

Bezpieczeństwo aplikacji
mobilnych

Podręcznik hakera



Dominic Chell, Tyrone Erasmus
Shaun Colley, Ollie Whitehouse

Helion 

Tytuł oryginału: The Mobile Application Hacker's Handbook

Tłumaczenie: Robert Górczyński

ISBN: 978-83-8322-548-7

Copyright © 2015 by John Wiley & Sons, Inc., Indianapolis, Indiana

Ali Rights Reserved. This translation published under license with the original publisher John Wiley & Sons. Inc.

Translation copyright © 2017, 2023 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. 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.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

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

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność



Spis treści

O autorach	15
O recenzencie technicznym	17
Wstępniak	19
Wprowadzenie	21
Ogólne omówienie książki	22
W jaki sposób zorganizowana jest ta książka?	22
Kto powinien przeczytać tę książkę?	27
Niezbędne narzędzia	27
Co znajdziesz w witrynie internetowej?	27
Rozdział 1. (Nie)bezpieczeństwo aplikacji mobilnych	29
Ewolucja aplikacji mobilnych	30
Najczęstsze kategorie aplikacji mobilnych	31
Zalety aplikacji mobilnych	32
Bezpieczeństwo aplikacji mobilnych	32
Kluczowe czynniki problemu	35
Projekt OWASP Mobile Security	36
Narzędzia rekomendowane przez OWASP Mobile Security	40
Przyszłość dziedziny zapewniania bezpieczeństwa aplikacjom mobilnym	42
Podsumowanie	43

Rozdział 2.	Analiza aplikacji iOS	45
	Poznajemy model bezpieczeństwa	45
	Inicjalizacja systemu iOS za pomocą łańcucha procedur bezpiecznego rozruchu	46
	Wprowadzenie Secure Enclave	47
	Ograniczenie procesów aplikacji dzięki podpisywaniu kodu	47
	Izolacja aplikacji za pomocą piaskownicy na poziomie procesu	48
	Ochrona informacji za pomocą szyfrowania przechowywanych danych	48
	Ochrona przed atakami dzięki funkcjom ograniczającym możliwość wykorzystania luk w zabezpieczeniach	49
	Poznajemy aplikacje iOS	50
	Dystrybucja aplikacji iOS	51
	Struktura aplikacji	52
	Instalacja aplikacji	53
	Poznajemy uprawnienia aplikacji	54
	Omówienie jailbreakingu	56
	Powody przeprowadzania jailbreakingu urządzenia	57
	Rodzaje jailbreakingu	58
	JailbreakMe v3 Saffron	58
	Jailbreak evasi0n	60
	Jailbreak evasi0n7	60
	Przygotowanie środowiska testowego	61
	Przygotowanie podstawowego zestawu narzędzi	62
	Przeglądanie systemu plików	67
	Pliki typu property list	69
	Binarne pliki cookie	69
	Bazy danych SQLite	70
	Poznajemy API Data Protection	70
	Poznajemy pęk kluczy w systemie iOS	73
	Kontrola dostępu i polityka uwierzytelniania w iOS 8	75
	Uzyskanie dostępu do pęku kluczy w iOS	76
	Poznajemy Touch ID	77
	Inżynieria odwrotna plików binarnych w iOS	79
	Analiza plików binarnych w systemie iOS	79
	Identyfikacja funkcji związanych z zapewnianiem bezpieczeństwa	82
	Deszyfrowanie plików binarnych pochodzących ze sklepu App Store	85
	Analiza odszyfrowanych plików binarnych	88
	Deasemblacja i dekompilacja aplikacji iOS	93
	Podsumowanie	93

Rozdział 3.	Atakowanie aplikacji iOS	95
	Wprowadzenie do bezpieczeństwa w warstwie transportu	95
	Identyfikacja niebezpieczeństw czyhających w warstwie transportu	96
	CVE-2014-1266: SSL/TLS „Goto Fail”	101
	Przechwytywanie szyfrowanej komunikacji	103
	Niebezpieczeństwa związane z instalacją profili	106
	Ominięcie mechanizmu przypinania certyfikatu	106
	Niebezpieczeństwa związane z instalacją narzędzi pozwalających obejść kwestie zaufania	107
	Identyfikacja niebezpiecznego magazynu danych	107
	Modyfikacja aplikacji za pomocą deasemblera Hopper	111
	Atak na środowisko uruchomieniowe iOS	117
	Poznajemy języki Objective-C i Swift	118
	Instrumentacja środowiska uruchomieniowego iOS	120
	Poznajemy komunikację międzyprocesową	142
	Atak na procedurę obsługi protokołu	142
	Niebezpieczna procedura obsługi w aplikacji Skype	145
	Rozszerzenia aplikacji	145
	Ataki typu injection	147
	Atak injection na komponent UIWebView	147
	Aplikacja Skype na platformie iOS i ataki XSS	150
	Atak typu injection na magazyn danych po stronie klienta	150
	Atak typu injection na analizator składni XML	151
	Atak typu injection na procedurę obsługi pliku	153
	Podsumowanie	154
Rozdział 4.	Identyfikowanie problemów w implementacji aplikacji iOS	155
	Ujawnienie danych osobowych	155
	Obsługa identyfikatora urządzenia	156
	Przetwarzanie książki adresowej	157
	Obsługa danych geolokalizacji	157
	Wykrywanie wycieku danych	158
	Wyciek danych w dziennikach zdarzeń aplikacji	159
	Wykrywanie wycieku danych poprzez schowek systemowy	159
	Obsługa zmiany stanu aplikacji	161
	Buforowanie klawiatury	162
	Buforowanie odpowiedzi HTTP	163
	Uszkodzenie pamięci w aplikacjach iOS	164
	Luki w zabezpieczeniach związane z ciągiem tekstowym formatowania	164
	Próba użycia obiektu po jego usunięciu	167
	Inne problemy związane z implementacjami kodu natywnego	168
	Podsumowanie	168

Rozdział 5.	Tworzenie bezpiecznych aplikacji iOS	169
	Ochrona danych w aplikacji	169
	Ogólne reguły projektowe	170
	Implementacja szyfrowania	171
	Ochrona danych w trakcie transportu	174
	Unikanie luk w zabezpieczeniach pozwalających na przeprowadzenie ataku typu injection	176
	Unikanie ataków typu SQL injection	176
	Unikanie ataków typu XSS	177
	Ochrona aplikacji z użyciem zabezpieczeń binarnych	178
	Wykrycie przeprowadzenia jailbreakingu urządzenia	179
	Zabezpieczenie środowiska uruchomieniowego aplikacji	183
	Zabezpieczenie aplikacji przed modyfikacjami	187
	Implementacja zabezpieczeń antydebugowania	188
	Zaciemnienie aplikacji	189
	Podsumowanie	190
Rozdział 6.	Analiza aplikacji Android	191
	Przygotowanie pierwszego środowiska Android	192
	Poznajemy aplikacje Android	197
	Podstawy systemu Android	197
	Poznajemy pakiety Androida	199
	Użycie narzędzi do przeglądania zasobów Androida	203
	Wprowadzenie do komponentów aplikacji	213
	Spojrzenie pod maskę	218
	ART, czyli zamiennie środowisko uruchomieniowe dla oprogramowania Dalvik	222
	Poznajemy model zapewniania bezpieczeństwa	223
	Podpisanie kodu	223
	Poznajemy uprawnienia	229
	Słowo ostrzeżenia dotyczące taktyki najczęściej stosowanej przez oprogramowanie typu malware	235
	Piaskownica aplikacji	235
	Szyfrowanie systemu plików	237
	Ogólne mechanizmy obronne przed wykorzystaniem luk w zabezpieczeniach	239
	Dodatkowe mechanizmy obronne jądra przed eskalacją uprawnień	241
	Poznajemy kwestię uzyskania uprawnień użytkownika root	241
	Gingerbreak — wykorzystanie luk w zabezpieczeniach kodu jądra AOSP	245
	Exynos — wykorzystanie luk w zabezpieczeniach niestandardowych sterowników	245
	Samsung Admire — nadużycie uprawnień pliku za pomocą dowiązań symbolicznych	246

Acer Iconia — wykorzystanie luk w zabezpieczeniach plików binarnych SUID	247
Błędy klucza głównego — wykorzystanie luk w zabezpieczeniach kodu systemowego AOSP	247
Towelroot — wykorzystanie luk w zabezpieczeniach jądra w systemie Android	248
Modyfikacja w urządzeniu Nexus własnego obrazu przeznaczonego do odzyskiwania systemu	249
Inżynieria odwrotna aplikacji	250
Pobieranie plików APK	251
Wyświetlenie pliku manifestu	252
Wyświetlanie plików XML	253
Wygenerowanie danych wyjściowych do pliku	254
Deasemblacja kodu bajtowego DEX	254
Dekompilacja kodu bajtowego DEX	256
Dekompilacja zoptymalizowanego kodu DEX	259
Deasemblacja kodu natywnego	261
Narzędzia dodatkowe	261
Praca ze środowiskiem ART	262
Podsumowanie	264
Rozdział 7. Atakowanie aplikacji Android	265
Dziwactwa modelu zapewniania bezpieczeństwa	266
Współpraca z komponentami aplikacji	266
Atak na komponenty aplikacji	272
Poznajemy intencje	273
Poznaj Sieve, czyli pierwszą atakowaną aplikację	275
Przeprowadzanie ataku na czynności	279
Kilka słów na temat aliasów czynności	281
Rzeczywisty przykład: CVE-2013-6271 — usunięcie blokady urządzenia z wydania Android 4.3 lub starszego	283
Rzeczywisty przykład — zmiana kodu PIN w urządzeniu bez podania dotychczasowego	287
Wykorzystanie luk w niezabezpieczonych dostawcach treści	289
Użycie istniejących narzędzi do wykrywania SQL injection	295
Rzeczywisty przykład — luki w zabezpieczeniach wielu aplikacji Androida zainstalowanych standardowo w urządzeniach Samsunga	295
Rzeczywisty przykład — aplikacja Shazam	300
Przeprowadzanie ataku na niezabezpieczone usługi	301
Rzeczywisty przykład — usługa schowka w urządzeniach firmy Samsung	302
Błędy podczas kompilacji własnych klas Javy	310
Przeprowadzanie ataku na odbiorcę komunikatów	310
Systemowe komunikaty rozgłoszeniowe	311

Rzeczywisty przykład: CVE-2013-6272 — zainicjowanie lub zerwanie połączenia bez odpowiednich uprawnień w systemie Android 4.4.2 lub starszym	312
Rzeczywisty przykład — zdalne usunięcie zawartości urządzenia Samsung Galaxy	318
Uzyskanie dostępu do pamięci masowej i dzienników zdarzeń	319
Uprawnienia plików i katalogów	319
Rzeczywisty przykład — możliwy do zapisu przez wszystkich użytkowników skrypt DroidWall wykonany przez użytkownika root	322
Praktyki dotyczące szyfrowania pliku	324
Pamięć masowa na karcie SD	325
Rzeczywisty przykład — pamięć masowa aplikacji WhatsApp	326
Rejestracja danych	326
Błędne zastosowanie niezabezpieczonej komunikacji	327
Przegląd ruchu sieciowego	327
CVE-2012-6636 — wykonanie dowolnego kodu zdefiniowanego w metodzie addJavaScriptInterface()	335
Inne mechanizmy komunikacji	336
Inne płaszczyzny ataku	340
Przeprowadzanie ataku na kod natywny	340
Wykorzystanie luk w zabezpieczeniach błędnie skonfigurowanych atrybutów pakietu	347
Przeprowadzenie ataku na aplikację z włączoną opcją debuggable z poziomu innej aplikacji bez szczególnych uprawnień	354
Dodatkowe techniki testowania	355
Stosowanie poprawek w aplikacji	356
Błędy podczas podpisywania pakietu	358
Manipulacja środowiskiem uruchomieniowym	359
Podsumowanie	364
Rozdział 8. Identyfikowanie problemów w implementacji aplikacji Android	365
Przegląd standardowo zainstalowanych aplikacji	365
Wyszukanie aplikacji o szerokich uprawnieniach	366
Wyszukiwanie płaszczyzn dla zdalnego ataku	369
Wyszukiwanie lokalnych luk w zabezpieczeniach	376
Wykorzystanie luk w zabezpieczeniach urządzeń	377
Narzędzia ataku	377
Poznajemy poziomy uprawnień	385
Praktyczne ataki fizyczne	387
Praktyczne zdalne ataki	398
Infiltracja danych użytkownika	426
Użycie istniejących modułów narzędzia drozer	426
Inne techniki do zastosowania w uprawnionych scenariuszach	431
Podsumowanie	436

Rozdział 9. Tworzenie bezpiecznych aplikacji Android	437
Reguła najmniejszego odkrycia się	437
Komponenty aplikacji	438
Magazyn danych	438
Współpraca z niezaufanymi źródłami	438
Żądanie minimalnych uprawnień	438
Sprawdzenie plików znajdujących się w pakiecie APK	439
Podstawowe mechanizmy obronne	439
Przegląd punktów wejścia w komponentach aplikacji	439
Bezpieczne przechowywanie plików	445
Bezpieczne udostępnianie plików innym aplikacjom	449
Prowadzenie bezpiecznej komunikacji	450
Zabezpieczenie komponentu WebView	453
Konfiguracja pliku manifestu	455
Rejestracja zdarzeń	457
Zmniejszenie ryzyka związanego z kodem natywnym	457
Skrypt checksec nie działa	458
Zaawansowane mechanizmy zabezpieczeń	458
Wykrycie obniżenia poziomu ochrony	459
Ochrona niewyeksportowanych komponentów	460
Spowolnienie procesu inżynierii odwrotnej	460
Zaciemnianie kodu źródłowego	460
Wykrywanie użycia konta użytkownika root	461
Wykrycie debugowania	463
Wykrycie modyfikacji aplikacji	463
Podsumowanie	464
Rozdział 10. Analiza aplikacji Windows Phone	467
Poznajemy model zapewniania bezpieczeństwa	468
Podpisywanie kodu i DRM	468
Piaskownica aplikacji	469
Szyfrowanie przechowywanych danych	471
Proces zgłaszania aplikacji do sklepu Microsoft Store	472
Poznajemy funkcje mechanizmów obronnych	474
Poznajemy aplikacje na platformie Windows Phone 8.x	482
Pakiety aplikacji	482
Języki programowania i typy aplikacji	482
Manifest aplikacji	484
Katalogi aplikacji	489
Rozpowszechnianie aplikacji Windows Phone	489
Przygotowanie środowiska testowego	493
Narzędzia SDK	494
Odblokowanie możliwości urządzenia	499

Wykorzystanie dostępu do systemu plików	512
Wykorzystanie dostępu do rejestru	514
Użyteczne narzędzia hakera	514
Analiza plików binarnych aplikacji	515
Proces inżynierii odwrotnej	515
Analiza funkcji mechanizmów obronnych	516
Podsumowanie	517
Rozdział 11. Atakowanie aplikacji Windows Phone	519
Analiza pod kątem punktów wejścia danych	519
Kontrolki WebBrowser i WebView	520
Bluetooth	522
Sesje HTTP	524
Gniazda sieciowe	525
NFC	525
Kod kreskowy	527
Karta SD	528
Interfejsy komunikacji międzyprocesowej	530
Powiadomienia typu toast	532
Atak na warstwę transportową	533
Identyfikacja i przechwytywanie komunikacji w postaci zwykłego tekstu	533
Identyfikacja i przechwytywanie komunikacji HTTPS	537
Przechwytywanie ruchu sieciowego innego niż HTTP i HTTPS	539
Błędy związane z weryfikacją certyfikatu SSL	539
Ataki na kontrolki WebBrowser i WebView	541
Ataki typu XSS	541
Ataki skryptów lokalnych	543
Komunikacja między językami JavaScript i C#	548
Identyfikacja luk w zabezpieczeniach implementacji IPC	549
Procedura obsługi protokołu	549
Procedura obsługi pliku	553
Powiadomienia typu toast	557
Atak na analizator składni XML	566
Wprowadzenie do API XDocument	566
Atak DoS podczas rozwinięcia encji	569
Atak typu XXE	571
Atak na bazę danych	574
API LINQ to SQL	574
SQLite i SQLCipher	575
Atak na procedurę obsługi pliku	579
Wprowadzenie do obsługi pliku	579
Ataki polegające na poruszaniu się po katalogach	581
Modyfikacje podzespołu .NET	584
Podsumowanie	591

