Przedmowa

> Firmy tracą rocznie 400 miliardów dolarów na skutek działań hackerów.[1]
>
> *– Inc. Magazine*

Raport bezpieczeństwa rynku informatycznego, opublikowany przez Cybersecurity Ventures w czwartym kwartale 2015 roku, stwierdza, że cyberataki kosztują biznes od 400 do 500 miliardów dolarów rocznie[2]. Jednocześnie wydatki na bezpieczeństwo IT wzrosły o 4,7% w roku 2015, osiągając wartość 75,4 miliarda USD, zaś szacuje się, że ogólne wydatki światowe na ten cel wzrosną do 101 miliardów dolarów w roku 2018 i będą dalej rosnąć do 170 miliardów w roku 2020. Można przewidywać, że w branży bezpieczeństwa informatycznego pojawi się niedobór pracowników sięgający 1,5 miliona osób w roku 2019, jako że ogólnoświatowe zapotrzebowanie ma wzrosnąć do 6 milionów.

Jako projektanci aplikacji i witryn Web, inżynierowie i twórcy nie możemy dłużej żyć w przekonaniu, że możemy powierzyć zagadnienia bezpieczeństwa tożsamości i danych komuś innemu. W dzisiejszych czasach projektant witryny może bezwiednie otworzyć lukę zabezpieczeń jedynie dlatego, że nie wiedział, jak właściwie ochronić dane w trakcie transmisji. Nieświadomość tego, że uważany dotąd za bezpieczny algorytm zabezpieczania haseł zawiera błędy, może spowodować podatność na potężny atak, o ile nie nadamy odpowiedniego priorytetu zadaniu ponownego zaszyfrowania bazy danych użytkowników. W interesie każdej osoby pracującej z jakimś systemem jest zadbanie o to, że nasi użytkownicy i ich dane są chronione.

Pomimo tego wydaje się, że każdego tygodnia pojawiają się nowe zdarzenia, gdy różne firmy, od startupów po wielkie korporacje, tracą poufne informacje użytkowników, dane kart kredytowych, rejestry medyczne i wiele innych typów informacji, które zdecydowanie powinny być chronione. Okazuje się często, że wiele z tych organizacji nigdy nie zadało sobie trudu, aby właściwie zabezpieczyć dane, przechowując wszystko w postaci jawnego tekstu i czekając, aż jakiś hacker to wykorzysta.

1 *http://www.inc.com/will-yakowicz/cyberattacks-cost-companies-400-billion-each-year.html*
2 *http://cybersecurityventures.com/cybersecurity-market-report/*

Prawdziwy problem polega na tym, że hacking nie jest już tylko hobby indywidualistów, którzy pragną udowodnić, że potrafią włamać się do jakiegoś systemu. Dziś jest to sfera zorganizowanej przestępczości, wykorzystującej techniki hackerskie dla pieniędzy lub w celu zniszczenia innej firmy.

W miarę przeglądania każdej koncepcji i idei w kolejnych rozdziałach będziemy pokazywali, jak pozatykać dziury w istniejących systemach, ochronić się przed potencjalnie skutecznymi wektorami ataku i jak pracować w środowiskach, które są w naturalny sposób niezabezpieczone. Przyjrzymy się takim zagadnieniom, jak:

- Aktualny stan zabezpieczeń sieci Web i aplikacji wraz omówieniem rozmaitych koncepcji.
- Zapewnianie bezpiecznego szyfrowania haseł i przeciwdziałanie możliwym atakom na hasła.
- Tworzenie cyfrowych „odcisków palców" w celu dodatkowej identyfikacji użytkowników poprzez unikatowe cechy ich przeglądarek, urządzeń mobilnych lub wykrywaniu sparowanych elementów.
- Budowanie systemów bezpiecznego uwierzytelniania przy użyciu standardów OAuth i OpenID Connect.
- Korzystanie z alternatywnych metod identyfikacji jako drugiego składnika uwierzytelniania.
- Wzmacnianie odporności aplikacji na potencjalny atak.
- Tworzenie systemu bezpiecznej transmisji danych przy użyciu SSL/TLS, a także symetrycznej i asymetrycznej kryptografii.

