	511
Identyfikacja .....	512
Uwierzytelnianie .....	512
Zarządzanie sesją .....	512
Autoryzacja .....	513
Błędy bezpieczeństwa .....	513
Rodzaje mechanizmów uwierzytelniania .....	513
Sposób klasyczny (zapytanie oraz ciasteczka HTTP) .....	513
HTTP Basic Authentication .....	514
OpenID Connect .....	514
Klucze API .....	515
Uwierzytelnianie certyfikatem .....	515
Kerberos oraz NTLM .....	515
Uwierzytelnianie .....	516
Nie ma HTTPS, nie ma poświadczeń .....	517
Brak uwierzytelnienia .....	518
Po co wyważać, skoro można obejść – omijanie uwierzytelnienia .....	520
Poświadczenia podane na tacy .....	522
Praca po stronie serwera .....	524
Enumeracja użytkowników .....	525
Automatyzacja ataku, czyli ataki brute-force .....	526
Zabezpieczenia tak dobre, że aż złe – jak skutecznie bronić się przed atakami siłowymi .....	529
Jak nie drzwiami, to oknem .....	531

Reset hasła .....	531
Niebezpieczne pytania bezpieczeństwa .....	534
Podaj hasło jeszcze raz – krytyczne akcje .....	535
Higiena przechowywania haseł .....	536
Polityka bezpieczeństwa haseł .....	536
Logowanie zdarzeń .....	536
Zarządzanie sesją .....	537
Wymyślanie koła na nowo .....	538
Regenerowanie sesji .....	539
Session Fixation .....	540
Obsługa równoległych sesji .....	541
Strzec jak oka w głowie .....	541
Odpowiednia złożoność .....	544
Odpowiedni czas życia .....	544
Unieważnianie sesji .....	546
ID sesji jako fingerprint .....	546
ID sesji jako zagrożenie .....	546
Kłopotliwa funkcja „Zapamiętaj mnie” .....	547
Autoryzacja .....	549
Brak autoryzacji, autoryzacja na warstwie interfejsu oraz zaniechania .....	549
Podejmowanie decyzji i eskalacja uprawnień .....	551
Problem globalnych identyfikatorów .....	552
Centralizacja .....	553
Rozliczalność oraz niezaprzeczalność .....	554
Krytyczne operacje .....	554
Wybór modelu autoryzacji .....	555
Zasada najmniejszego uprzywilejowania .....	555
Co można zrobić lepiej .....	556
Uwierzytelnianie dwuskładnikowe .....	556
Poszerzanie wiedzy .....	558
Podsumowanie .....	558
Polecane zasoby w sieci .....	558
<b>Pułapki w przetwarzaniu plików XML .....</b>	<b>563</b>
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
<b>Bezpieczeństwo API REST .....</b>	<b>577</b>
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
RESEasy .....	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 oczekują 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 .....	600
Dostęp do panelu administracyjnego .....	601
Kradzież środków z jednego z największych banków w Indiach .....	602
Dostęp do wewnętrznego API .....	602
Podsumowanie .....	603
<b><u>Niebezpieczeństwa JSON Web Token (JWT)</u></b> .....	<b>607</b>
Wstęp .....	609
Definicja .....	609
Kolejny przykład JWT i pierwsze problemy bezpieczeństwa .....	610
Kolec pierwszy: nadmierna komplikacja .....	612
Kolec drugi: none .....	612
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
JWT – DOBRE PRAKTYKI .....	622
Polecane zasoby w sieci .....	623
<b><u>Zalety i wady OAuth 2.0 z perspektywy bezpieczeństwa</u></b> .....	<b>625</b>
Wstęp .....	627
Czym jest OAuth 2.0? .....	627

Wykorzystywana terminologia .....	628
Zasada działania OAuth .....	629
Korzyści wynikające z wykorzystania 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 .....	636
Serwer autoryzujący .....	637
Problematiczne 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
<b><u>Bezpieczeństwo protokołu WebSocket</u></b> .....	<b>655</b>
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
<b>Flaga SameSite – jak działa i przed czym zapewnia ochronę?</b> .....	<b>671</b>
Wstęp .....	673
Podstawy – przeglądarki, ciasteczka i sesja .....	673
Nadużycie .....	674
SameSite na ratunek .....	674
Lax – nawigacja „top-level” oraz bezpieczne metody HTTP .....	679
Bilans zysków i strat .....	681
Podsumowanie: lek na całe zło? .....	682
<b>Niebezpieczeństwa deserializacji w PHP</b> .....	<b>685</b>
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
<b>Niebezpieczeństwa deserializacji w Pythonie (moduł pickle)</b> .....	<b>701</b>
Wstęp .....	703
Jak działa moduł pickle .....	703
Złośliwe użycie pickle .....	705
Sposoby ochrony .....	706
Podsumowanie .....	708
<b>Niebezpieczeństwa deserializacji w .NET</b> .....	<b>711</b>
Wstęp .....	713
Środowisko programistyczne .....	714
Json.Net .....	715
XmlSerializer .....	726
CVE .....	731
Breeze (CVE-2017-9424) .....	732
DotNetNuke Platform (CVE-2017-9822) .....	732
Microsoft SharePoint (CVE-2019-0604) .....	734
<b>Niebezpieczeństwa deserializacji w Javie</b> .....	<b>737</b>
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
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 vs. 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
<b>Wprowadzenie do programów bug bounty .....</b>	<b>775</b>
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