Rozdział 12. Identyfikowanie problemów w implementacji aplikacji Windows Phone	593
Identyfikacja niebezpiecznego magazynu danych ustawień aplikacji	594
Wykrywanie wycieku danych	597
Magazyn danych plików cookie HTTP(S)	598
Buforowanie HTTP(S)	599
Rejestracja danych w aplikacji	599
Identyfikacja niebezpiecznego magazynu danych	600
Niezaszyfrowany magazyn danych plików	600
Niebezpieczny magazyn bazy danych	603
Niebezpieczne generowanie liczby losowej	607
Przewidywalność System.Random	607
Wiele egzemplarzy System.Random	610
Bezpieczeństwo wątku System.Random	610
Niebezpieczna kryptografia i użycie hasła	611
Klucze kryptograficzne na stałe umieszczone w kodzie	612
Niebezpieczny magazyn kluczy kryptograficznych	612
Przechowywanie klucza lub hasła w niemodyfikowalnym obiekcie ciągu tekstowego	613
Nieudane usunięcie z pamięci klucza kryptograficznego lub hasła	614
Niebezpieczne generowanie klucza	615
Użycie słabych algorytmów kryptograficznych i trybów oraz kluczy o niewystarczającej długości	617
Użycie statycznych wektorów inicjalizacji	620
Błędne użycie API Data Protection na platformie Windows Phone	621
Wykrywanie luk w zabezpieczeniach kodu natywnego	623
Przepełnienie bufora stosu	624
Przepełnienie bufora sterty	626
Inne błędy związane z obsługą liczb całkowitych	628
Błędy ciągu tekstowego formatowania	630
Błędy związane z indeksowaniem tablicy	631
Błędy związane z odmową usług	632
Niebezpieczny kod C#	632
Podsumowanie	633
Rozdział 13. Tworzenie bezpiecznych aplikacji Windows Phone	635
Ogólne rozważania dotyczące bezpiecznego projektu aplikacji	635
Bezpieczne szyfrowanie i przechowywanie danych	636
Bezpieczne algorytmy i tryby szyfrowania	636
Generowanie klucza i zarządzanie nim	636
Szyfrowanie pliku	637
Szyfrowanie bazy danych	639

Bezpieczne generowanie liczby losowej	640
Zapewnianie bezpieczeństwa danych w pamięci i usuwanie zawartości pamięci	640
Uniknięcie ataku typu SQL injection	642
Implementacja bezpiecznej komunikacji	643
Użycie SSL i TLS	643
Weryfikacja certyfikatu SSL i TLS	644
Uniknięcie ataków typu XSS w komponentach WebBrowser i WebView	645
Użycie SSL i TLS w komunikacji sieciowej	645
Wyłączenie obsługi JavaScript	646
Bezpieczne tworzenie dynamicznego kodu HTML i JavaScript	646
Unikanie ataków przeprowadzanych przez skrypty lokalne	647
Bezpieczne przetwarzanie danych XML	647
Usunięcie bufora internetowego i plików cookie	648
Usunięcie plików cookie	648
Usunięcie bufora internetowego	649
Unikanie błędów kodu natywnego	649
Użycie funkcji mechanizmów obronnych	650
Podsumowanie	651
Rozdział 14. Tworzenie aplikacji mobilnych niezależnych od platformy	653
Wprowadzenie do aplikacji mobilnych niezależnych od platformy	653
Zastosowanie funkcjonalności natywnej	655
Udostępnienie funkcjonalności natywnej w systemie Android	656
Udostępnienie funkcjonalności natywnej w systemie iOS	657
Udostępnienie funkcjonalności natywnej w systemie Windows Phone	658
Poznajemy frameworki PhoneGap i Apache Cordova	659
Funkcje standardowe PhoneGap	659
Zapewnienie bezpieczeństwa aplikacji utworzonych za pomocą frameworków PhoneGap i Cordova	660
Wiele luk w zabezpieczeniach frameworka Cordova	661
Ominięcie białej listy we frameworku Cordova dla protokołu innego niż HTTP	663
Podsumowanie	664
Skorowidz	665

(Nie)bezpieczeństwo aplikacji mobilnych

Nie ulega wątpliwości, że urządzenia mobilne zmieniły świat. W szczególności sposób pracy, a także współdziałania i komunikacji z innymi osobami nie będzie już więcej taki sam. Użytkownicy urządzeń mobilnych otrzymali wręcz nieskończone możliwości, które przez cały czas pozostają dostępne w zasięgu ich palców. Sprawdzenie salda konta bankowego, poczty elektronicznej, przeprowadzenie transakcji na giełdzie i znacznie więcej innych operacji jest po prostu na wyciągnięcie ręki. Opracowywanie aplikacji stało się na tyle popularnym zajęciem, że slogan Apple „Do tego celu jest aplikacja” nie odbiega od rzeczywistości.

W tym rozdziale dowiesz się, jak przebiegała ewolucja aplikacji mobilnych, oraz poznasz oferowane przez nie korzyści. Przedstawię również dane dotyczące podstawowych luk w zabezpieczeniach, które występują w aplikacjach mobilnych. Te dane pochodzą bezpośrednio z mojego doświadczenia i pokazują, że większość aplikacji mobilnych jest daleka od bezpiecznych. Następnie przejdę do skategoryzowania wspomnianych luk w zabezpieczeniach na podstawie opracowanej przez organizację OWASP (ang. *open web application security project*) listy dziesięciu największych zagrożeń dla aplikacji mobilnych. Zaprezentuję również ogólne omówienie wybranych narzędzi typu open source zalecanych przez OWASP przeznaczonych do sprawdzania bezpieczeństwa aplikacji mobilnych. Zobaczysz też, jak za pomocą tych narzędzi identyfikować pewne problemy wskazywane przez OWASP i gdzie ich szukać. Na końcu rozdziału przejdę do najnowszych trendów w zakresie zapewniania bezpieczeństwa aplikacjom mobilnym i tego, czego można w tym obszarze oczekiwać w przyszłości.

Ewolucja aplikacji mobilnych

Pierwsze aplikacje dla telefonów komórkowych zostały opracowane samodzielnie przez samych producentów urządzeń. Dla tych aplikacji istniała szczątkowa dokumentacja i można było znaleźć niewiele informacji dotyczących wewnętrznego sposobu działania tych programów. Przyczyną takiego stanu rzeczy były prawdopodobnie obawy producentów, że otwarcie platformy na aplikacje opracowane przez firmy zewnętrzne mogłoby doprowadzić do ujawnienia tajemnic jeszcze nie do końca dopracowanej technologii. Wczesne aplikacje były podobne do wielu dostarczanych przez producentów także w telefonach obecnej generacji, czyli na przykład książka adresowa, kalendarz i proste gry, takie jak popularna gra *Snake* stworzona przez Nokię.

Wraz z pojawieniem się smartfonów jako następców palmtopów (ang. *personal digital assistant*, PDA) tworzenie oprogramowania naprawdę nabrało rozpędu. Wzrost liczby dostępnych aplikacji dla urządzeń mobilnych mógł być też spowodowany bezpośrednio zwiększeniem mocy obliczeniowej oraz możliwości oferowanych przez smartfony, a także pochodzącym z rynku konsumencięgo wzrostem popytu na funkcjonalność. Gdy smartfony ewoluowały, aplikacje mobilne zaczęły wykorzystywać usprawnienia wprowadzane na platformach. Do wspomnianych usprawnień zaliczamy między innymi dodanie systemu nawigacji satelitarnej (ang. *global positioning system*, GPS), aparatu fotograficznego, zwiększenie czasu działania urządzenia na baterii, wprowadzenie lepszej jakości ekranów oraz procesorów. Te wszystkie ulepszenia spowodowały pojawianie się coraz bardziej rozbudowanych aplikacji, które dzisiaj znamy.

Opracowywanie aplikacji mobilnych przez firmy zewnętrzne urzeczywistniło się w 2008 roku, gdy firma Apple ogłosiła powstanie pierwszej usługi przeznaczonej do rozpowszechniania tego rodzaju aplikacji — App Store. Sklep App Store powstał rok po wypuszczeniu na rynek pierwszego smartfona Apple, czyli iPhone'a. Taki sam ruch wykonała firma Google ze swoim Android Market, który obecnie jest znany pod nazwą Google Play. Do dzisiaj pojawiło się wiele kolejnych sklepów oferujących aplikacje dla urządzeń mobilnych, między innymi Microsoft Store, Amazon Appstore i BlackBerry World.

Coraz większa konkurencja na rynku tworzenia aplikacji dla urządzeń mobilnych doprowadziła do pewnej fragmentacji rynków programistycznych. Większość aplikacji mobilnych jest przeznaczona dla konkretnej platformy, a twórcy oprogramowania są zmuszeni do pracy z różnymi systemami operacyjnymi, językami programowania i narzędziami, aby móc opracować programy dla wielu platform. Aplikacje na platformę iOS tradycyjnie były tworzone w języku Objective-C, aplikacje dla systemów Android i BlackBerry są tworzone w języku Java (w przypadku BlackBerry 10 również w Qt), natomiast aplikacje Windows Phone z użyciem .NET Framework. Tego rodzaju fragmentacja bardzo często prowadzi do konieczności istnienia w firmie tworzącej oprogramowanie dla urządzeń mobilnych wielu zespołów programistów oraz wielu baz kodu.

Jednak w ostatnim czasie na polu tworzenia aplikacji działających na różnych platformach mobilnych można dostrzec pewne zmiany, które wynikają z chęci zmniejszenia kosztów związanych z opracowywaniem programów. Niezależne od platformy frameworki oraz aplikacje oparte na standardzie HTML5 i działające w przeglądarkach WWW zyskały popularność dokładnie z wymienionych powodów. Można się spodziewać, że ten trend będzie coraz większy.

Najczęstsze kategorie aplikacji mobilnych

Aplikacje mobilne powstały dla praktycznie każdego możliwego do wyobrażenia sobie celu. Tylko w sklepach utrzymywanych przez Apple i Google znajduje się ponad 2 miliony aplikacji przeznaczonych do różnorodnych zastosowań, między innymi wymienionych poniżej:

- bankowość internetowa (Barclays),
- zakupy (Amazon),
- serwisy społecznościowe (Facebook),
- streaming (Sky Go),
- hazard (Betfair),
- komunikatory internetowe (WhatsApp),
- komunikatory głosowe (Skype),
- poczta elektroniczna (Gmail),
- współdzielenie plików (Dropbox),
- gry (*Angry Birds*).

Aplikacje mobilne bardzo często oferują funkcjonalność dostępną także za pomocą aplikacji internetowych. W wielu przypadkach używane jest to samo podstawowe API po stronie serwera, a widok przeznaczony dla smartfonów jest generowany na poziomie warstwy prezentacyjnej.

Poza aplikacjami dostępnymi w różnych sklepach istnieją jeszcze aplikacje, które zostały szeroko zaadaptowane w świecie biznesu i przeznaczone do zapewniania obsługi kluczowych zadań biznesowych. Wiele tego rodzaju aplikacji oferuje dostęp do ściśle strzeżonych danych korporacyjnych. Poniżej wymienię wybrane rodzaje programów, na które się natknąłem podczas wykonywania zadań zleconych przez klientów:

- Aplikacje magazynu dokumentów, pozwalające użytkownikom na uzyskanie dostępu na żądanie do ściśle strzeżonych dokumentów biznesowych.
- Aplikacje przeznaczone do prowadzenia małej księgowości, umożliwiające użytkownikom tworzenie, przechowywanie i przekazywanie do wewnętrznych systemów zestawień wydatków.
- Aplikacje HR, pozwalające użytkownikom na uzyskanie dostępu do listy płac, kart pracy, informacji o dniach wolnych oraz innych danych, które można uznać za poufne.
- Aplikacje usług wewnętrznych, takie jak zoptymalizowane pod kątem zapewniania dostępu do zasobów wewnętrznych, na przykład intranetu korporacyjnego.
- Aplikacje wewnętrznych komunikatorów, umożliwiające użytkownikom komunikację w czasie rzeczywistym niezależnie od ich położenia.

We wszystkich wymienionych powyżej przykładach aplikacje są uznawane za „wewnętrzne” i zwykle są opracowywane przez daną organizację lub specjalnie na jej zlecenie. Dlatego też wiele tych aplikacji wymaga użycia wirtualnej sieci prywatnej (ang. *virtual private network*, VPN) lub dostępu do sieci wewnętrznej, aby mogły prawidłowo funkcjonować wraz z wewnętrzną infrastrukturą. Zyskującym coraz większą popularność trendem w aplikacjach korporacyjnych jest

wprowadzenie technologii „geo-fence”, w której to aplikacja używa wbudowanego w urządzenie GPS-u w celu sprawdzenia, czy użytkownik na pewno znajduje się w określonej lokalizacji, na przykład biurze firmy, a następnie na podstawie wyniku tego sprawdzenia udziela dostępu lub nakłada ograniczenia funkcjonalności.

Zalety aplikacji mobilnych

Nietrudno dostrzec, dlaczego aplikacje mobilne nabrały tak dużego znaczenia w stosunkowo krótkim okresie. Będzie komercyjne i korzyści przynoszone przez aplikacje mobilne są oczywiste. Dają organizacjom możliwość dotarcia do użytkowników końcowych praktycznie w każdej chwili, a ze względu na ogromną popularność smartfonów trafiają do znacznie większej grupy docelowej. Jednak na sukces aplikacji mobilnych wpływ miało również wiele innych czynników, z których wybrane wymienię poniżej:

- Fundamenty dla aplikacji mobilnych powstały na bazie istniejących i popularnych protokołów. W szczególności protokół HTTP jest doskonale znany przez programistów i dlatego pozostaje dość powszechnie stosowany we wdrożeniach mobilnych.
- Techniczny rozwój smartfonów pozwolił aplikacjom mobilnym na zaoferowanie znacznie bardziej zaawansowanych funkcji oraz lepszych wrażeń, jakie odnoszą użytkownicy tych aplikacji. Usprawnienia wprowadzone w zakresie rozdzielczości ekranu oraz pojawienie się ekranów dotykowych stanowiły najważniejsze czynniki wpływające na poprawę interaktywności programu, szczególnie w grach. Wydłużenie czasu działania urządzenia na baterii oraz zwiększenie mocy obliczeniowej pozwalają nowoczesnym smartfonom na jednoczesne uruchomienie nie tylko jednej, ale wielu aplikacji, które na dodatek będą działały znacznie dłużej. To oznacza doskonałe udogodnienie dla użytkowników końcowych, ponieważ otrzymują pojedyncze urządzenie, które może służyć do wielu celów.
- Usprawnienia wprowadzone w technologiach sieci komórkowych doprowadziły do znacznego wzrostu szybkości komunikacji. W szczególności powszechny zasięg 3G i 4G pozwala użytkownikom na szerokopasmowy dostęp do internetu w smartfonach. Aplikacje mobilne w pełni wykorzystują zalety szybkiej komunikacji, oferując dostęp do naprawdę wielu usług internetowych.
- Prostota podstawowych technologii oraz języków programowania używanych do tworzenia aplikacji mobilnych pomogła w rewolucji mobilnej. Programy mogą być tworzone za pomocą popularnych i dopracowanych języków, takich jak Java, które są doskonale znane i mają ogromne bazy użytkowników.

Bezpieczeństwo aplikacji mobilnych

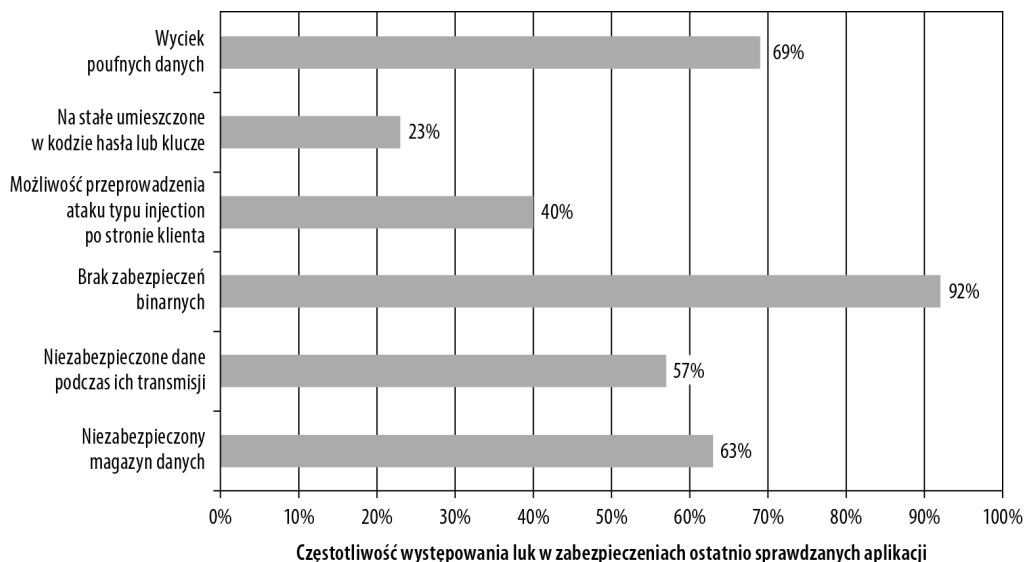
Aplikacje mobilne są narażone na istnienie wielu różnych luk w zabezpieczeniach, z których część jest dziedziczona po tradycyjnych atakach zarówno na aplikacje internetowe, jak i zwykle programy komputerowe. Jednak kilka innych klas ataków jest charakterystycznych wyłącznie dla platform mobilnych i pojawia się jako efekt sposobu użycia aplikacji mobilnych i na skutek istnienia względnie unikatowych punktów wejścia i płaszczyzn ataku tworzonych przez te aplikacje.