Na koniec lektury Czytelnik powinien mieć szerokie pojęcie na temat bieżącego stanu zabezpieczeń tożsamości i danych, będzie znał techniki ochrony siebie samego przed potencjalnymi atakami, a przede wszystkim będzie wiedział, jak zabezpieczyć dane swoich użytkowników.

Konwencje użyte w tej książce

W tej książce używane są następujące konwencje typograficzne:

Kursywa
: Wskazuje nowe terminy, adresy URL, adresy e-mail, nazwy plików i rozszerzenia plików.

`Stała szerokość`
: Służy do wydruków programów, a także wewnątrz akapitów do odwołań do elementów programu, takich jak nazwy zmiennych lub funkcji, bazy danych, typy danych, zmienne środowiskowe, instrukcje i słowa kluczowe.

`Stała szerokość i pogrubienie`
: Pokazuje polecenia lub inny tekst, który powinien być wpisany dokładnie tak przez użytkownika.

`Stała szerokość i kursywa`
: Pokazuje tekst, który powinien być zastąpiony wartościami podanymi przez użytkownika lub wyznaczonymi przez kontekst.

Ten element oznacza wskazówkę lub sugestię.

Ten element oznacza uwagę ogólną.

Ten element wskazuje ostrzeżenie lub przestrogę.

Podziękowania

Przede wszystkim chcielibyśmy podziękować wydawnictwu O'Reilly za opublikowanie tej książki i umożliwienie nam podzielenia się swoją wiedzą, przemyśleniami i opiniami z czytelnikami na całym świecie. Szczególne podziękowania należą się naszej redaktorce Meg Foley za jej cierpliwość i pomoc w doprowadzeniu tej pracy do końca.

Podziękować chcielibyśmy również recenzentom książki, Lenny Markusowi, Allenowi Tomowi Aaronowi Pareckiemu, którzy starannie przejrzeli rękopis i pomogli znacząco poprawić jego jakość.

Wyrazy wdzięczności należą się naszemu zespołowi projektowemu za korekty, uwagi krytyczne, a przede wszystkim za zapewnienie dostatecznej ilości czasu, abyśmy mogli pracować nad książką.

Na koniec chcemy wyrazić wdzięczność Wam, naszym Czytelnikom, za kupienie tej książki. Mamy nadzieję, że się spodoba!

Jonathan

Chciałbym zacząć od podziękowań dla mojego wspólnika zbrodni, Tima, za to, że mogłem z nim współpracować przy tej książce. Bez naszych ciągłych rozmów, tworzenia pomysłów i rozkładania ich na czynniki w celu stworzenia nowych, hybrydowych koncepcji, książka nie byłaby tym, czym jest. Twoje koncepcje, zapał i humor sprawiły, że było to wspaniałe doświadczenie.

Moja żona Heather pomogła mi zachować zdrowie umysłowe, gdy postanowiłem napisać swą pierwszą książkę blisko pięć lat temu. Pomimo tego, że trwało to wówczas tak długo, wsparłaś mnie ponownie, gdy zdecydowałem się napisać kolejną. Bez ciebie nie poradziłbym sobie z tą pracą bez popadania w szaleństwo. Wielką pomoc okazała też moja córka Scarlett – dziękuję za twoją cierpliwość i zrozumienie.

Wreszcie chciałbym podziękować mojej grupie, moim przyjaciołom. Choć podążamy odrębnymi drogami, rozproszeni po różnych firmach i różnych częściach świata, zawsze będę uważał was za najbliższych przyjaciół.

Tim

Chciałbym podziękować Jonathanowi, który jest nie tylko fantastycznym kolegą i przyjacielem, ale również wielkim współautorem tej książki. To było niezwykłe doświadczenie, ciągła wymiana idei i pomysłów i jestem przekonany, że książka ta byłaby znacznie mniej interesująca bez jego udziału.

Mojej żonie, Karin, należą się wielkie podziękowania – i zapewne jeszcze większy bukiet kwiatów – za zapewnienie mi całego tego czasu, którego potrzebowałem na ukończenie pracy.

Joe Nash, Alan Wong, Steven Cooper i Cristiano Betta stanowili fantastyczny zespół pomagający w tworzeniu i redagowaniu tej książki i zdecydowanie zasługują na to, aby o nich wspomnieć w tym miejscu.

Szczególne podziękowania należą się Danese Cooper, szefowej działu Open Source w PayPal, która jako pierwsza zachęcała mnie, aby spisać moje przemyślenia w czymś więcej, niż postach na blogu.

Na koniec chcę podziękować Johnowi Lunnowi i Taylorowi Nguyen, którzy nieustannie wspierali mnie i doradzali podczas pisania tej książki – i nie tylko.

Rozdział 1

Wprowadzenie

Jonathan LeBlanc i Tim Messerschmidt

Jedne z najważniejszych inwestycji, jakie możemy poczynić w swoim systemie, przedsiębiorstwie lub aplikacji, dotyczą infrastruktury zabezpieczeń i tożsamości. Nie ma tygodnia, byśmy nie usłyszeli o kolejnym wycieku danych klientów, przechwyconych numerach kart kredytowych lub kradzieży tożsamości. Nawet jeśli umieścimy całą serię przeszkód na drodze potencjalnego napastnika, zawsze będzie istniało prawdopodobieństwo, że nasze bazy danych zostaną naruszone, informacje skradzione, a napastnicy zaczną próbować złamać przechowywane w nich poufne dane (o ile zostały zaszyfrowane).

Nie istnieje całkowicie niezawodna, bezpieczna metoda ochrony naszych danych i tożsamości – bezpieczeństwo danych zawsze sprowadza się do ograniczania ryzyka, ochrony zabezpieczonych danych i kupowania czasu wystarczającego na podjęcie działań i zredukowania strat, gdy coś takiego przydarzy się właśnie nam.

W miarę zagłębiania się w koncepcje, techniki i metodologie programistyczne kryjące się za budowaniem bezpiecznego interfejsu dla danych i tożsamości będziemy analizować decyzje, kompromisy i kluczowe koncepcje, które trzeba zrozumieć, by móc podejmować świadome decyzje dotyczące zabezpieczeń.

Najlepszym początkiem będzie przeanalizowanie głównych problemów dotyczących bezpieczeństwa danych i tożsamości, przed którymi obecnie stoi branża informatyczna.

Problemy z obecnymi modelami zabezpieczeń

Bieżący (niezadowalający) stan bezpieczeństwa informatycznego nie polega na tym, że technika nie jest w stanie poradzić sobie z potencjalnymi kierunkami ataków. Wynika on z tego, że wybory projektowe doprowadziły nas do powstawania słabych systemów. Jednym z największych błędów, które wielu z nas stale popełnia, jest założenie, że użytkownik będzie wiedział, jak chronić swoje własne zasoby, na przykład stosując silne hasła lub uwierzytelnianie dwuskładnikowe, a nawet jeśli to zrobi, że nie będzie wybierał

najłatwiejszych rozwiązań. To my, jako projektanci, musimy chronić naszych użytkowników tak samo, jak próbujemy ochronić nasze własne systemy i musimy zakładać, że użytkownicy sami tego nie zrobią.

Oznacza to, że musimy wyrzucić z naszych przemyśleń kilka błędnych założeń:

Użytkownik zawsze posłuży się najbezpieczniejszymi opcjami Prosty fakt jest taki, że najgorsze, co możemy zrobić, to polegać na tym, że użytkownik będzie w stanie lub będzie chciał użyć opcji, która zabezpieczy jego samego i jego dane. To na właścicielu witryny lub usługi musi spoczywać brzemię odpowiedzialności za to, że dane dostarczane przez użytkownika dla jego bezpieczeństwa (takie jak hasło) są dostatecznie silne, aby zapewnić minimalny wymagany poziom zabezpieczeń (więcej informacji o szyfrowaniu danych i zabezpieczeniach zawiera rozdział 2). Dla przykładu w systemach oferujących jako opcję uwierzytelnianie dwuskładnikowe typowy poziom zainteresowania wynosi od 5 do 10% użytkowników.