Poniżej wymieniałem możliwe płaszczyzny ataku na aplikacje mobilne — programiści powinni mieć świadomość ich istnienia i potrafić się przed nimi bronić.

- Większość aplikacji mobilnych prowadzi pewnego rodzaju komunikację sieciową. Ze względu na sposób używania urządzeń mobilnych wspomniana komunikacja może odbywać się przez niezaufane lub niebezpieczne sieci, takie jak Wi-Fi udostępniane przez hotel lub lokal gastronomiczny, mobilne hot-spots lub sieć komórkowa. Jeśli dane nie są odpowiednio zabezpieczone podczas ich przesyłania, aplikacja może być narażona na wiele różnych niebezpieczeństw, między innymi ujawnienie poufnych danych i ataki typu injection.
- Skoro użytkownik nosi ciągle przy sobie urządzenie mobilne, rodzi to naprawdę wiele możliwości jego zagubienia lub kradzieży. Programiści aplikacji mobilnych muszą zdawać sobie sprawę z niebezpieczeństwa, jakie wiąże się z próbami odzyskania danych z systemu plików urządzenia. Każda trwała treść pozostawiana przez aplikację w systemie plików — niezależnie od tego, czy w trwałym magazynie danych, czy w buforze tymczasowym — może potencjalnie doprowadzić do ujawnienia atakującemu poufnych danych.
- Dość unikatowym dla aplikacji mobilnych scenariuszem jest zagrożenie płynące ze strony samego urządzenia. Złośliwie działające malware szerzy się na platformie mobilnej, co przynajmniej po części wynika z istnienia nieoficjalnych kanałów dystrybucji oprogramowania. Programiści muszą zdawać sobie sprawę z niebezpieczeństwa, jakim może być atak ze strony innej aplikacji zainstalowanej w urządzeniu mobilnym.
- Aplikacje mobilne mogą pobierać dane wejściowe z wielu różnych możliwych źródeł, co powoduje utworzenie znacznej liczby potencjalnych punktów wejścia. Na przykład nietrudno natknąć się na aplikację akceptującą dane z jednego lub wielu wymienionych źródeł: NFC (ang. *near field communication*), Bluetooth, aparat fotograficzny, mikrofon, SMS (ang. *short message service*), USB (ang. *universal serial bus*), kody QR (ang. *quick response*) itd.

Do najpoważniejszych ataków na aplikacje mobilne zaliczamy te, które powodują ujawnienie poufnych danych lub też ułatwiają włamanie do urządzenia. Tego rodzaju luki w zabezpieczeniach są najczęściej związane z urządzeniem mobilnym i danymi użytkownika końcowego, a dużo rzadziej z wszystkimi użytkownikami danej usługi. Wprawdzie istniejące po stronie serwera luki w zabezpieczeniach stanowią największe zagrożenie dla wdrożeń aplikacji mobilnych jako całości, ponieważ mogą umożliwić niczym nieograniczony dostęp do systemów back endu, ale te problemy są doskonale udokumentowane i rozumiane przez programistów. Istniejące po stronie serwera luki w zabezpieczeniach wpływające na aplikacje mobilne nie będą omawiane w tej książce. Jeżeli chcesz dowiedzieć się więcej na temat tego rodzaju ataków, zachęcam Cię do zapoznania się z książką *The Web Application Hacker's Handbook* (<http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118026470.html>).

Kwestie bezpieczeństwa aplikacji mobilnych nadal są błędnie postrzegane, choć z drugiej strony ten obszar nie jest jeszcze w pełni dojrzały. Tak naprawdę większość aplikacji mobilnych wciąż można uznać za niewystarczająco zabezpieczone. Po przetestowaniu w ostatnich latach setek aplikacji mobilnych w większości z nich znajdowałem jeden lub więcej poważnych błędów w zabezpieczeniach. Na rysunku 1.1 pokazałem procentowe dane dotyczące przetestowanych od 2012 roku aplikacji mobilnych, w których znalazłem pewne najczęściej spotykane kategorie luk w zabezpieczeniach po stronie klienta.



Rysunek 1.1. Częstotliwość występowania luk w zabezpieczeniach aplikacji ostatnio sprawdzanych przez autora

- **Niezabezpieczony magazyn danych (63%).** Ta kategoria luk w zabezpieczeniach obejmuje różne defekty, które prowadzą do tego, że aplikacja przechowuje w urządzeniu mobilnym dane w postaci zwykłego tekstu. Z reguły jest stosowany zaciemniony format wykorzystujący na stałe umieszczony w kodzie klucz lub też inne rozwiązanie pozwalające osobie atakującej w dość łatwy sposób odczytać informacje przechowywane w magazynie danych.
- **Niezabezpieczone dane podczas ich transmisji (57%).** Ta kategoria obejmuje każdy przypadek, gdy aplikacja nie używa szyfrowania warstwy transportu w celu ochrony danych podczas ich transmisji. To obejmuje także sytuacje, w których wykorzystano szyfrowanie warstwy transportu, choć sam mechanizm szyfrowania został zaimplementowany w niewystarczająco bezpieczny sposób.
- **Brak zabezpieczeń binarnych (92%).** Ten defekt oznacza, że aplikacja nie stosuje żadnej formy mechanizmu obronnego utrudniającego przeprowadzenie procesu inżynierii odwrótej, złośliwe manipulowanie programem lub jego debugowanie.
- **Możliwość ataku typu injection po stronie klienta (40%).** Ta kategoria luk w zabezpieczeniach dotyczy scenariuszy, w których niezaufane dane wejściowe są przekazywane aplikacji, gdzie następnie będą przetwarzane w niebezpieczny sposób. Typowe źródła ataków typu injection obejmują na przykład inne aplikacje zainstalowane w urządzeniu oraz pochodzące z serwera dane wejściowe przekazywane do aplikacji.
- **Na stałe umieszczone w kodzie hasła lub klucze (23%).** Ten problem powstaje, gdy programista na stałe umieszcza w kodzie aplikacji pewne informacje wrażliwe, takie jak hasło lub klucz szyfrowania.
- **Wyciek poufnych danych (69%).** Ta kategoria obejmuje sytuacje, w których aplikacja przypadkowo ujawnia poufne dane w wyniku ataku typu *side-channel*, polegającego na utworzeniu dodatkowego kanału informacyjnego. W szczególności mam tutaj na myśli następujący bez wiedzy programisty wyciek danych na skutek użycia pewnego frameworka lub systemu operacyjnego.

Kluczowe czynniki problemu

Problemy związane z bezpieczeństwem aplikacji mobilnych pojawiają się z wielu różnych powodów. Z kolei luki w zabezpieczeniach zwykle występują, gdy aplikacja musi obsługiwać lub chronić poufne dane bądź też przetwarzać dane pochodzące z niezaufanych źródeł. Jednak połączenie kilku innych czynników może jeszcze bardziej spotęgować problem.

Niewystarczająca świadomość dotycząca zapewniania bezpieczeństwa

W przeciwieństwie do większości aplikacji internetowych, w których płaszczyzna ataku jest ograniczona do pochodzących od użytkownika danych wejściowych, programiści tworzący aplikacje mobilne muszą uwzględnić wiele różnych sytuacji powodujących problemy i odpowiednio chronić program przed nimi. W porównaniu do budowy innego rodzaju aplikacji, tworzenie aplikacji mobilnych jest dość unikatowe z tego względu, że nie można ufać systemowi operacyjnemu urządzenia, a nawet własnej aplikacji. Świadomość istnienia wielu płaszczyzn ataku i sposobów obrony przed nimi jest dość ograniczona i właściwie słabo rozpowszechniona w społeczności programistów tworzących tego rodzaju aplikacje. W kwestii zapewniania bezpieczeństwa mobilnego nadal mamy do czynienia z szeroko rozpowszechnioną dezorientacją i błędnym pojmowaniem podstawowych koncepcji. Jako przykład można tutaj przedstawić przekonanie wielu programistów o braku konieczności szyfrowania lub ochrony danych trwale przechowywanych w urządzeniu. Ci programiści tłumaczą to istnieniem szyfrowania standardowo zapewnianego przez wiele urządzeń. Jak się później przekonasz, to założenie jest niewłaściwe i może doprowadzić do ujawnienia poufnych danych użytkownika.

Nieustannie zmieniające się płaszczyzny ataku

Badanie poziomu bezpieczeństwa urządzeń mobilnych i aplikacji to nieustannie ewoluująca dziedzina, w której regularnie są odkrywane nowe zagrożenia i koncepcje. Szczególnie w przypadku urządzeń odkrywane są nowe luki w zabezpieczeniach, które mogą podważyć powszechnie przyjęte i stosowane w aplikacjach strategie ich ochrony. Jako przykład można tutaj podać odkrycie w urządzeniach Apple luki o nazwie „goto fail” (<https://support.apple.com/en-us/HT202934>), osłabiającej spójność tego, co wcześniej było uznawane za bezpieczny kanał komunikacji. W przypadku tej luki można było obejść nawet zalecane środki bezpieczeństwa, takie jak przypinanie certyfikatu. To doprowadziło do tego, że wielu programistów i ludzi profesjonalnie zajmujących się kwestiami zapewniania bezpieczeństwa zaczęło szukać drugiego schematu szyfrowania i implementować go w celu ochrony danych przepływających przez kanał SSL i TLS. Tego rodzaju luki w zabezpieczeniach potwierdzają, że nieustanne badania mogą wpływać na profil zagrożenia dla aplikacji lub go zmieniać, nawet w trakcie prowadzenia już prac nad nią. Zespół programistów rozpoczynający pracę nad projektem i posiadający dokładną wiedzę na temat aktualnych zagrożeń może być zmuszony do wprowadzenia zmian w kwestii zapewniania bezpieczeństwa, jeszcze zanim aplikacja zostanie ukończona i wydana.

Ograniczenia ekonomiczne i czasowe

Większość projektów programistycznych boryka się ze ściśle określonymi ograniczeniami zasobów i czasu; projekty aplikacji mobilnych nie są pod tym względem wyjątkiem. Kwestie ekonomiczne podczas tworzenia aplikacji bardzo często oznaczają, że zatrudnienie na stałe eksperta z dziedziny

bezpieczeństwa jest dla firmy po prostu niewykonalne. To dotyczy zwłaszcza mniejszych organizacji, w których testowanie pod kątem zapewniania bezpieczeństwa zwykle jest przeprowadzane na bardzo późnym etapie cyklu życiowego projektu. Nie ulega wątpliwości, że mała firma zwykle ma dużo mniejszy budżet, a tym samym charakteryzuje się mniejszą skłonnością do ponoszenia wydatków na usługi konsultingowe w zakresie zapewniania bezpieczeństwa. Wprawdzie krótki test penetracyjny prawdopodobnie pozwoli na wykrycie najprostszych luk w zabezpieczeniach, ale z reguły nie doprowadzi do wskazania znacznie subtelniejszych i skomplikowanych problemów, które można odkryć, mając więcej czasu i cierpliwości. Nawet jeśli podczas pracy nad projektem jest stale dostępny ekspert z dziedziny bezpieczeństwa, to ściśle ograniczenia czasowe mogą uniemożliwić dokładną analizę każdego wydania aplikacji. Stosowanie zwinnych (ang. *agile*) metod programowania, w których mamy do czynienia z wieloma iteracjami w krótkim okresie, może jeszcze bardziej spętłować ten problem.

Niestandardowy proces tworzenia aplikacji

Aplikacje mobilne są zwykle tworzone przez programistów danej organizacji lub zespoły zewnętrzne. W pewnych przypadkach stosowane jest połączenie obu wymienionych podejść. Ogólnie rzecz biorąc, jeśli organizacja regularnie opracowuje wiele aplikacji, doskonale przetestowane komponenty będą wielokrotnie używane w różnych projektach, co bardzo często prowadzi do promocji znacznie bardziej solidnego i bezpiecznego kodu. Jednak nawet jeśli aplikacje wykorzystują już istniejące komponenty z innych projektów, to i tak bardzo często w tych programach pojawiają się biblioteki lub frameworki, które nie zostały opracowane przez programistów budowanej aplikacji. W takim przypadku twórcy projektu mogą nie mieć pełnej i dokładnej wiedzy dotyczącej używanego kodu, co może z kolei doprowadzić do powstania problemów związanych z zapewnianiem bezpieczeństwa. Co więcej, w niektórych przypadkach biblioteki również mogą zawierać luki w zabezpieczeniach, jeśli nie zostaną dokładnie przetestowane. Przykładem może być tutaj luka `addJavaScriptInterface`, wpływająca na komponent `WebView` w systemie Android. Po jej wykorzystaniu osoba atakująca może zdalnie włamać się do urządzenia. Dalsze badania nad tą luką pozwoliły określić, że występowała ona w bibliotekach używanych w celu zapewnienia integracji z reklamami i potencjalnie mogła wpływać na ogromną liczbę aplikacji (<https://labs.mwrinfosecurity.com/blog/webview-addjavascript-interface-remote-code-execution/>).

Projekt OWASP Mobile Security

Projekt OWASP Mobile Security (https://www.owasp.org/index.php/OWASP_Mobile_Security_Project) to inicjatywa utworzona przez OWASP, czyli grupę typu non profit, która jest doskonale znana ze swojej pracy nad kwestiami związanymi z zapewnianiem bezpieczeństwa aplikacji internetowych. Z uwagi na wiele podobieństw istniejących między aplikacjami mobilnymi i internetowymi naturalne jest to, że grupa OWASP stara się promować i zwiększać świadomość w zakresie problemów dotyczących bezpieczeństwa mobilnego.

Projekt oferuje bezpłatne, scentralizowane źródło klasyfikowania ryzyka w zakresie bezpieczeństwa mobilnego. Ponadto dokumentuje działania programistyczne, które mogą zmniejszyć wpływ lub prawdopodobieństwo wykorzystania danej luki w zabezpieczeniach. Projekt jest skoncentrowany na warstwie aplikacji, a nie bezpieczeństwie platformy mobilnej. Jednak pod uwagę jest brane również ryzyko nierozzerwalnie związane z użyciem poszczególnych platform mobilnych.

10 najważniejszych zagrożeń aplikacji mobilnych według OWASP

Podobnie jak w przypadku dziesięciu największych zagrożeń OWASP, twórcy projektu Mobile Security zdefiniowali odpowiadających im dziesięć najważniejszych problemów związanych z bezpieczeństwem na platformach mobilnych. W tym zakresie projekt dość szeroko kategoryzuje pewne największe zagrożenia. Teraz pokrótce podsumuję pozycje wymienione na tej liście OWASP. Znacznie bardziej szczegółowe omówienie poszczególnych problemów i wskazówki dotyczące ich rozwiązań znajdziesz na stronie wiki projektu pod adresem https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks. Graficznie dziesięć najważniejszych problemów według OWASP pokazałem na rysunku 1.2.



Rysunek 1.2. Wymienione przez grupę OWASP dziesięć najważniejszych problemów związanych z bezpieczeństwem na platformach mobilnych

Oto krótkie omówienie dziesięciu największych zagrożeń aplikacji mobilnych według projektu OWASP Mobile Security:

- **M1: słaba kontrola po stronie serwera.** Ta kategoria ryzyka została oceniona jako najpoważniejsze zagrożenie dla aplikacji mobilnych. Wpływ tej kategorii został określony jako niezwykle ciężki, poważny błąd w kontroli po stronie serwera, który może mieć istotne konsekwencje dla biznesu. Ta kategoria obejmuje wszelkie luki w zabezpieczeniach, które mogą występować po stronie serwera, między innymi w usługach sieciowych, konfiguracjach serwerów WWW oraz w tradycyjnych aplikacjach internetowych. Umieszczenie tej kategorii na liście najpoważniejszych zagrożeń dla platformy mobilnej można uznać za kontrowersyjne, ponieważ problem nie występuje w urządzeniu mobilnym. Istnieją oddzielne projekty, które wyraźnie zajmują się zagrożeniami aplikacji internetowych. Wprawdzie zdają sobie

sprawę z wagi tego zagrożenia, ale ta kategoria nie jest dokładnie omówiona w książce, ponieważ została już doskonale udokumentowana w innych publikacjach, na przykład we wspomnianej wcześniej książce zatytułowanej *The Web Application Hacker's Handbook* (<http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118026470.html>).