Powinniśmy zawsze zapewnić większe bezpieczeństwo systemów, nawet kosztem użyteczności Jest to jedna z typowych reakcji na poprzedni punkt. Decydujemy się sprawić, by system był tak bezpieczny, jak to możliwe, kosztem użyteczności i wygody dla użytkownika. W rzeczywistości nie ma konieczności takiego kompromisu. Istnieją liczne mechanizmy, które można wdrożyć w celu podniesienia bezpieczeństwa, a które nie wpływają znacząco na doświadczenia użytkownika. Tym zagadnieniem zajmiemy się w podpunkcie „Bezpieczeństwo kontra użyteczność" w dalszej części rozdziału.

Nasze zabezpieczenia nigdy nie zostaną złamane Od startupów po wielkie korporacje, wielu inżynierów pokłada zbyt wielkie zaufanie w jakości zabezpieczeń swoich systemów. Prowadzi to do lekceważenia standardów szyfrowania danych, przez co osobiste i poufne informacje, takie jak dane kart kredytowych, adresy itp. są przechowywane jako czysty tekst, dane nieszyfrowane w żaden sposób. Gdy nastąpi włamanie, hackerzy nie muszą się wysilać, aby przejąć i wykorzystać te dane.

Zawsze należy zakładać, że dane zostaną skradzione i stosować odpowiednie szyfrowanie

W czerwcu 2015 nastąpił masowy wyciek danych instytucji rządowych Stanów Zjednoczonych, w którym ujawniono dane osobiste milionów pracowników, ponieważ dane same w sobie nie zostały zaszyfrowane (Computer World). Nieważne, jak mała lub wielka jest firma, powinniśmy zawsze zakładać, że istnieje ryzyko przełamania zabezpieczeń naszej bazy danych i kradzieży danych. Wszystkie informacje poufne powinny zawsze być odpowiednio szyfrowane.

Spróbujmy bardziej zagłębić się w niektóre z tych problemów, aby zrozumieć przyczyny i skutki wyborów dokonywanych przez nas jako projektantów.

Kiepskie hasła

Jak wspomniałem wcześniej, użytkownicy notorycznie wybierają skrajnie niebezpieczne hasła dla swoich kont. Aby to pokazać, przyjrzyjmy się hasłom, które najczęściej były stosowane w roku 2015, zgodnie z zestawieniem przygotowanym przez *SplashData* z plików zawierających miliony skradzionych haseł opublikowanych w sieci w minionym roku[1].

Tabela 1-1 *Najpopularniejsze hasła roku 2015*

1: 123456	6: 123456789	11: welcome	16: dragon	21: princess
2: password	7: football	12: 1234567890	17: master	22: qwertyuiop
3: 12345678	8: 1234	13: abc123	18: monkey	23: solo
4: qwerty	9: 1234567	14: 111111	19: letmein	24: passw0rd
5: 12345	10: baseball	15: 1qaz2wsx	20: login	25: starwars

Zanim jednak zaczniemy potępiać ludzi wybierających takie hasła, musimy zauważyć, że dane te są obarczone pewnymi wadami, które trzeba wziąć pod uwagę:

- Ponieważ większość tych danych pochodzi z wycieków informacji, można założyć, że hasła te były łatwiejsze do złamania atakiem słownikowym lub siłowym.
- Nie znamy pochodzenia większości tych danych, zatem nie możemy zweryfikować jakości zabezpieczeń stosowanych w witrynach lub usługach, z których je wykradziono.
- Dane te mogą zawierać anomalie lub po prostu błędy. Jeśli jakieś hasło jest domyślnie ustawiane przez usługę, w której następuje wiele wycieków danych (a hasło to nie jest nigdy zmieniane), znajdzie się ono na wyższej pozycji w tabeli, co jednak nie oznacza wcale, że jest powszechnie używane. Jeśli analizujemy dane z wielu źródeł używając informacji, które zostały źle przetworzone lub zawiera podobne anomalie, cała lista będzie zdeformowana.

Tym niemniej, nawet jeśli pokazane hasła mogą występować w mniejszej liczbie, niż sugeruje powyższa lista, a dane mogą być mocno „skrzywione", nadal istnieją. Oznacza to, że przy budowaniu systemu zabezpieczeń dla danych i tożsamości musimy zapewnić odpowiedni poziom ochrony ludziom, którzy wybierają takie hasła. Typowo będziemy chcieli tworzyć konstrukcję eliminującą najsłabsze elementy systemu uwierzytelniania.

Pod wieloma względami właśnie stąd wynikają nasze oczekiwania co do haseł tworzonych przez użytkowników: używania różnych wielkości liter, co najmniej jednego symbolu oraz cyfry, tworzenia ciągów znaków nierozpoznawalnych w słownikach lub możliwych do wydedukowania na podstawie znajomości danej osoby. Tego typu wymagania powodują jednak, że system jest mało wygodny dla użytkownika i nie będzie on w stanie zapamiętać hasła, co sprawi, że będzie szukał najprostszego sposobu wejścia lub po prostu zapisze

1 *http://www.teamsid.com/worst-passwords-2015/*

hasło na karteczce przylepionej do monitora. Użyteczność (a może lepiej byłoby napisać *użytkowalność*) musi być częścią projektu systemu zabezpieczeń, aby był on skuteczny.

Bezpieczeństwo kontra użyteczność

> Jeśli zabezpieczenia będą miały zbyt duży priorytet względem wygody, używanie witryny stanie się trudne lub niemożliwe.
>
> *– Anthony T, założyciel UX Movement*

Naszym głównym celem podczas obsługiwania danych użytkowników jest zapewnienie bezpieczeństwa tych danych i ich tożsamości, ale jednocześnie nie chcielibyśmy zniechęcać ich wszystkich używając zbyt złożonego systemu logowania. Jeszcze gorsze mogłoby być wymuszanie wieloekranowego, pracochłonnego procesu rejestracji przy kupowaniu produktów lub ciągłe żądanie od użytkownika podawania szczegółów identyfikacyjnych w trakcie korzystania z serwisu. To gwarantowane sposoby zapewnienia, że użytkownik nigdy nie wróci.

Niektóre z głównych powodów wskazywanych przez użytkowników, którzy porzucają swoje koszyki i rezygnują z zakupów, obejmują niewygodę samego procesu zakupowego (jest zbyt złożony lub długotrwały), a także wymóg zalogowania się/rejestracji przed zakupem. Wiele z tych zagadnień można rozwiązać dzięki rozwiązaniom zwiększającym użyteczność, takim jak pojedyncza strona logowania i umożliwienie uproszczonej rejestracji gości.

Zagadnienie „bezpieczeństwo czy użyteczność" jest zawsze sprawą kompromisu. Musimy zapewnić, że mamy wystarczająco wysoką pewność co do bezpieczeństwa naszych użytkowników, a jednocześnie zrobić to w tak dyskretny i przezroczysty sposób (w tle), aby nie niszczyć ich doświadczeń ciągłym wymaganiem weryfikacji.

Możemy sobie w tym momencie postawić następujące pytania:

- Czy możemy uzyskać informacje o tożsamości, zwiększające nasze przekonanie, że użytkownik jest tym, za kogo się podaje, bez nakładania na niego dodatkowych wymogów weryfikacji?
- Czy jeśli mamy wysokie przekonanie, że użytkownik jest tym, za kogo się podaje, możemy zbudować bardziej użyteczne środowisko dla tego użytkownika w porównaniu z takim, dla którego brak takiej pewności?
- Jaka zawartość wymaga identyfikacji użytkownika i kiedy powinienem zastosować dodatkowe poziomy zabezpieczeń, aby ją zweryfikować?