- **M2: niezabezpieczony magazyn danych.** Ta kategoria ryzyka wiąże się z sytuacją, gdy aplikacja przechowuje w urządzeniu mobilnym poufne dane w formacie zwykłego tekstu lub innej postaci łatwej do odczytania przez osobę atakującą. Wpływ tej kategorii został określony jako niezwykle ciężki, poważny błąd, który zwykle może mieć poważne konsekwencje dla biznesu, na przykład doprowadzi do kradzieży tożsamości, oszustwa lub utraty reputacji. Poza umożliwieniem fizycznego dostępu do urządzenia, niebezpieczeństwo wykorzystania luki w tej kategorii wiąże się również z uzyskaniem dostępu do systemu plików, co może nastąpić poprzez użycie oprogramowania typu malware lub złamanie zabezpieczeń urządzenia w inny sposób.
- **M3: niewystarczająca ochrona warstwy transportowej.** Ta kategoria ryzyka odnosi się do ochrony ruchu sieciowego i może pojawić się w każdej sytuacji, w której dane są przekazywane w postaci zwykłego tekstu. Może dotyczyć także scenariuszy, w których ruch jest szyfrowany, choć samo szyfrowanie zostało zaimplementowane w nieprawidłowy sposób, na przykład przez dopuszczenie użycia samodzielnie podpisanych certyfikatów, przeprowadzenie niewystarczającej weryfikacji certyfikatów lub użycie niebezpiecznych pakietów szyfrowania. Tego rodzaju problemy mogą zostać wykorzystane przez osobę atakującą znajdującą się w sieci lokalnej lub sieci operatora; nie jest wymagany fizyczny dostęp do urządzenia.
- **M4: niezamierzony wyciek danych.** Tego rodzaju problem pojawia się, gdy programista przypadkowo umieszcza poufne informacje lub dane w miejscach urządzenia mobilnego, które są łatwo dostępne z poziomu innych aplikacji. Bardzo często tego rodzaju zagrożenia pojawiają się jako efekt uboczny danej platformy, przede wszystkim na skutek braku wystarczającej wiedzy programistów odnośnie sposobów przechowywania danych przez system operacyjny. Często spotykałem się z przykładami niezamierzonego wycieku danych, między innymi z powodu stosowania buforowania, migawek i dzienników zdarzeń aplikacji.
- **M5: kiepski mechanizm autoryzacji i uwierzytelniania.** Ta kategoria ryzyka jest związana z błędami popełnianymi podczas uwierzytelniania i autoryzacji, które mogą występować w aplikacji mobilnej lub w implementacji działającej po stronie serwera. Lokalne uwierzytelnianie w aplikacji mobilnej jest dość powszechne, szczególnie w aplikacjach zapewniających dostęp do poufnych danych i wymagających działania w trybie offline. W przypadku braku zastosowania odpowiednich środków bezpieczeństwa istnieje ryzyko obejścia mechanizmu uwierzytelniania i tym samym uzyskania dostępu do aplikacji. Tego rodzaju niebezpieczeństwo wiąże się również z błędami autoryzacji, które mogą występować w aplikacji działającej po stronie serwera. Skutkiem jest wówczas umożliwienie uzyskania dostępu lub wykonanie pewnych funkcji ponad uprawnieniami przydzielonymi danemu użytkownikowi.

- **M6: nieprawidłowe stosowanie kryptografii.** Powszechnie przyjętą koncepcją jest to, że aplikacja przechowująca dane w urządzeniu mobilnym powinna je szyfrować w celu zapewnienia poufności tych danych. Stosowanym w takich przypadkach rozwiązaniem jest szyfrowanie, ale ryzyko pojawia się, gdy jego implementacja zawiera pewne słabe strony. W najgorszym przypadku osoba atakująca może uzyskać fragmenty danych w postaci zwykłego tekstu lub nawet oryginalne dane w niezaszyfrowanej postaci. Najczęściej niebezpieczeństwo pojawia się na skutek kiepskiego procesu zarządzania kluczem szyfrowania. Przykładem może być tutaj umieszczenie klucza prywatnego w aplikacji, umieszczenie na stałe w kodzie klucza statycznego lub użycie klucza, który może być w niezwykle łatwy sposób znaleziony w urządzeniu — jak użycie jako klucza po prostu identyfikatora urządzenia Android.
- **M7: możliwość przeprowadzenia ataku typu injection po stronie klienta.** Ataki typu injection mogą występować, gdy aplikacja mobilna akceptuje dane wejściowe pochodzące z niezaufanych źródeł. To mogą być źródła wewnętrzne dla urządzenia, na przykład inna aplikacja, lub też zewnętrzne, takie jak komponent działający po stronie serwera. Rozważ przykład aplikacji społecznościowej pozwalającej użytkownikom na publikowanie komunikatów zawierających różne informacje o tym, co się w danej chwili dzieje z daną osobą. Aplikacja mobilna pobiera z witryny internetowej opublikowane przez innych użytkowników komunikaty, które następnie wyświetla. Jeżeli osoba atakująca będzie w stanie utworzyć fałszywy komunikat przechowywany we wspomnianej witrynie internetowej i pobrany później przez innego użytkownika aplikacji mobilnej, wówczas po umieszczeniu tego komunikatu w komponencie widoku sieciowego bądź w bazie danych działającej po stronie klienta powstaje potencjalne niebezpieczeństwo przeprowadzenia ataku typu injection.
- **M8: podejmowanie ważnych decyzji na podstawie niezaufanych danych wejściowych.** Ryzyko w tej kategorii pojawia się, gdy decyzja dotycząca bezpieczeństwa jest podejmowana na podstawie danych wejściowych pochodzących z zaufanego źródła. W większości przypadków ryzyko wiąże się z mechanizmem komunikacji międzyprocesowej (IPC). Na przykład rozważ organizację posiadającą zestaw aplikacji komunikujących się z tym samym back endem. Programista decyduje, że zamiast wymagać uwierzytelnienia użytkownika w każdej aplikacji, będą one współdzieliły jeden token sesji. Takie podejście pozwala każdej aplikacji na uzyskanie dostępu do tokenu sesji. Do współdzielenia wspomnianego tokenu używany jest mechanizm IPC, taki jak dostawca treści. Jeżeli mechanizm IPC nie zostanie prawidłowo zabezpieczony, wszelkie inne zainstalowane w urządzeniu aplikacje o złośliwym działaniu mogą potencjalnie użyć interfejsu IPC w celu pobrania tokenu sesji i uzyskania dostępu do sesji użytkownika.
- **M9: nieprawidłowa obsługa sesji.** Zarządzanie sesją to ważna koncepcja podczas tworzenia oprogramowania. Sesja to mechanizm, który jest przez serwer używany do przechowywania informacji o stanie podczas pracy z protokołem bezstanowym, takim jak HTTP lub SOAP. Ryzyko tej kategorii obejmuje wszelkie luki w zabezpieczeniach powodujące ujawnienie tokenu sesji osobie atakującej. Pod pewnymi względami nakłada się na koncepcje przedstawione w punkcie „A2. Nieprawidłowe uwierzytelnianie i zarządzanie sesją” na liście¹ dziesięciu największych niebezpieczeństw dla aplikacji internetowych.

¹ Tę listę znajdziesz na stronie wiki pod adresem https://www.owasp.org/index.php/Top_10_2013-Top_10 – przyp. tłum.

- **M10: brak zabezpieczeń binarnych.** Ta kategoria ryzyka powoduje osłabienie mechanizmów obronnych, które programista może — i w większości przypadków powinien — wbudować w aplikację mobilną. Zabezpieczenia binarne z reguły spowalniają osobę atakującą, która próbuje analizować lub modyfikować kod binarny aplikacji albo przeprowadzić proces inżynierii odwrotnej.

Powyższa lista to niewątpliwie użyteczny zasób informacji o pojawiających się typach luk w zabezpieczeniach, które mogą wystąpić w aplikacjach mobilnych. Można się spodziewać, że wraz z postępującym coraz lepszym zabezpieczaniem aplikacji mobilnych będzie uzupełniana o nowo odkryte zagrożenia. Ponadto będzie odgrywała jeszcze większą rolę w edukacji programistów i profesjonalistów zajmujących się kwestiami zapewniania bezpieczeństwa.

Narzędzia rekomendowane przez OWASP Mobile Security

Narzędzia stanowią ważny element arsenału każdego profesjonalisty zajmującego się bezpieczeństwem, niezależnie od tego, czy są przeznaczone po prostu do wspomaganie ręcznego przeprowadzania analizy, dostarczania frameworka do opracowania innych narzędzi czy pracy w charakterze zasobu zapewniającego możliwość wzmocnienia możliwości programistów. Projekt OWASP Mobile Security oferuje wiele narzędzi typu open source, które można wykorzystać podczas zapewniania bezpieczeństwa aplikacji. Informacje o nich znajdziesz na stronie wiki pod adresem https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Mobile_Tools. Te narzędzia mogą okazać się użyteczne w trakcie poznawania tematu zapewniania bezpieczeństwa aplikacji mobilnych. Poniżej przedstawiłem krótkie omówienie wspomnianych narzędzi:

- **iMAS** (https://www.owasp.org/index.php/OWASP_iMAS_iOS_Mobile_Application_Security_Project). Ten projekt utworzony przez MITRE Corporation jest rozprowadzany jako *open source* frameworkiem bezpiecznej aplikacji dla systemu iOS. Stanowi idealny zasób dla programistów lub profesjonalistów zajmujących się zapewnianiem bezpieczeństwa, którzy chcą poznać lub zrozumieć implementację mechanizmów bezpieczeństwa na platformie iOS. Celem tego projektu jest zademonstrowanie i dostarczenie implementacji chroniących aplikacje iOS oraz dane w znacznie większym stopniu niż za pomocą modelu bezpieczeństwa oferowanego przez Apple. W efekcie powinno nastąpić wyraźne zmniejszenie ryzyka, że osoba atakująca będzie w stanie przeprowadzić proces inżynierii odwrotnej aplikacji, manipulować nią lub wykorzystać ewentualne luki w jej zabezpieczeniach. Aby osiągnąć postawiony sobie cel, w ramach projektu powstało wiele implementacji open source zapewniających rozwiązania w wielu obszarach, w których mogą występować luki w zabezpieczeniach, czyli między innymi w zakresie kodów PIN w aplikacji, wykrycia jailbreakingu urządzenia, ochrony przez debugowaniem i modyfikowaniem środowiska uruchomieniowego aplikacji. Choć niektórymi z wymienionych tematów dokładnie zajmiemy się w rozdziałach 2. i 3., projekt iMAS jest niewątpliwie użytecznym zasobem pomagającym w poznaniu technik defensywnych, a także może służyć w charakterze przewodnika dla programistów.

- **GoatDroid** (https://www.owasp.org/index.php/Projects/OWASP_GoatDroid_Project). Projekt GoatDroid, opracowany przez Jacka Mannino i Kena Johnsona, dostarcza samodzielne środowisko testowe dla aplikacji Android. To środowisko oferuje dwie przykładowe implementacje pomagające w doprowadzeniu do perfekcji własnych umiejętności. Pierwsza to FourGoats, czyli oparta na danych geolokalizacji sieć społecznościowa. Druga to Herd Financial, czyli fikcyjna aplikacja bankowości internetowej. Razem te dwa projekty zapewniają szerokie pokrycie większości niebezpieczeństw wymienionych wcześniej na liście OWASP Top 10 i stanowią dobry punkt wyjścia dla osób stawiających pierwsze kroki na obszarze zapewniania bezpieczeństwa aplikacji Android.
- **iGoat** (https://www.owasp.org/index.php/OWASP_iGoat_Tool_Project). Podobnie jak w przypadku projektu GoatDroid, iGoat to aplikacja treningowa pozwalająca Ci na udoskonalenie własnych umiejętności w zakresie oceny i analizy programów przeznaczonych dla platformy iOS. Ten projekt typu open source został opracowany przez Kena van Wyka, Jonathana Cartera i Seana Eidermillera. Kod źródłowy znajdziesz na stronie <https://code.google.com/archive/p/owasp-igoat/>. Dostarczony jest zarówno serwer, jak i aplikacja klienta wraz z wieloma ćwiczeniami dotyczącymi ważnych tematów, na przykład lokalnego magazynu danych, pęku kluczy, SQL injection itd.
- **Damn Vulnerable iOS** (https://www.owasp.org/index.php/OWASP_DVIA). Ten utworzony przez Prateeka Gianchandaniego projekt dostarcza kolejną zawierającą luki w zabezpieczeniach aplikację przeznaczoną do celów testowych. W połączeniu z projektem iGoat obie aplikacje zapewniają dość dobre pokrycie niebezpieczeństw wymienionych wcześniej na liście OWASP Top 10. Aplikacja składa się z wielu zadań przeznaczonych do wykonania, co pozwala na jeszcze dokładniejsze poznanie i zrozumienie tematów z zakresu bezpieczeństwa na platformie iOS. Uwzględnione zostały tutaj tematy pominięte w iGoat, na przykład wykrycie poddania urzędnika jailbreakingowi, manipulacje środowiskiem uruchomieniowym aplikacji, modyfikacje plików binarnych aplikacji, a także zagadnienia związane z kryptografią.
- **MobiSec** (https://www.owasp.org/index.php/OWASP_Mobile_Security_Project). To jest utworzone przez Tony'ego DeLaGrange'a i Kevina Johnsona środowisko typu live, przeznaczone do przeprowadzania testów penetracyjnych aplikacji mobilnych. Celem przyświecającym temu projektowi jest dostarczenie pojedynczego zasobu oraz utrzymywanie najnowszych wersji wszystkich narzędzi, które mogą okazać się przydatne podczas oceny i analizy aplikacji mobilnych. W ramach tego projektu otrzymujemy rozwiązanie podobne, jak stosowane w przypadku innych dystrybucji typu live, na przykład w popularnej Kali Linux. Jednak projekt MobiSec jest zorientowany na zapewnianie bezpieczeństwa na platformach mobilnych.
- **Androick** (https://www.owasp.org/index.php/Projects/OWASP_Androick_Project). W odróżnieniu od pozostałych projektów, ten dotyczy nieco innego tematu i jest skoncentrowany na zautomatyzowanych zadaniach analizy śledczej aplikacji na platformie Android, a nie na testach penetracyjnych lub samodzielnym udoskonalaniu umiejętności. Ten projekt, utworzony przez Floriana Pradinesa, automatyzuje zdobywanie kluczowych informacji i danych, jak pliki w formacie APK (ang. *android package*), dane aplikacji, bazy danych i dzienniki zdarzeń z urządzenia.

Oczywiście w trakcie swojej działalności na polu zapewniania bezpieczeństwa aplikacjom mobilnym napotkasz również inne, niewymienione tutaj narzędzia i nawet będziesz zmuszony do ich użycia. O wielu z nich wspomnę w dalszych rozdziałach. Jednak projekty przygotowane i zalecane przez OWASP są szczególnie użyteczne podczas doskonalenia własnych umiejętności, ponieważ zostały dobrze udokumentowane, są dostępne jako oprogramowanie open source oraz zostały opracowane specjalnie w celu pokrycia zagrożeń wymienionych na liście Top 10. Dlatego też gorąco zalecam zapoznanie się z tymi projektami i potraktowanie ich przez początkujących jako punktu wyjścia do dalszych eksperymentów.

Przyszłość dziedziny zapewniania bezpieczeństwa aplikacjom mobilnym

W ciągu minionych pięciu lat smartfony i aplikacje mobilne zyskały wielką popularność wśród użytkowników. Obecnie nie ma absolutnie żadnych znaków wskazujących na zahamowanie tego procesu i można oczekiwać, że ten trend będzie kontynuowany w przyszłości. Konsekwencją tej rosnącej rewolucji mobilnej jest coraz większy nacisk na poznanie i zrozumienie zagrożeń pojawiających się podczas wdrażania aplikacji mobilnych, a także opracowanie znacznie efektywniejszych sposobów ich pokonywania. Jestem przekonany, że obecne zagrożenia na polu bezpieczeństwa mobilnego nie są dobrze rozumiane, zwłaszcza w społeczności programistów.

Dlatego też można się spodziewać, że klasyczne luki w zabezpieczeniach, takie jak niezabezpieczony magazyn danych i niewystarczający poziom bezpieczeństwa warstwy transportowej, nadal będą dominowały w najbliższej przyszłości.

Z tego względu zapewnianie bezpieczeństwa na platformach mobilnych to nieustannie ewoluująca dziedzina i możemy się spodziewać pojawienia się nowych kategorii ataków wraz z kolejnymi zmianami w technologiach mobilnych. Wprowadzanie nowych komponentów sprzętowych, takich jak czytniki linii papilarnych, oraz coraz szersze wykorzystywanie istniejących technologii, na przykład NFC, bez wątpienia doprowadzi do odkrycia nowych luk w zabezpieczeniach. Szczególnie będzie to dotyczyć wdrożeń w środowiskach takich jak przetwarzanie płatności mobilnych, na przykład używanych przez technologie Google Wallet i Apple Pay.

Podobnie jak w przypadku innych obszarów oprogramowania, w szczególności używanych do obsługi transakcji finansowych, przestępcy będą szukali możliwości wykorzystania luk w zabezpieczeniach i łatwego zdobycia pieniędzy. Jesteśmy już świadkami wzrostu ilości oprogramowania typu malware atakującego aplikacje przeznaczone do obsługi bankowości internetowej oraz oszustw związanych z wiadomościami SMS typu premium. Należy spodziewać się kontynuacji tego trendu w przyszłości. Wspomniany wzrost doprowadził już do pewnych zmian w krajobrazie zagrożeń. W odpowiedzi na ten wzrost niektórzy programiści zaczęli stosować zabezpieczenia binarne, aby móc chronić aplikacje przed niebezpieczeństwami. Coraz większa świadomość czytających niebezpieczeństw prawdopodobnie doprowadzi do szerszego stosowania tego rodzaju zabezpieczeń oraz użycia technologii takich jak dwupoziomowe uwierzytelnianie (ang. *two-factor authentication*).