Koncepcje te przeanalizujemy głębiej w rozdziale 3, gdy będziemy omawiać strefy zaufania i budowanie informacji weryfikujących tożsamość dla użytkownika.

Niewłaściwe szyfrowanie danych

Problematyka bezpieczeństwa danych i identyfikowania użytkowników nie polega na planowaniu na najlepszy możliwy wariant, ale na najgorszy. Jeśli istnieje prawdopodobieństwo, że coś może się wydarzyć, należy zakładać, że się wydarzy i mieć na taką okoliczność gotowy plan, pozwalający zmniejszyć lub ograniczyć powstałe szkody.

27 marca 2015 Slack ogłosił, że ich systemy zostały przełamane i skradziono dane użytkowników. Straty wynikające z tego incydentu zostały jednak ograniczone dzięki stosowanym metodom silnego szyfrowania danych. Jak można przeczytać na firmowym blogu, „Slack utrzymuje centralną bazę danych użytkowników, która zawiera nazwy użytkowników, ich adresy email oraz jednokierunkowo zaszyfrowane (haszowane) hasła. Używana przez Slack funkcja haszująca to bcrypt, z losowo generowanym komponentem salt dla każdego hasła, co oznacza, że odtworzenie haseł z tej postaci jest obliczeniowo niewykonalne". Dodatkowo po tym zdarzeniu firma wprowadziła uwierzytelnianie dwuskładnikowe oraz przełącznik wygaszania haseł dla właścicieli zespołów, który pozwalał im automatycznie wylogować wszystkich użytkowników ze wszystkich urządzeń i wymusić utworzenie nowych haseł.

W tym przypadku szyfrowanie danych oraz szybka reakcja uniemożliwiły masową kradzież kont użytkowników i zmniejszyły straty wizerunkowe oraz spadek zaufania użytkowników. Szyfrowanie danych nie musi jedynie służyć ochronie danych przed kradzieżą. Pozwala też spowolnić działania hackerów, aby móc podjąć odpowiednie przeciwdziałanie, a w części przypadków rozszyfrowywanie danych jest wystarczająco długotrwałe, aby działanie takie straciło sens.

Najsłabsze ogniwo: ludzie

Naszym najważniejszym zadaniem, jako projektantów czy dostawców usług, powinno być traktowanie danych naszych użytkowników z największą troską, jaką możemy zapewnić. Z tego względu staramy się zabezpieczyć dowolne rodzaje informacji dostarczane przez użytkowników, wykorzystując algorytmy szyfrowania, oferując bezpieczne metody komunikacji i ciągle wzmacniając naszą infrastrukturę w nieustannych wysiłkach.

Najważniejszy element tego łańcucha, człowiek, często jest pomijany w równaniu, przez co tworzone aplikacje czy usługi bywają otwarte na zagrożenia, które w ogóle nie były rozważane przy projektowaniu i wdrażaniu zabezpieczeń. Prawda jest następująca: ludzie mają skłonność do wybierania najłatwiejszych dróg. Dla nas oznacza to, że ludzie będą wybierać łatwe do zapamiętania i krótkie hasła, proste do odgadnięcia nazwy użytkowników i nie będą mieli pojęcia o nowoczesnych technikach uwierzytelniania, takich jak uwierzytelnianie dwuskładnikowe (nazywane czasem 2FA od angielskiego terminu *two-factor authentication*). Technikę tę omówimy szczegółowo w piątym rozdziale tej książki – zdecydowanie zasługuje ona na większą uwagę i skupienie. Przeanalizujemy tam również wywodzącą się z 2FA technikę nazywaną po prostu *uwierzytelnianiem*

n-składnikowym, która reprezentuje skalowane podejście do zabezpieczeń, zależne od konkretnego zastosowania.