Istnieje również prawdopodobieństwo dalszej ewolucji aplikacji mobilnych niezależnych od platformy sprzętowej, ponieważ programiści dążą do zmniejszenia fragmentacji wśród różnych platform mobilnych. To można dostrzec we wzroście dwóch wymienionych poniżej trendów w programowaniu:

- **Aplikacje oparte na przeglądarce WWW.** To pojęcie oznacza aplikacje, które są zwykle „przyjaznym urządzeniom mobilnym” klonem głównej witryny internetowej, wczytywanym poprzez przeglądarkę WWW urządzenia.
- **Aplikacje hybrydowe.** To pojęcie odwołuje się do aplikacji mobilnych będących natywnymi opakowaniami dla widoku sieciowego i bardzo często używających frameworka w celu uzyskania dostępu do natywnych funkcji urządzenia.

Uzupełnieniem dla wspomnianych trendów jest opracowanie ogromnej liczby zarówno komercyjnych, jak i dostępnych bezpłatnie frameworków, z których każdy ma swoje dziwactwa i niuanse prowadzące do powstania różnorodnych luk w zabezpieczeniach. Jak w przypadku większości zmian w technologii, te trendy przyniosły nowe rodzaje ataków oraz kolejne warianty istniejących. Związane z tym implikacje w obszarze zapewniania bezpieczeństwa przedstawię w rozdziale 14.

Pomimo wszystkich zmian w aplikacjach mobilnych nie ma żadnego znaku, że klasyczne ataki mogą zaniknąć. Jednak za pozytywny objaw można uznać coraz większą świadomość zagrożeń na polu bezpieczeństwa mobilnego i istniejących tutaj luk w zabezpieczeniach, co zawdzięczamy dokładnej dokumentacji, klasyfikacji oraz projektom takim jak opracowane przez grupę OWASP. Jestem przekonany, że poprzez wymienione i podobne projekty świadomość w zakresie bezpieczeństwa mobilnego będzie się zwiększała i pomoże w podjęciu przez programistów działań prowadzących do zmniejszenia liczby luk w zabezpieczeniach aplikacji mobilnych.

Podsumowanie

W ciągu minionych pięciu lat niezwykła popularność nowoczesnych smartfonów doprowadziła do gwałtownego skoku w dziedzinie tworzenia aplikacji mobilnych przez firmy zewnętrzne. Usprawnienia wprowadzone w smartfonach pomogły w szybkiej ewolucji aplikacji od prostych do oferujących potężne możliwości i zintegrowanych z wieloma technologiami internetowymi. Podczas tej ewolucji wiele aspektów technicznych, ekonomicznych i związanych z programowaniem miało wpływ na stosowanie kiepskiego poziomu zabezpieczeń, co można dostrzec w licznych obecnych aplikacjach mobilnych.

Poza tradycyjnymi problemami z zakresu bezpieczeństwa (dane wejściowe) wpływającymi na wszystkie rodzaje aplikacji, w aplikacjach mobilnych pojawiło się także wiele względnie unikatowych luk w zabezpieczeniach, co wynika z natury tych aplikacji. Te kwestie bardzo często nie są zbyt dobrze rozumiane przez programistów, co może prowadzić do ataków, gdy urządzenie jest używane w niezaufanej sieci, zostanie zgubione lub skradzione. Może nawet dojść do ataków przeprowadzanych przez inne komponenty platformy mobilnej.

Badania nad obecnym stanem bezpieczeństwa mobilnego pokazały, że luki w zabezpieczeniach aplikacji nie są dobrze rozumiane, a większość aplikacji pozostaje podatna na ataki. Co więcej, ewolucja nowych technologii oraz kolejne integracje prawdopodobnie doprowadzą do powstania zupełnie nowych kategorii ataków, które będą mogły narazić organizacje na poważne niebezpieczeństwo, o ile w porę nie zostaną podjęte odpowiednie działania.



Skorowidz

.NET Framework, 30, 607, 621

A

activity manager service,

Patrz: menedżer czynności

address space layout

randomization, *Patrz:* ASLR

adres

IP, 339

loopback, 328

URI, 334, 371, 444

uprawnienia, 449, 450

URL, 142, 144, 521, 549, 550,
661

system wczytywania, 97, 98

URL MPNS, 565, 566

akcelerator kryptograficzny

sprzętowy, 71

algorytm

AES, 326, 618

AES-128, 606, 636

CBC, 472, 639

AES-256, 606, 636, 637

CBC, 472, 636, 637

EBC, 636

AES-XTS, 172

DES, 617, 636

hash, 446

PBKDF2, 70, 238, 448, 613, 636

RC4, 606, 619

SHA-256, 172, 615

TEA, 619

wektor inicjalizacji, 620

wymiany kluczy, 102

XOR, 619

Allegra Nicholas, 248

Allitori, 461

Amazon Appstore, 30, 202

analiza

poziomu zabezpieczenia

aplikacji, *Patrz:* aplikacja

analiza poziomu

zabezpieczenia

śledcza, 41

analizator składni, 375

Androick, 41

Android, 191

aplikacja, *Patrz:* aplikacja

Android

czynność, 214

dostawca treści, 215

działanie systemu, 197

emulator, 192, 193, 195

alternatywy, 196

ograniczenia, 195

jądro, 248

moduł binder, 213, 216, 237

komponent, 214, 215, 216

komunikacja, 216

mechanizm obronny, 239,

241

użytkownik root, 242, 244,

245

model zapewniania

bezpieczeństwa, 266

odbiorca komunikatów, 215

odzyskiwanie, 249

pakiet, 199

proces, 197

przeglądarka WWW, 239,

453

słownik, 441

usługa, 214

użytkownik, 198

identyfikator, *Patrz:* UID

wersja, 260, 456

zasoby, 203

Android Developer Tools, 192

Android Device Manager, 201

Android ID, 447

Android Keystore System, 449

Android Market, 30

Android NDK, *Patrz:* NDK

Android SDK, *Patrz:* SDK

Android Software Development

Kit, *Patrz:* SDK

- API
- AesManaged, 637
 - CFNetwork, 99, 101
 - ClearCookieAsync, 648
 - ClearInternetCacheAsync, 649
 - Cydia Substrate C, 121, 122
 - Data Protection, 48, 70, 71, 108, 161, 600, 621, 622
 - Framework Carbon, 97
 - HTTP, 524, 525, 536
 - HttpNotificationChannel, 563
 - HttpUtility.HtmlEncode, 647
 - IsolatedStorageSettings, 596
 - Java Reflection, 208, 373
 - JavaScript, 134, 135
 - JavaScript to Objective-C, 51
 - LINQ to SQL, 574, 575, 603, 605
 - PhoneGap, 659, 660
 - Rfc2898DeriveBytes, 637
 - RNGCryptoServiceProvider, 607, 640
 - Secure Transport, 97, 98, 101
 - Shell_PostMessageToast, 557, 558
 - ShellToast, 557
 - SQLiteCipher, 575
 - SSL, 540
 - Substrate, 121, 122, 123, 126
 - funkcje, 123
 - system wczytywania adresu
 - URL, 97, 98
 - System.Random, 607, 608, 609, 610, 615, 636, 640
 - Trust Services, 99
 - UIWebView, 148
 - Win32, 473
 - XDocument, 566, 571
- ApkProtect, 461
- aplikacja
- lPassword, 146
 - analiza dynamiczna, 110
 - analiza poziomu
 - zabezpieczenia, 493, 494, 495, 496, 497, 499, 500, 501, 502, 505, 506, 512, 514, 515, 516, 522, 533, 595, 596, 599, 600, 602, 603, 606, 635
 - Android, 197, 199
 - instalacja, 201, 202, 203, 218, 219, 221
 - komunikacja, 213
 - uprawnienia, 220, 229, 231, 232, 237
 - uruchamianie, 221
 - bankowości internetowej, 41
 - blokowanie debugera, 188, 349, 350, 351, 352, 354, 455, 463
 - bufor, 161, 170
 - BYOD, *Patrz:* BYOD
 - Citygroup, 70
 - czynność, *Patrz:* czynność
 - deszyfrowanie, *Patrz:* deszyfrowanie aplikacji
 - dystrybucja, 51, 52
 - działająca w tle, 161, 162, 641
 - etykieta, 462
 - Find and Call, 157
 - GBA4iOS, 52
 - granice zaufania, 288, 442
 - hosta, 145, 146
 - HR, 31
 - hybrydowa, 43, 50, 51, 483, 484, 653
 - internetowa, 31
 - iOS
 - atak, *Patrz:* atak na aplikację iOS
 - deasemblacja, 93
 - dekompilacja, 93
 - dystrybucja, 51, 52
 - instalacja, 53
 - katalog, 53
 - modyfikacja, 65, 111
 - struktura, 52
 - uprawnienia, 54
 - Kaseya Browser, 137
 - Kaseya Secure Browser, 132, 133
 - Kaseya Secure Docs, 132
 - kategoria, 31
 - kod binarny, 34, 40
 - komponent, 438, 439, *Patrz też:* komponent
 - eksportowanie, 439
 - kopia zapasowa, 347
 - korporacyjna, 31, 52
 - atak, 132
 - dystrybucja, 52
 - łamanie, 111
 - magazynu dokumentów, 31
 - MDM, *Patrz:* MDM
 - mobilna
 - Android, 30
 - bezpieczeństwo, *Patrz:* bezpieczeństwo
 - BlackBerry, 30
 - ewolucja, 30
 - iOS, 30
 - kategoria, 31
 - komunikacja sieciowa, 95, 96
 - plaszczyna ataku, *Patrz:* atak
 - Windows Phone, *Patrz:* Windows Phone, WP7, WP8
 - zagrożenia, 37
 - zalety, 32
 - Native Access Webservice, 514
 - natywna, 50, 51, 457, 458, *Patrz też:* kod natywny
 - mechanizm obronny, 458
 - zabezpieczenie, 457
 - zaciemnianie, 190, 461
 - niezależna od platformy, 653, 654, 655, 656, 657, 658
 - pomost natywny, *Patrz:* pomost natywny
 - wady, 654
 - zalety, 654
 - okno, 130
 - oparta na przeglądarce
 - WWW, 43, 50, 51
 - ostatnio używana, 285, 440
 - Password Manager Free, 130
 - Path, 54
 - Polaris Viewer, 401
 - projektowanie, 170
 - prywatny katalog danych, 445, 449, 455
 - przeznaczona do wewnętrznego użycia, 491

- punkt wejścia, 171, 437, 439, 459, 519, 520, 524, 527, 530, 532, 558, 635
 - removableStorage, 472
 - rozszerzenie, 145, 146
 - Safe Password Free, 131
 - SamWP8 Tools, 505
 - Shazam, 300
 - Sieve, 276, 288, 325, 341
 - manifest, 277
 - społecznościowa, 39
 - sprawdzanie legalności zakupu, 420
 - suma kontrolna, 187
 - szkoleniowa, 275, 276
 - tożsamość, 223
 - uaktualnianie, 226, 415
 - UniversalMDMClient, 417
 - uprawnienia, 386, 438, 439, 470
 - DISABLE_KEYGUARD, 390, 391
 - INSTALL_PACKAGES, 232, 233, 234, 373, 398, 401, 438
 - RECORD_AUDIO, 426
 - wymuszone, 440
 - usług wewnętrznych, 31
 - VoIP, 143, 487, 550, 551
 - w trybie programisty, 492
 - wewnętrznych
 - komunikatorów, 31
 - Windows Phone, 593
 - WP
 - gra, 483, 484
 - implementacja, 593
 - instalowanie, 482
 - katalog, 489
 - rozpowszechnianie, 489, 491
 - standardowa, 483
 - WP8, 470
 - wykrywanie modyfikacji, 187, 463
 - XSS, 148
 - zgodność wsteczna, 456
 - App Store, 30, 45, 51
 - aplikacja, *Patrz:* aplikacja
 - recenzja aplikacji, 51, 57
 - uwierzytelnianie, 77
 - zgłoszenie, 47
 - AppContainer, 469
 - Apple Developer Enterprise, 52
 - Apple Developer Program, 51
 - apple system log, *Patrz:* konsola
 - ASL
 - AppSync, 57, 67
 - ARC, 84
 - architektura
 - arm64, 81, 82
 - armv6, 82
 - armv7, 81, 111
 - armv7s, 81
 - armv8, 80
 - MIPS, 261
 - x86, 261
 - ARM, 111, 204
 - arm64, 80
 - assembler ARM, *Patrz:* ARM
 - ASLR, 49, 82, 346, 456, 473, 476, 477, 517
 - atak, 33
 - billion laughs, 151, 569, 570
 - blokady usług, *Patrz:* atak
 - DoS
 - brute force, 128, 131, 132, 172, 306, 447
 - na hasło, 616, 617
 - Cross-Application
 - Navigation Forgery, 558
 - DoS, 151, 569, 570
 - dostęp fizyczny, 387, 435, 602
 - injection, 39, 147, 176, 422, 645, *Patrz też:* atak SQL
 - injection
 - na procedurę obsługi pliku, 153
 - wykorzystujący encje XML, 152
 - man in the middle, 96, 107, 373, 377, 380, 381, 385, 412, 418, 539, 543, 656
 - między aplikacjami, 661
 - na analizator składni XML, 151, 152
 - na aplikację iOS, 95
 - na bazę danych, 574
 - na certyfikat, 451
 - na implementację AIDL, 308
 - na kod natywne, 340, 341, 345
 - na odbiorcę komunikatów, 310
 - na procedurę obsługi pliku, 579
 - na rozszerzenia aplikacji, 146
 - na usługę, 302, 303
 - phishing, *Patrz:* phishing
 - polegający na poruszaniu się po katalogu, 443
 - powodujący obniżenie poziomu ochrony uprawnień, 269
 - skryptu lokalnego, 543, 544, 547
 - SQL injection, 146, 150, 151, 176, 292, 293, 295, 443, 552, 578, 642, *Patrz też:* atak
 - injection
 - blokowanie, 575
 - tapjacking, *Patrz:* tapjacking
 - wstrzykiwanie fragmentów, 441
 - wykorzystujący tajny kod, 318
 - XSS, 96, 148, 149, 150, 177, 541, 542, 645, 656, 660, 661
 - kodowanie metaznaków, 178
 - XXE, 569, 571
 - zdalny, 398
 - automatic reference counting, *Patrz:* ARC
 - autoryzacja, 38, *Patrz też:*
 - uwierzytelnianie
 - Auty Michael, 295
- ## B
- baza danych
 - dostęp, 574
 - hasła, 298
 - lokalna, 603, 605
 - na karcie SD, 325, 326
 - natywna, 639
 - SQL, 292
 - SQLite, 150, 151, 443, 575, 607, *Patrz:* SQLite
 - atak SQL injection, 642
 - SQLCipher, 639

- baza danych
 SSE, 639
 szyfrowanie, 173
 szyfrowanie, 173, 606, 639
- Bedrune Jean-Baptiste, 77
- Bergman Neil, 335
- bezpieczeństwo, 32, 33, 34, 35, 42, 82, 169, 170, 174, 439
 biblioteka, 36, 171
 dostawców treści, 289, 290, 292
 iOS, 45, 56, 57
 jailbreaking, *Patrz:* jailbreak
 język natywny, 46
 łańcuch procedur
 bezpiecznego rozruchu, 46
 piaskownica, 46, 48, *Patrz też:* piaskownica
 podpisywanie kodu, 46, 47
 szyfrowanie danych, 46, 48
- klasyfikowanie ryzyka, 36
- komunikacji, 327, 451
- luka, *Patrz:* luka
- narzędzia, 40, 41, 42
- nowe zagrożenia, 35
- ograniczenia ekonomiczne, 35
- poziom ochrony, 232, 385, 386, 440
 dangerous, 233
 development, 233
 normal, 269
 normal, 233
 obniżenie, 459
 signature, 233, 234, 269, 311, 460
 signatureOrSystem, 233, 234
 system, 233
 szyfrowanie pliku, 71
- Windows Phone 8, 469
- zabezpieczenia binarne, *Patrz:* zabezpieczenia binarne
- biblioteka
 bezpieczeństwo, 36, 171
 Common Crypto, 173
 dynamiczna, 121, 138, 183
 wczytanie, 123
 libdecrypt.so, 341
- libencrypt.so, 341
 mdsectweak.dylib, 121, 122
 natywna, 239, 341, 641
 substrate, 121
 tweaks, 121
- Binder, 214
- BlackBerry World, 30
- Bluetooth, 171, 438, 522, 523
- Borgaonkar Ravi, 318
- Boschert Bas, 326
- Bradberry Daniel, 276
- Brama domyślna, 412
- broadcast receiver,
Patrz: Android odbiorca komunikatów
- bufor
 aplikacji, *Patrz:* aplikacja bufor
 ARP, 412
 zatrucie, 265, 332, 412
 przeglądarki, *Patrz:* przeglądarka bufor ramki, 428
- buforowanie
 klawiatury, 162
 odpowiedzi HTTP, 163
- BYOD, 132
- ## C
- Carter Jonathan, 41
- Caunt David, 457
- CC Tools, 62, 63
- certificate pinning, *Patrz:* certyfikat przypinanie
- certyfikat, 35, 47, 73, 223
 akceptacja, 97, 99
 Certification Authority, Microsoft Marketplace CA, 468
 fałszywy, 175, 226, 451
 główny, 99
 instalowanie, 104, 329
 przypięcie, 361
 przypinanie, 97, 103, 106, 174, 329, 355, 356, 451
 SSL, 361, 644, 645
 przypinanie, 452
 TSL, 644, 645
- urząd certyfikacji, 226, 330, 451
 weryfikacja, 97, 98, 99, 101, 226, 541
 niewystarczająca, 38
 własna procedura, 99
 wyłączenie, 107, 539, 540
 Windows Phone 8, 644
 wygasły, 99
 X.509, 223, 452
- chamber, *Patrz:* komora ciąg zaborzący, 446
 ciąg zaborzący, 172, 448
 cipher suite, *Patrz:* zestaw szyfrowy
- Citygroup, 70
- CMAS, 375
- Cocoa Touch, 50
- Comex, 58
- Corona, 59
- Cycript, 128, 130, 131, 132
- Cydia, 57, 61, 62, 70
- Cydia Substrate, 58, 110, 121, 122, 183
- czujnik ruchu, 659
- czynność, 214, 267, 279
 alias, 281
 BROWSABLE, 371, 417
 wyszukiwanie, 372
 eksport, 279, 280, 281, 284
 logowania, 279
 niewyeksportowana, 282
 Package Installer, 420
 wyszukiwanie, 370
- czytnik
 kodu kreskowego, 527
 Touch ID, 75, 77, 78, 107, 170
- ## D
- dalvik virtual machine, *Patrz:* DVM
- Damn Vulnerable iOS, 41
- dane
 aplikacji, *Patrz:* aplikacja prywatny katalog danych
 dostęp, 54
 dostępność, 170
 hermetyzacja, 118