Łatwo można zrozumieć, dlaczego ludzie mają skłonność do używania, a zwłaszcza ponownego używania łatwych haseł – oszczędza im to czas przy tworzeniu profili użytkowników i sprawia, że uwierzytelnianie w usługach i aplikacjach jest zadaniem prostym i szybkim. Zwłaszcza w przypadku usług mobilnych często mamy do czynienia z niewielkim ekranem i dotykowymi klawiaturami, które sprawiają dodatkowe kłopoty.

Zjawisko to znane jest jako zmęczenie hasłami. Szczęśliwie istnieje wiele narzędzi, które my, jako projektanci, możemy wykorzystać w celu poradzenia sobie z tymi problemami i zapewnić płynny i wygodny przebieg rejestracji i uwierzytelniania w naszych aplikacjach, nadal zapewniając bezpieczeństwo użytkowników.

Wiele systemów operacyjnych, przeglądarek lub aplikacji próbuje rozwiązać problem zmęczenia hasłami, pozwalając na generowanie losowych haseł i jednocześnie oferując metodę przechowywania tych haseł pod kontrolą pojedynczego hasła głównego.

Popularnym przykładem może być aplikacja zarządzania hasłami *Keychain*, wprowadzona w systemie Mac OS 8.6. Keychain jest głęboko zintegrowany z OS X, a obecnie również z iOS (za pośrednictwem iCloud) i pozwala przechowywać różne rodzaje danych, w tym numery kart kredytowych, hasła i klucze prywatne.

Coraz liczniejsze usługi, takie jak 1Password, Dashlane lub LastPass, oferują generowanie haseł dla swoich użytkowników. Usuwa to potrzebę samodzielnego wymyślenia bezpiecznego hasła i jest często postrzegane jako wygodna metoda przyśpieszenia rejestracji konta użytkownika.

Katie Sherwin, członek Nielsen Norman Group, opublikowała artykuł[2] o upraszczaniu przebiegu uwierzytelniania hasłem i wyliczyła trzy poniższe podejścia jako sposoby poprawienia wrażeń użytkownika:

- Pokaż reguły
- Pokaż to, co wpisuje użytkownik
- Pokaż miernik siły hasła

Dzięki zastosowaniu tych reguł możemy zagwarantować, że użytkownik będzie się czuł pewnie co do używanych przez siebie haseł i otrzyma jasny sygnał, jak są silne. Dalsze badanie wskazują, że użytkownicy widzący miernik siły wybierają bardziej bezpieczne hasła – nawet jeśli miernik ten nie jest zbyt dobrze zaimplementowany[3].

2 *http://www.nngroup.com/articles/password-creation*

3 *http://research.microsoft.com/pubs/227130/WhatsaSysadminToDo.pdf*

> Ci, którzy widzą miernik, wykazują skłonność do wybierania silniejszych haseł od tych, którzy takiego miernika nie mieli, ale typ samego miernika nie robi znaczącej różnicy[4].
>
> *—Dinei Florencio, Cormac Herley i Paul C. van Oorschot, w publikacji „An Administrator's Guide to Internet Password Research"*

Pojedyncze logowanie

Mechanizm pojedynczego logowania (*single sign-on*, SSO) polega na wykorzystaniu istniejącego konta użytkownika do uwierzytelniania w wielu różnych usługach. Koncepcja sprowadza się do wypełnienia i zabezpieczenia centralnego konta użytkownika zamiast zmuszania go do rejestrowania się ciągle w kolejnych usługach i systemach.

Często spotykane rozwiązania próbujące wykorzystać ponowne użycie profilu użytkownika, aby bądź udostępnić informacje profilu, bądź po prostu zapewnić uwierzytelnienie w innych usługach, obejmują OpenID, OAuth 1.0, OAuth 2.0 oraz różne modele hybrydowe, takie jak OpenID Connect. W rozdziale 4 zajmiemy się wybranymi technikami uwierzytelniania i omówimy zarówno techniczne szczegóły implementacji, jak i implikacje dla zabezpieczeń.