- integralność, 644
 - kategoria, 54
 - kradzież, 128
 - losowe, 607, 615
 - magazyn, *Patrz:* magazyn danych
 - mapowanie losowe, 49
 - ochrona dostępu, 170
 - ochrona w aplikacji, 169
 - ochrona w trakcie transportu, 174
 - osobowe, 155, 600
 - otrzymane przez Bluetooth, 524
 - poufne, 326, 533, 600, 614, 636
 - szyfrowanie, 611
 - usuwanie, 640
 - prywatne, 156
 - przechowywanie, 107, 636
 - pseudolosowe, 607
 - rejestracja, 326
 - uwierzytelniające, 73, 326, 596
 - do konta Google, 395
 - wejściowe, 147, 171, 172, 176, 177, 520, 527, 530, 532
 - niezaufane, 39
 - sprawdzanie, 437, 438, 439
 - weryfikacja, 635
 - wrażliwe, 73, 107, 238
 - wyciek, 158, 162, 597
 - schowek systemowy, 159
 - w dziennikach zdarzeń, 159
 - wyciek niezamierzony, 38
 - DashO, 461
 - data execution protection, *Patrz:* DEP
 - deasemblacja
 - aplikacji
 - iOS, *Patrz:* aplikacji iOS
 - deasemblacja
 - deassembler, 112, 261
 - Hopper, 112
 - IDA, 255
 - debuger, 65, 188, 345, 349, *Patrz też:* narzędzie JIT, 345, 346
 - debugowanie USB, 387, 388
 - dekompilacja, 261
 - aplikacji
 - iOS, *Patrz:* aplikacji iOS
 - dekompilacja
 - DeLaGrange Tony, 41
 - demon
 - adbd, 203
 - installd, 53
 - lockdownd, 60
 - multipleksowania, 61
 - usbmuxb, 53, *Patrz też:* usługa usbmuxd
 - denial of service, *Patrz:* atak DoS
 - DEP, 49, 478, 517
 - deszyfrowanie
 - aplikacji, 85
 - automatyzacja, 87
 - device firmware upgrade, *Patrz:* DFU
 - DexProtector, 461
 - DFU, 59
 - Dhanjani Nitesh, 145
 - digital rights management, *Patrz:* DRM
 - dokument DTD, 568, 569, 647
 - encja, 569
 - dokument XML, 567, 568, 569
 - domena
 - najwyższego poziomu
 - odwrócona, 74
 - dopasowanie
 - wzorca, 444
 - dopasowanie wzorca, 300, 301
 - dostawca
 - treści, 215, 289, 449
 - eksportowanie, 267, 442, 456
 - mapowanie na interfejs przeglądarki WWW, 295
 - niechroniony, 289, 290, 292
 - oparty na pliku, 298
 - Dowd Mark, 399
 - drive-by download, *Patrz:* technika automatycznego pobierania pliku
 - DRM, 85
 - DroidWall, 322
 - DVM, 222, 259
 - dyld, 49
 - dynamic linker, *Patrz:* dyld
 - DYNAMICBASE, 650
 - dyrektywa
 - %ctor, 127
 - %end, 127
 - %hook, 127
 - %log, 127
 - %orig, 127
 - preprocesora, 127
 - dziennik zdarzeń, 127, 159, 204, 326, 597
 - dane poufne, 326
 - hasło do bazy danych, 327
- ## E
- ECB szyfrowanie, 446
 - Eidermiller Sean, 41
 - ekran, 30, 32
 - blokada, 109, 130, 137, 387, 390, 394, 395
 - kod zabezpieczający, 130, 131, 137
 - wzorzec, 392, 432
 - zerowanie, 395, 396
 - nagranie wideo, 428
 - rozdzielczość, 32
 - zrzut, 161, 286, 428
 - Elenkow Nikołaj, 226
 - Erasmus Tyrone, 295, 322
 - Esser Stefan, 49
 - execute never, *Patrz:* funkcja XN
- ## F
- Facebook, 486
 - FairPlay, 85
 - fat binary, 79, 80, 81, 82, 111
 - F-Droid, 202
 - filtr intencji, *Patrz:* intencja filtr format
 - .aidl, 200, 303
 - .apk, 199
 - .class, 200
 - .doc, 148
 - .ext, 528
 - .java, 200

format
 .keynote, 148
 .numbers, 148
 .pages, 148
 .plist, 69
 .ppt, 148
 .xls, 148
 3GP, 427
 APK, 41
 HTML, 147, 177
 Mach-O, 79, 80, 81, 83
 OAT, 262
 ODEX, 259
 PDF, 147, 370, 401, 449, 450
 RTF, 147
 smali, 356
 TLD, 74
 XML, 151, 177
 analizator składni, 151
 dokument DTD, 647
 Forristal Jeff, 425
 fotmat TAR, 348
 FourGoats, 41
 fragment, 286
 biała lista, 441
 wstrzykiwanie, 441
 framework
 AddressBook, 157
 AdSupport, 156
 Apache Cordova, 659, 660
 Appcelerator, 655
 Carbon, 98
 Cocoa Touch, *Patrz:* Cocoa Touch
 Cordova, 662, 663
 Core Location, 157, 158
 CoreGraphics, 58
 Corona SDK, 655
 Cydia Substrate, *Patrz:* Cydia Substrate
 LINQ, 566, 567, 568
 LocalAuthentication, 78
 manipulacji, 110
 MobileKeyBag, 77
 PhoneGap, 655, 659, 660
 biała lista domen, 662, 663
 Security, 173
 Splunk MINT Express, 599

WPClogger, 599
 Xamarin, 655
 Freeman Jay, 247, 352, 360
 Frida, 134, 137, 183
 funkcja
 class_replaceMethod, 138
 dladdr, 183
 enumerate_devices, 135
 fork, 182, 185, 186
 generowania klucza,
 Patrz: PBKDF2
 metadane, 91
 MISValidateSignatureAnd
 ↳ CopyInfo, 67
 MSFindSymbol, 123, 124
 MSGetImageByName, 123
 MSHookFunction, 123, 124
 MSHookMessageEx, 123
 objc_msgSend, 120
 PBKDF2, *Patrz:* algorytm
 PBKDF2
 ptrace, 188
 rejestracji danych, 457
 SecRandomCopyBytes, 173
 setResult, 282
 sqlite3_bind_text, 176
 sqlite3_prepare, 176
 sysctl, 188
 wywołanie dynamiczne, 89
 XN, 49

G

gadżet ROP, 478
 Game Boy Advanced, 52
 generator
 liczb losowych, 173, 607, 615, 640
 liczb pseudolosowych, 607
 załączek, 447, 607, 608, 609
 geohot, 60
 geolokalizacja, 157, 158, 428, 659
 GetJar, 202
 Gianchandani Prateek, 41
 Gingerbreak, 245
 gniazdo, 337
 @jdpw-control, 349
 Bluetooth, 523
 sieciowe, 525

GoatDroid, 41
 Google Cloud Messaging, 201
 Google Play, 30
 Google Play Store, 191, 197
 bezpieczeństwo, 201
 ważność certyfikatu, 226
 GPS, 30, 32, 157, 428, *Patrz też:*
 geolokalizacja
 GUID, 559

H

Hartley David, 406, 453
 hasło, 305, 433
 ciąg zaburzający, 446, 448
 deszyfrowanie, 341
 do zapisanych punktów
 dostępowych Wi-Fi, 431
 Gmail, 431
 maskowanie, 442
 szyfrowanie, 341
 w kodzie źródłowym, 448
 Hay Roe, 286, 287, 661
 Herd Financial, 41
 Hoggard Henry, 388
 Hotz George, 248
 HTML5, 30
 Huene Peter, 575

I

iBeacon, 171
 iBooks Store, 77
 iBoot obraz, 46
 IDE, 192
 identyfikator
 GUID, *Patrz:* GUID
 unikatowy urządzenia,
 Patrz: UDID
 iGoat, 41
 iMAS, 40
 IMEI, 447
 intencja, 216, 273, 311
 BROWSABLE, 442
 filtr, 217, 273, 284, 316, 370,
 442
 internetowa, 371
 jawna, 217, 273
 niejawna, 273

przechwytywanie, 314
 przekazywanie, 274, 275
 z adresem URI, 334

interfejs, 118
 DPAPI, 600
 JavaScript, 412
 JavaScriptInterface, 453, 456
 loopback, 181
 Manifest Designer, 470
 skanowanie pod kątem
 niestandardowych portów,
 181
 UriMapperBase, 550

interprocess communication,
Patrz: komunikacja
 międzyprocesowa

inżynieria
 odwrotna, 34, 40, 79, 85, 88,
 111, 178, 189, 250, 261, 341,
 370, 486, 515, 516, 612
 spowolnienie, 460, 461
 zapobieganie, 40
 społeczna, 382, 420

iOS
 bezpieczeństwo, *Patrz:*
 bezpieczeństwo iOS
 dostęp do urządzenia, 61
 koprocessor, 47
 parowanie z komputerem, 67,
 68
 partycja, 182
 SDK, 50
 ustawienia prywatności, 54,
 55, 56

iOS 8 uwierzytelnianie,
Patrz: uwierzytelnianie iOS 8

ios app store package,
Patrz: pakiet IPA

IOUSBDeviceFamily, 60

IPC, *Patrz:* komunikacja
 międzyprocesorowa

J

jailbreak, 111, 121, 159
 evasi0n, 58, 60
 evasi0n7, 60
 kategoria, 59
 oprogramowanie zdalnego
 dostępu, 181

Pangu, 52, 58
 poziom trwałości, 59
 semi-tethered jailbreak, 59
 ślady, 180
 testowanie, 61, 138
 tethered jailbreak, 59
 untethered jailbreak, 59
 wykrycie, 179, 180

jailbreaking, 41, 46, 47, 56, 57, 58

JailbreakMe, 58

Java, 585
 maszyna wirtualna, *Patrz:*
 maszyna wirtualna Javy

jądro
 CE, 467
 iOS, 46, 48
 modyfikacja, 59
 NT, 467

język, 32
 AIDL, 303
 C, 50, 121
 C#, 483, 484, 548, 632
 C++, 50, 121
 CSS, 653, 654
 HTML, 653, 654
 Java, *Patrz:* Java
 JavaScript, 148, 177, 453, 542,
 548, 653, 654
 blokowanie, 646, 647
 natywny, 46
 Objective-C, 30, 50, 117, 118,
 120, 121, 164, 177
 interfejs, 118
 programowania
 sieciowego, 653
 zorientowanego obiektowo,
 118
 Python, 134, 208, 381
 Qt, 30
 Swift, 50, 89, 90, 117, 118,
 121, 164
 dekorowanie, 91
 modyfikator dostępu, 119
 typu opcode, 585

JINI, 340

Johnson Ken, 41

Johnson Kevin, 41

K

kanarek stosu, 50, 84, 346, 475

Kaplan David, 661

karta
 eMMC, 602
 SD, 238, 325, 386, 430, 438,
 446, 449, 472, 528
 szyfrowanie, 238
 wewnętrzna, 325
 zewnętrzna, 325

Kaseya BYOD, 132

catalog, 319, 320

keybag, 68

keychain, *Patrz:* pęk kluczy

klasa, 118
 AccountManager, 448
 android.os.Build, 462
 android.os.Debug, 463
 bezpieczeństwa, 70, 71
 Binder, 303
 CFNetwork, 97
 ClipboardManager, 337
 CLLocationAccuracy, 158
 Context, 386
 CordovaWebView, 661
 DexClassLoader, 374
 EmployeeDataContext, 604
 ExternalStorageDevice, 529
 ExternalStorageFolder, 529
 HostnameVerifier, 330
 IsolatedStorageSettings, 594,
 595
 JavascriptInterface, 335
 języka Swift, 89, 90
 kSecAttrAccessibleAfterFirst
 ↪Unlock, 73, 108
 kSecAttrAccessibleAfterFirst
 ↪UnlockThisDeviceOnly,
 73
 kSecAttrAccessibleAlways,
 73, 108
 kSecAttrAccessibleAlways
 ↪ThisDeviceOnly, 73
 kSecAttrAccessibleWhenPass
 ↪codeSetThisDeviceOnly,
 73
 kSecAttrAccessibleWhen
 ↪Unlocked, 73

- kSecAttrAccessibleWhen
 - ↳ UnlockedThisDeviceOnly, 73
- libxml2, 152
- Log, 326
- Messenger, 303
- NSData, 71, 72
- NSUserDefaults, 69
- NSFileHandle, 153
- NSFileManager, 71, 72, 153
- NSFileProtectionCompleteUntil
 - ↳ FirstUserAuthentication, 108
- NSFileProtectionNone, 108
- NSObject, 123
- NSURLConnection, 97
- NSURLSession, 97, 98
- NSXMLParser, 152
- ochrony, 108, 109, 110
- PackageManager, 462
- PreferenceActivity, 286, 441
- rejestracji danych, 457
- RootViewController, 89
- ScriptRunner, 322
- SecureRandom, 447
- SQLiteDatabase, 443
- SQLiteStatement, 443
- UIDevice, 157
- WebSettings, 333
- WebView, 334, 656
- WebViewClient, 455
- zaczepianie, 127
- Klein Tobias, 458
- klient
 - czatu internetowego, 375
 - poczty elektronicznej, 375
 - sqlite3, 70
- klucz
 - algorytm wymiany, *Patrz:* algorytm wymiany kluczy
 - asymetryczny, 449
 - debugowania, 200
 - generowanie, 615, 636, 639, 640, *Patrz też:* PBKDF2 na podstawie hasła, 615, 616, 617, 636
 - główny, 172
 - kryptograficzny, 170
 - miejsce umieszczenia, 447, 448
 - ochrona, 172
 - pęk, *Patrz:* pęk kluczy
 - protection_class, 110
 - prywatny, 39, 172, 223
 - publiczny, 46, 172
 - infrastruktura, *Patrz:* PKI
 - oczekiwany, 106
 - RSA, 620
 - stałe zdefiniowany
 - w aplikacji, 612
 - statyczny, 39, 174
 - symetryczny, 449
 - systemu plików, 48
 - szyfrowania, 73, 448
 - UID, 70
 - usuwanie, 614, 641
- kod
 - C#, 632
 - dynamiczny, 453
 - HTML, 51
 - dla UIView, 177
 - dynamiczny, 646
 - JavaScript, 548, 646
 - dynamiczny, 646
 - komunikatu, 303
 - kreskowy, 527
 - natywny, 484, 623, *Patrz też:*
 - aplikacja natywna
 - atak, *Patrz:* atak na kod natywny
 - bezpieczeństwo, 649
 - deasemblacja, 261
 - dekompilacja, 261
 - samodzielna weryfikacja, 187
 - wyszukiwanie, 340
 - PIN, 306, 387, 392, 432, 433, 543
 - podpisywanie, 46, 47, 51, 223
 - QR, 171, 527
 - rozruchowy, 46
 - shellcode, 478
 - systemowy AOSP, 247
 - tajny, 316, 317, 318, 445
 - uwierzytelniania
 - wiadomości, *Patrz:* MAC
 - wstrzykiwanie, 442, 543
 - wykonanie zdalne, 374, 405
 - XML, 566, 569
 - zabezpieczający, 171
 - szyfrowanie, 71
 - źródłowy zaciemnianie, 180, 181, 184, 189, 190, 358, 460, 461
- komora LPC/TCB, 469
- kompas, 659
- kompilator
 - clang, 138
 - LLVM, 84, 187
- komponent
 - czynność, *Patrz:* czynność
 - eksport, 266, 267, 279, 280, 334
 - jawny, 267
 - niejawny, 267
 - wyszukiwanie, 268, 272
 - niewyeksportowany, 268
 - UIView, 177, 654
 - kod HTML, 177
 - usługa, *Patrz:* usługa
- WebBrowser, 520, 521, 522, 534, 541, 548, 553, 597, 598, 599, 645, 646, 647, 648, 649
 - włączenie obsługi
 - JavaScript, 542, 543
- WebRequest, 649
- WebView, 36, 332, 374, 412, 453, 520, 522, 534, 541, 548, 553, 598, 599, 645, 646, 647, 649, 661
 - JavaScriptInterface, 453, 456
 - wtyczka, 426, 454
 - zabezpieczenie, 453, 454
- komunikacja, 451
 - bezpieczeństwo, *Patrz:*
 - bezpieczeństwo komunikacji między aplikacjami, 452
 - między JavaScript i C#, 548
 - międzyprocesowa, 39, 142, 145, 303, 530, 549, 635, 661
 - binder, 213, 216, 237
 - powiązanie rozszerzenia plików, 530, 531, 553, 556, 579
 - procedura obsługi
 - protokołu, 530, 549, 550, 552

- sieciowa, 327, 328, 329, 330, 339, 377, 380, 525, 533, 536
 - przechwytywanie, 331, 332, 333, 339, 537, 539
 - testowanie, 338
 - komunikat
 - kod, *Patrz:* kod komunikatu
 - NFC, 527
 - odbiorca, 267, 310, 312, 314, 445
 - niechroniony, 310
 - rozgłoszeniowy, 311
 - toast, *Patrz:* toast
 - konsola ASL, 159
 - kontroler widoku, 128, 130
 - kontrolka
 - WebBrowser, 520, 521, 522, 534, 541, 548, 553, 597, 598, 599
 - włączenie obsługi
 - JavaScript, 542, 543
 - WebView, 520, 522, 534, 541, 548, 553, 598, 599
 - kopia zapasowa, 108
 - kradzież tożsamości, 38
 - kryptografia, *Patrz:* szyfrowanie
 - książka adresowa, 157, 235, 427
 - Kurtz Andreas, 139
- L**
- least privilege chamber, *Patrz:* komora LPC
 - liczba
 - całkowita, 628, 629
 - losowa, 446, 447, 607, 615, 640
 - kolekcja, 610
 - pseudolosowa, 607
 - linker dynamiczny, *Patrz:* dyld
 - Linux
 - dstrybucja, 192
 - użytkownik, 197
 - LLB, 46
 - Logos, 127
 - low-level bootloader, *Patrz:* LLB
 - luka
 - 0x24000 Segment Overflow, 59
 - addJavascriptInterface, 36
 - CVE-2010-1759, 399
 - CVE-2011-0226, 58
 - CVE-2011-0227, 58
 - CVE-2012-6636, 335, 374
 - CVE-2013-0979, 60
 - CVE-2013-0981, 60
 - CVE-2013-6271, 283, 394
 - CVE-2013-6272, 312
 - CVE-2014-1266, 101
 - CVE-2014-1272, 60
 - CVE-2014-1273, 60
 - CVE-2014-1278, 60
 - CVE-2014-3153, 248
 - CVE-2014-3500, 661
 - CVE-2014-7911, 425
 - Google Bug #13678484, 425
 - Google Bug #8219321, 227
 - Google Bug #9695860, 228
 - Google Bug #9950697, 229
 - goto fail, 35
 - po stronie serwera, 37
 - przepelnienia sterty na stosie
 - USB, 59
 - w aplikacji Skype, 145, 150
 - w zabezpieczeniach
 - aplikacji, 369
 - aplikacji WhatsApp, 326
 - aplikacji z włączoną opcją debuggable, 345, 346, 349, 350, 352, 354
 - blokady ekranu, 283, 394
 - błędnie skonfigurowanych atrybutów pakietu, 347
 - do zainstalowania pakietów dodatkowych, 422
 - dostawców treści, 289, 295
 - fałszywa tożsamość, 425
 - filtrowania pakietów jądra, 59
 - frameworka Cordova, 661
 - funkcji rejestrowania danych, 327
 - interfejsów JavaScript, 412
 - jądra, 422
 - jądra w systemie Android, 235, 248
 - klucza głównego, 227, 228
 - kodu jądra AOSP, 245
 - kodu systemowego AOSP, 247
 - komponentu WebView, 333, 334
 - mechanizmu futex, 414
 - menedżera sterty, 481, 626
 - metody normalize, 399
 - natywnego kodu C, 340, 341, 345, 623, 624
 - niestandardowych sterowników, 245
 - plików binarnych SUID, 247
 - po stronie klienta, 369
 - po stronie serwera, 369
 - prowadząca do kradzieży danych, 472
 - przeglądarek WWW, 399
 - Samsung Service Mode Application, 428
 - SEH, 480
 - urządzeń, 377
 - usług, 302
 - wtyczek WebView, 454
 - w zabezpieczeniach
 - pozwalająca
 - na inicjowanie i zrywanie połączeń, 312, 335, 395
 - na poruszaniu się po strukturze katalogów, 300, 581
 - na wstrzyknięcie fragmentu, 287
 - na zdalne wykonanie kodu, 405
 - na zdalne wykorzystanie aplikacji, 417
 - w zabezpieczeniach
 - związanych
 - z ciągiem tekstowym formatowania, 164, 165, 166, 630
 - z indeksowaniem tablicy, 631
 - z odmową usług, 632
 - z próbą użycia obiektu po jego usunięciu, 167
 - z uprawnieniami plików, 246, 322
 - z uszkodzeniem pamięci, 164, 399, 628, 629

luka

- z weryfikacją podpisu cyfrowego pakietu APK, 227, 228, 229
- wyszukiwanie, 366, 369
 - aplikacja o szerokich uprawnieniach, 366, 367
 - automatyczne, 376

Ł

ładunek, 49

łańcuch procedur bezpiecznego rozruchu, 46

M

MAC, 102

magazyn danych, 67, 438, 489

- identyfikacja, 600
- niezabezpieczony, 34, 38, 42, 471, 594, 600, 602, 603, 606
- ochrona, 158
- poufnych, 593, 594
- SQLite, *Patrz:* SQLite

magazyn kluczy

kryptograficznych, 612

malware, 33, 57, 235, 420

Android, 474

iKee, 57

iKee.B, 58

RootSmart, 235

Unflod Baby Panda, 58

Unfold Baby Panda, 183

Mandt Tarjei, 481

manifest, 201, 216, 217, 236, 252, 281, 312, 371, 440, 442, 443, 449, 453, 471, 484, 485, 486, 530, 554

konfiguracja, 455

przekierowanie do pliku, 253

Mannino Jack, 41

Marlinspike, 451

marshaling, 303

maszyna wirtualna Javy, 198

MDM, 132

mechanizm

AppContainer, *Patrz:*

AppContainer

automatycznego zliczania

odwołań, *Patrz:* ARCMPNS, *Patrz:* MPNSSEHOP, *Patrz:* SEHOP

menedżer

czynności, 237

haseł, 131, 146, 337, 341, 433

sterty, 481

message authentication code,

Patrz: MAC

metadane

funkcji, 91

nadpisanie, 479

szyfrowanie, 173

metoda

addJavascriptInterface, 656, 657

application:openURL, 142, 143, 144

attach, 135

BaddressBookCopyArray

↳ OfAllPeople, 157

canEvaluatePolicy, 78

checkPassword, 131, 132

checkServerTrusted, 452

decrypt, 637

delete, 439

didReceiveAuthentication

↳ Challenge, 97

egzemplarza, 118, 119

encrypt, 637, 638

enumerate_modules, 135

evaluatePolicy, 78

getCanonicalPath, 443

getPassword, 448

getter, 118

handleMessage, 303

insert, 439

InvokeScript, 543

InvokeScriptAsync, 543

isDebuggerConnected, 463

isValidFragment, 441

klasy, 118, 119

loadHTMLString, 177

loadUrl, 661

NativeToString, 543

NSLog, 159

ObjC.use, 136

onBind, 302, 439

onCreate, 280, 439, 440

OnNavigatedTo, 551, 552, 558, 559, 560

onReceive, 439

onStartCommand, 439

openFile, 439

query, 439

z poleceniami

składowanymi, 443

rawQuery, 443

SecItemAdd, 73, 74

SecItemUpdate, 73

SecTrustSetOptions, 99

sendBroadcast, 273, 311, 314

setClass, 273

setComponent, 273

setter, 118

sharedInstance, 129

shouldInterceptRequest, 455, 663

shouldStartLoadWithRequest, 657

SSLSetProtocolVersion-Enabled, 102

SSLSetProtocolVersion, 102

SSLSetSessionOption, 106

startActivity, 273

startService, 273

strcpy, 625

stringByEvaluatingJavaScript

↳ FromString, 177

swizzling, *Patrz:* swizzling

TransformationMethod, 442

update, 439

XMLHttpRequest, 660

źródło pochodzenia, 183

Microsoft Store, 468, 489, 490

aplikacja, 472, 473, 474

mikrofon, 426

MMS, 375

Mobile Enterprise Application Platform, 51

MobiSec, 41

Moulu André, 302, 417, 418, 422

Moxie, 451, 452

MPNS, 562, 563

Murphy Mark, 269

N

- nadawanie uprawnień URI, 449, 450
- nagrywanie dźwięku
 - z mikrofonu, 426
- narzędzie, *Patrz też*: polecenie
 - .NET reflector, 514, 522, 560, 578, 584
 - aapt, 193, 200, 252
 - adb, 192, 202, 203
 - Adios, 157
 - aidl, 200
 - am, 274
 - android, 193
 - apkbuilder, 200
 - apktool, 261, 356
 - Application Deployment, 492, 497
 - apt, 62
 - AXMLPrinter2, 253
 - baksmali, 255, 259, 260
 - BigBoss, 62
 - BinScope, 516, 650
 - Burp, 380, 381, 413, 415
 - Burp Suite, 96, 104, 328, 338, 361, 534, 536, 537
 - odszukanie treści HTTP, 329
 - odszukanie treści HTTPS, 329, 332
 - Burp Suite Pro, 514
 - BusyBox, 204
 - CC Tools, *Patrz*: CC Tools
 - chmod, 320
 - class-dump-z, 88, 89, 90, 129, 131
 - Clutch, 87
 - codesign, 66
 - cp, 205
 - Crackulous, 87
 - Cycript, 128, 130, 131, 132
 - Cydia Impactor, 247, 392, 394
 - Cydia Substrate, 360, 363
 - dex2jar, 256
 - dexdump, 254, 262
 - dexopt, 221
 - drozer, 207, 213, 217, 230, 233, 254, 268, 370, 376, 382, 413
 - agent, 383, 386, 391, 397, 398
 - instalacja, 208
 - konsola, 209
 - ładunek, 384, 386
 - moduły, 426
 - plik JAR, 398
 - polecenie, 382
 - protokół, 383
 - przestrzeń nazw, 210
 - rozszerzenie Burp, 385
 - uprawnienia, 275
 - uruchamianie, 383
 - wbudowane programy, 384
 - wersja, 207
 - dumpsys, 206
 - dx, 200
 - Ettercap, 377, 378, 413, 415
 - frida-trace, 134
 - gdb, 65, 85
 - getprop, 206
 - grep, 205, 357, 376
 - Hopper, 93
 - IDA, 255, 261, 341
 - IDA Pro, 93, 514
 - iExplorer, 68
 - ILSpy, 514
 - ios-dataprotection, 108
 - ipainstaller, 67
 - JAD, 262
 - Jadx, 262
 - jarsigner, 200
 - JD-GUI, 257
 - JEB, 257, 258
 - keychain_dump, 77, 109
 - keychain_dumper, 76, 77
 - keytool, 225
 - ldid, 66
 - lipo, 65, 82
 - lldb, 65, 85
 - logcat, 206, 327, 343, 363
 - Mallory, 539
 - monitor, 192
 - Native Access Webserver, 514
 - nm, 65
 - openssl, 224
 - otool, 63, 81, 83
 - pm, 206
 - powłoki, 62
 - przechwytywania pakietów, 96
 - Reflexil, 514, 584
 - rkBreakout, 511
 - SamWP8, 505
 - screenrecord, 428
 - SDK, 494
 - SEE, 606
 - smali, 259, 260
 - Snoop-it, 139, 141
 - SQLCipher, 173, 575, 606
 - SQLite Encryption Extension, *Patrz*: SEE
 - sqlite3, 299
 - sqlmap, 295
 - SSL Kill Switch, 107
 - su, 243
 - swift-demangle, 92
 - tcpdump, 338
 - TrustMe, 107
 - ultrasn0w, 57
 - Visual Studio, 494, 495
 - WhatsApp Xtract, 326
 - which, 462
 - Wireshark, 96, 338, 339
 - WP8 File Explorer, 514
 - Xposed Framework, 360
 - zipalign, 200
 - NDK, 190, 461
 - NFC, 525
 - NXCOMPAT, 650

O

- Obfuscator-LLVM, 190
- odbiorca komunikatów, *Patrz*: komunikat odbiorca
- odcisk palców, 47, 75, 77
- O-LLVM, 461
- OpenBinder, 214
- OpenSSH, 61
- oprogramowanie
 - cykl życiowy, *Patrz*: SDL
 - DVM, *Patrz*: DVM
 - IDA, 255
 - zdalnego dostępu, 181
- OWASP, 29, 36, 534
- OWASP Mobile Security, 36, 40

P

- p0sixninja, 60
 - pakiet
 - adv-cmds, 62
 - APK, 219, 439
 - pobieranie, 251
 - struktura katalogów, 200
 - ścieżka dostępu, 251
 - APPX, 473, 482, 484
 - CC Tools, *Patrz:* CC Tools
 - coreutils, 62
 - dodatkowy, 422
 - IPA, 52, 53
 - system-cmds, 62
 - Theos, 126, 128
 - XAP, 482
 - pakiet
 - podgląd na żywo, 339
 - szyfrowania, *Patrz:*
 - szyfrowanie pakiet
 - palmtop, 30
 - pamięć
 - alokacja, 164
 - masowa, 325
 - NAND flash, 48
 - uszkodzenie, 164
 - PAN, 375
 - Pangu jailbreak, 52, 58
 - Parmar Ketan, 219
 - partycja
 - systemowa, 48
 - tabela, *Patrz:* tabela partycji
 - password-based key derivation function, *Patrz:* PBKDF2
 - password-based key derivation function 2, *Patrz:* algorytm PBKDF2
 - payload, *Patrz:* ładunek
 - PBKDF2, 172
 - PDA, *Patrz:* palmtop
 - pęk kluczy, 48, 108, 110
 - dodawanie elementu, 73, 74
 - iOS, 73
 - przenoszenie, 73
 - reguły dostępu, 74
 - uzyskiwanie dostępu, 75, 76, 77
 - współdzielenie, 74
 - phishing, 58, 552, 644, 645
 - piaskownica, 46, 48, 54, 60, 67, 142, 179, 181, 223, 235, 237, 319, 422, 453, 599, 602
 - Android, 445
 - WP8, 469
 - PIE, 49, 82, 83, 456
 - PII, *Patrz:* dane osobowe
 - piractwo, 57
 - ochrona, 85
 - PKI, 226
 - Plaskett Alex, 558
 - plik
 - .aidl, 303
 - .ext, 528
 - .key, 392
 - accounts.db, 431
 - AndroidManifest.xml, 201, 216, 229, 252
 - APK, 255
 - APPX, 468
 - ARM, 624
 - backup.ab, 348
 - banned.h, 650
 - binarny, 65, 200, 515
 - .NET, 522, 560, 578, 584, 589
 - iOS, 79
 - Mach-O, 63, 79, 80, 81
 - PIE, 456
 - podpisany cyfrowo, 66
 - screencap, 428
 - skompilowany, 111
 - su, 356, 358, 462
 - SUID, 242, 247
 - zaszyfrowany, 85
 - CERT.RSA, 224
 - classes.dex, 200, 221
 - config.xml, 662
 - cookie, 598, 648
 - binarny, 69
 - CSV, 601
 - default.prop, 462
 - DEX, 221, 222, 254, 255, 256, 259
 - optymalizacja, 221
 - exploit.html, 334
 - format, *Patrz:* format
 - frida-server, 134
 - gesture.key, 432
 - grupa, 319, 320, 392
 - JAR, 398
 - Jython JAR, 381
 - manifestu, *Patrz:* manifest
 - MBN, 506, 507, 508
 - multimedialny, 659
 - OAT, 262
 - objektowy, 65
 - ODEX, 259, 374
 - Package.appxmanifest, 470, 482, 484, 528, 530, 550, 554
 - package.list, 220
 - packages.xml, 220
 - password.key, 433
 - property list, 69
 - ROM, 511
 - szyfrowanie, *Patrz:*
 - szyfrowanie pliku
 - Tweak.xm, 127
 - uprawnienia, 246, 319, 320, 321, 392, 445
 - właściciel, 319, 320, 392
 - WMAAppManifest.xml, 482, 484, 528, 530, 549, 554
 - XAML, 483, 542, 543
 - XAP, 468
 - XML, 567, 568, 572
- polecenie, *Patrz też:* narzędzie
- #include, 650
 - adb shell, 388
 - chmod, 458
 - ipainstaller, 67
 - składowane, 443
 - which su, 462
- polityka
- hasła, 617
 - LAPolicyDeviceOwnerAuthenticationWithBiometrics, 78
 - tęgo samego źródła, *Patrz:* SOP zapisu, ale nie wykonania, *Patrz:* W^X
- pomost natywny, 335, 655, 656
- Android, 656
 - iOS, 657
 - Windows Phone, 658
- port
- 5555, 337
 - 80, 380, 405
 - nasłuchujący, 374

- numer, 339
 - skanowanie, 374
 - TCP 22, 181
 - TCP 5039, 346
 - position independent executable, *Patrz:* PIE
 - position-independent executable, *Patrz:* PIE
 - position-independent execution, *Patrz:* PIE
 - powiadomienie
 - push, 563, 565
 - toast, *Patrz:* toast
 - powłoka ADB, 387, 388, 389, 394, 397
 - Pradines Florian, 41
 - prefiks, 444
 - PRNG, *Patrz:* generator liczb pseudolosowych
 - procedura
 - bezpiecznego rozruchu, 46, 47, 107
 - obsługi pliku, 528, 530, 531, 553, 556, 579
 - obsługi protokołu, 142, 171, 530, 549, 550, 552
 - atak, 143
 - wywołanie zdalne, *Patrz:* RPC
 - proces działający w tle, 48, 161, 162
 - procesor, 30
 - profil
 - provisioning, 47, 51
 - seatbelt, 48
 - tymczasowy dystrybucyjny, 51
 - program, *Patrz też:* narzędzie
 - abdb, 462
 - Apple Developer Enterprise, *Patrz:* Apple Developer Enterprise
 - Apple Developer Program, *Patrz:* Apple Developer Program
 - Decompiler, 261
 - gingerbread, 235
 - iTunes, *Patrz:* iTunes
 - limeraln, 59
 - rozruchowy, 249, 389, 398
 - iBoot, 46
 - niskiego poziomu, *Patrz:* LLB
 - odblokowanie, 389
 - uruchamianie, 249
 - Towelroot, 248, 392, 414
 - Xcode, *Patrz:* Xcode
 - programowanie
 - zorientowane obiektowo, 118
 - zwinne, 36
 - ProGuard, 457, 461
 - protection level downgrade attack, *Patrz:* atak powodujący obniżenie poziomu ochrony uprawnień
 - protokół, 171
 - drozerp, 208, 383
 - fastboot, 249
 - HTTP, 32, 96, 374, 383, 524, 525, 537
 - HTTPS, 103, 453, 455, 537, 539, 645
 - OpenSSL, 96
 - PolarSSL, 96
 - procedura obsługi, *Patrz:* procedura obsługi protokołu
 - SSL, 96, 102, 330, 451, 643, 644, 645
 - atak, 99
 - certyfiat, 97
 - Goto Fail, 101
 - implementacja, 96
 - słownik ustawień, 98
 - testowanie, 99, 101
 - wersja, 101
 - TCP, 338, 340
 - TLS, 96, 643, 644, 645
 - certyfiat, 97
 - Goto Fail, 101
 - implementacja, 96
 - testowanie, 99, 101
 - wersja, 101
 - UDP, 338, 340
 - proxy, 99, 103, 380, 534, 536
 - Burp, 412
 - Burp Suite, 96, 328, 338, 361, 380
 - odszukanie treści HTTP, 329, 380, 381
 - odszukanie treści HTTPS, 329, 332
 - przechwytyjące, 327, 328, 361
 - HTTP, 329
 - HTTPS, 329, 332
 - przeglądarka
 - bufor, 377, 648, 649
 - Safari, *Patrz:* Safari
 - WWW, 369, 386
 - przestrzeń adresowa układ losowy, *Patrz:* ASLR
 - przyspieszeniometer, 659
- ## R
- rainbow table, *Patrz:* tablica tęczowa
 - refleksja, 307, 335, 523
 - reguła najmniejszych uprawnień, 469, 471
 - reklama, 156, 412, 413
 - bezpieczeństwo, 171
 - remote procedure call, *Patrz:* RPC
 - ROP, 49, 58, 478
 - RPC, 302
- ## S
- Safari mobilne, 50
 - SafeSEH, 480, 517, 650
 - salt, *Patrz:* ciąg zaburzający
 - same-origin policy, *Patrz:* SOP
 - Samsung Apps, 202
 - sandbox, *Patrz:* piaskownica
 - saurik, 66, 70, 121
 - Sawyer Jon, 461
 - schowek
 - globalny, 336, 337
 - niestandardowy, 159, 160
 - operacji wyszukiwania, 159
 - systemowy, 159, 336, 337
 - użytkownika, 433
 - SDK, 192
 - SDL, 171
 - secure boot chain, *Patrz:* łańcuch procedur bezpiecznego rozruchu
 - Secure Enclave, 47, 70, 78
 - SEH, 479
 - Selma Sebastián Guerrero, 300

serwer
 C&C, 58
 OpenSSH, 181
 SID, 471
 sideloadng, 468, 491
 sieć
 bezprowadowa, 412, 415
 społecznościowa
 komunikat, 176
 oparta na danych
 geolokalizacji, 41
 Twitter, *Patrz:* Twitter
 Sigwald Jean, 77
 silnik czcionek FreeType, 58
 skrypt
 checksec, 458
 lokalny, 543, 544, 547
 SlideMe, 202
 SMS, 375, 427
 premium, 42, 235
 software development lifecycle,
 Patrz: SDL
 SOP, 544
 SQLite, 70
 stack canary, *Patrz:* kanarek stosu
 standard HTML5, *Patrz:*
 HTML5
 stenografia, 170
 sterownik IOUSBDeviceFamily,
 60
 stos
 kanarek, 50, 84, 346, 475
 przepelnienie, 50, 401, 475,
 624, 625
 USB, 59
 zabezpieczenie przed
 uszkodzeniem, 50, 84
 Stringer, 461
 swizzling, 120, 123, 138
 system
 iOS, 40, 45, *Patrz też:* iOS
 bezpieczeństwo, *Patrz:*
 bezpieczeństwo iOS
 jądro, *Patrz:* jądro iOS
 nawigacji satelitarnej,
 Patrz: GPS
 odzyskiwanie, 249
 operacyjny
 Android, *Patrz:* Android
 Captain Hook, 123
 iOSOpenDev, 123
 Linux, *Patrz:* Linux
 Theos, 123
 plików, 153, 319, 512, 659
 deszyfrowanie, 48, 237
 dostęp, 67
 interesujące lokalizacje, 68
 przeglądanie, 67
 szyfrowanie, 237
 Windows Phone, 499
 wczytywania adresu URL, 97,
 98
 zabezpieczeń warstwowy, 355
 szyfrowanie, 34, 38, 39, 46, 70, 96,
 174, 446, 461, 471, *Patrz też:*
 algorytm
 AES, 48, 173, 446, 447
 aes-cbc-essiv:sha256, 238
 algorytm, 172
 API Data Protection, 48
 aplikacji, 85
 asymetryczne, 446
 bazy danych, 46, 48, 173, 606,
 639
 dysku pełne, 237, 238
 hashowanie, 172
 implementacja, 171
 reguły, 172
 klucz, *Patrz:* klucz
 szyfrowania
 kodu zabezpieczającego,
 Patrz: kod zabezpieczający
 szyfrowanie
 metadanych, 173
 oparte na bloku, *Patrz:*
 szyfrowanie AES
 pakiet, 38
 partycji, 238
 pliku, 70, 324, 637
 poziom ochrony, 71, 72
 schemat własny, 446
 SHA, 446
 SQLCipher, 173
 SSL, 643, 644, 645
 symetryczne, 446
 TLS, 643, 644, 645
 wektor inicjalizacji, 620, 636
 zestaw szyfrowy, *Patrz:* zestaw
 szyfrowy

Ś

środowisko
 uruchomieniowe
 ART, 222, 262
 atak, 117, 120
 instrumentacja, 120, 121,
 128, 130, 131, 134, 137,
 183
 manipulacja, 359
 WinRT, 483
 zaczep, 185, 186
 Xcode, 53, 65, 82, 84, 92, 123,
 159, 190

T

tabela
 partycji, 48
 v-table, 89, 120, 121
 tablica
 indeksowanie, 631
 tęczowa, 172, 446, 447
 UIApp.windows, 130
 tamper detection, 463
 tapjacking, 284, 441
 zapobieganie, 285
 technika
 ASLR, *Patrz:* ASLR
 automatycznego pobierania
 pliku, 420
 ROP, 478, *Patrz:* ROP
 SEH, *Patrz:* SEH
 technologia
 3G, 32
 4G, 32
 BitLocker, 472
 FairPlay DRM, 468
 geo-fence, 32
 NFC, 525
 SafeSEH, *Patrz:* SafeSEH
 test
 analizatora binarnego
 BinScope, 473
 penetracyjny, 41, 130, 192,
 486, 493
 tethering, 57
 Thomas Braden, 60

toast, 284, 532, 557, 558
 wysłanie, 558, 561, 562, 563
 zdalne, 562
 token, 39, 471
 żądania, 460

Touch ID, *Patrz:* czytnik Touch ID
 trusted computing base,
Patrz: komora TCB
 TrustZone, 47
 Twitter, 251, 270, 271, 361
 two-factor authentication,
Patrz: uwierzytelnianie
 dwupoziomowe

U

UDID, 156
 UID, 197, 198, 211, 345, 449
 UIWebView, 147, 148, 153, 163,
 170, *Patrz:* komponent
 UIWebView
 unique user identifier, *Patrz:* UID
 usługa
 AppleKeyStore, 77
 AuthService, 304, 305
 CryptoService, 306
 GTalkService, 201
 installd, 67
 MobileBackup, 58
 MTP, 505
 nasłuchująca, 375
 niezabezpieczona, 301, 302
 odmowa, 632
 SSH, 57, 58, 61
 uruchomiona, 301
 usbmuxd, 53, 61
 uwierzytelnianie, 38, 73, 438, 442
 dwupoziomowe, 42
 iOS 8, 75
 mechanizm lokalny, 128, 131
 Uzzel Matthew, 276
 użytkownik
 konto, 431
 post-jailbreak, 180
 root, 204, 241, 268, 269, 355,
 356, 363, 387, 389, 428, 462
 uprawnienia, 241, 242, 245,
 249
 shell, 428
 system, 268, 269, 387, 428

V

Valasek Chris, 481
 van Wyk Ken, 41

W

W^X, 49
 WACK, 473
 Walker Nick, 558
 warstwa
 prezentacyjna, 31
 transportowa, 34, 38, 42, 96
 wektor inicjalizacji, 620
 WhatsApp, 326
 widok
 kontroler, *Patrz:* kontroler
 widoku
 UIAlertView, 111, 112, 115
 Wi-Fi, 33
 windows app certification ki,
Patrz: WACK
 Windows Phone, *Patrz też:* WP7,
 WP8
 emulator, 494, 496, 497
 komora bezpieczeństwa,
Patrz: komora
 mechanizmy obronne, 468,
 474, 650
 ASLR, 476, 477
 DEP, 478
 menedżer sterty, 481, 626
 SafeSEH, 480
 SEH, 479
 SEHOP, 480
 w przestrzeni jądra, 481
 zabezpieczenie stosu, 475
 odblokowanie urządzenia,
 499, 500, 501, 502, 505, 506,
 507, 508, 511, 512, 514, 602
 Windows Phone Dev Center, 491
 Windows Phone Marketplace,
 491
 Windows Phone Store, 30
 WP7, 467, 482
 WP8, 467, 468, 482
 certyfikat, 644
 tryb programisty, 468, 492,
 497, 498, 499

wstrzykiwanie
 fragmentów, *Patrz:* fragment
 wstrzykiwanie
 kodu, *Patrz:* kod
 wstrzykiwanie
 polecień SQL, 292, 293
 wyjątek SecurityException, 459
 wywołanie zwrotne, 337
 wzorzec
 blokady ekranu, *Patrz:* ekran
 blokada wzorzec
 dopasowanie, *Patrz:*
 dopasowanie wzorca

X

Xcode, 53, 65, 82, 84, 92, 123,
 159, 190

Z

zabezpieczenie binarne, 34, 40,
 42, 178, 179
 zapytanie SQL, 150, 176, 576
 parametryzowane, 176, 642
 zatrucie bufora ARP, *Patrz:* bufor
 ARP zatrucie
 zdarzenie
 rejestracja, 457
 ScriptNotify, 658
 zestaw szyfrowy, 102, 103

Ż

żądanie
 do sieci wewnętrznej, 132
 HTTP, 69
 bezpieczne, 97
 w postaci zwykłego tekstu,
 96
 HTTPS, 329, 330, 332
 token, *Patrz:* token żądania

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

Aplikacja mobilna

— popatrz na nią oczami hakera i zabezpiecz ją!

Urządzenia mobilne zapewniają ogromną wygodę. Natychmiastowy dostęp do informacji czy dokumentu, niezależnie od lokalizacji użytkownika, jest czymś oczywistym. W ten sposób wszelkie ważne i wrażliwe informacje, takie jak dane pozwalające na identyfikację, dane finansowe czy poufne dokumenty, są cały czas na wyciągnięcie ręki — niestety, często ta ręka należy do kogoś, kto w żadnym razie nie powinien takich informacji uzyskać. Każdy, kto pisze aplikacje mobilne, musi pamiętać o kwestiach związanych z ich bezpieczeństwem. Konsekwencje nieuprawnionego dostępu do danych mogą być niezwykle poważne!

Ta książka jest całościowym, a równocześnie bardzo praktycznym kompendium wiedzy o bezpieczeństwie aplikacji mobilnych. Uwzględniono tu problemy charakterystyczne dla platform iOS, Android i Windows Phone, dzięki czemu zaproponowanie najwłaściwszej strategii zabezpieczenia aplikacji jest o wiele prostsze. Wyjaśniono przyczyny podatności aplikacji mobilnych na ataki, opisano też techniki prowadzenia ataku i wykorzystywania luk w zabezpieczeniach. Bardzo dokładnie przedstawiono także strategię obrony i działania, dzięki którym programiści mogą chronić swoje aplikacje. Poradnik ten docenią przede wszystkim osoby przeprowadzające testy penetracyjne, konsultanci z zakresu bezpieczeństwa oraz oczywiście programiści.

Najciekawsze zagadnienia:

- 10 najważniejszych zagrożeń aplikacji mobilnych według OWASP Mobile Security
- Analiza aplikacji i identyfikowanie problemów bezpieczeństwa
- Ataki typu injection, brute force, XSS, tapjacking i wiele innych
- Wykorzystanie inżynierii wstecznej
- Mechanizmy obronne w aplikacjach na poszczególne platformy
- Zabezpieczanie aplikacji niezależnych od platformy

DOMINIC CHELL

jest ekspertem w dziedzinie zabezpieczeń aplikacji na platformę iOS. Jest znany z wystąpień na konferencjach oraz jako autor wielu publikacji dotyczących bezpieczeństwa urządzeń mobilnych.

TYRONE ERASMUS

zajmuje się badaniami z zakresu bezpieczeństwa systemów. Interesuje się testami penetracyjnymi. Bada nowe narzędzia i techniki pojawiające się w tej sferze.

SHAUN COLLEY

jest ekspertem z zakresu bezpieczeństwa w urządzeniach mobilnych i oceny kodu natywnego. Zajmuje się też inżynierią wsteczną.

OLLIE WHITEHOUSE

specjalizuje się w zapewnianiu bezpieczeństwa aplikacji i urządzeń mobilnych, interesuje się też technikami przewodowymi.

Bardzo dobre, szerokie wprowadzenie do tematyki bezpieczeństwa aplikacji mobilnych. Pokazuje nie tylko jak zabezpieczać, ale również jak testować aplikacje pod kątem ataków. Brak wypełniaczy i struktura sprawia, że każdy ma szansę zdobyć wiedzę w sposób rzeczywiście efektywny. Stanowi unikalną mapę, która bardzo często wspiera Czytelnika w wychodzeniu również poza omówione obszary.

MATEUSZ BILIŃSKI
HEAD OF MOBILE SECURITY
W NIEBEZPIECZNIK.PL

Helion	KOD KORZYŚCI Sięgnij po więcej! ▶	
 helion.pl	ISBN 978-83-8322-548-7	
 HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	 9 788383 225487	
Cena: 129,00 zł		