Pojęcie entropii a bezpieczeństwo haseł

Zanim zagłębimy się w szczegóły, musimy najpierw rozstrzygnąć, jak można odróżnić hasło słabe od silnego, gdy hasło to jest tworzone przez człowieka. Standardowy mechanizm ustalania siły hasła wykorzystuje pojęcie „entropii informacji", która jest mierzona liczbą bitów informacji w dostarczonym źródle, takim jak hasło.

Typowo przy stosowaniu fraz hasłowych dobry poziom entropii powinien wynosić (jako minimum) 36,86 bitów, co odpowiada średniemu poziomowi entropii trzech losowych słów wybranych z listy 5000 dostępnych, unikatowych wyrazów.

Entropia hasła jest po prostu miarą tego, jak bardzo nieprzewidywalne jest to hasło. Miara ta zależy od kilku kluczowych cech:

- Użyty zbiór symboli.
- Rozszerzenie tego zbioru symboli dzięki rozróżnianiu małych i wielki liter.
- Długość hasła.

4 *http://research.microsoft.com/pubs/227130/WhatsaSysadminToDo.pdf*

Przy użyciu powyższych informacji można wykorzystać entropię hasła (wyrażoną liczbą bitów) do przewidzenia, jak trudne będzie złamanie tego hasła poprzez odgadnięcie, atak słownikowy, atak siłowy (*brute force*) itp.

Przystępując do ustalania ogólnej entropii hasła można rozróżnić dwie główne metody generowania haseł, którymi należy się zająć: hasła tworzone losowo (generowane przez komputer) oraz hasła wybierane przez ludzi.

Zgodnie z badaniem zatytułowanym „A Large-Scale Study of Web Password Habits" autorstwa Dinei Florencio i Cormaca Herley'a z Microsoft Research, poziom entropii przeciętnego hasła można oszacować na 40,54 bitów[5].

Entropia losowo generowanych haseł

Jeśli przyjrzymy się entropii losowo wybieranych haseł (generowanych przez komputer), proces ustalenia ogólnej entropii jest stosunkowo prosty, gdyż nie trzeba uwzględniać wpływu elementu ludzkiego. Zależnie od wybranego zbioru symboli, z którego będziemy budować hasło, można łatwo utworzyć serię haseł o pożądanym poziomie entropii.

Powszechnie stosowany wzór używany do obliczania entropii informacji (w naszym przypadku hasła) to:

$$H = \log_2 (b^l)$$

gdzie

- H = Entropia hasła wyrażona w bitach
- b = Liczba dostępnych znaków w zbiorze symboli (moc zbioru symboli)
- l = Liczba symboli użytych w haśle (długość hasła)

Aby określić wartość b, możemy po prostu wybrać odpowiedni zbiór symboli spośród pokazanych w tabeli 1-2.

Tabela 1-2 *Entropia pojedynczego znaku z poszczególnych zbiorów symboli*

Nazwa zbioru symboli	Liczba znaków w zbiorze	Entropia na symbol (w bitach)
Cyfry arabskie (0-9)	10	3,332
Cyfry szesnastkowe (0-9, A-F)	16	4,000
Alfabet łaciński bez rozróżniania wielkości liter (a-z lub A-Z)	26	4,700
Znaki alfanumeryczne bez rozróżniania wielkości liter (0-9, A-Z lub a-z)	36	5,170
Alfabet łaciński z rozróżnianiem wielkości liter (A-Z, a-z)	52	5,700

5 *http://research.microsoft.com/pubs/74164/www2007.pdf*

Tabela 1-2 *Entropia pojedynczego znaku z poszczególnych zbiorów symboli*

Nazwa zbioru symboli	Liczba znaków w zbiorze	Entropia na symbol (w bitach)
Znaki alfanumeryczne z rozróżnianiem wielkości liter (A-Z, a-z, 0-9)	62	5,954
Wszystkie drukowalne znaki ASCII	95	6,570
Wszystkie drukowalne znaki rozszerzonego ASCII	218	7,768
Binarne znaki (0-255, 8-bitowy bajt)	256	8,000
Losowo wybierana lista wyrazów (*diceware*)	7776	12,925

Spośród powyższych zbiorów symboli mniej znanym może być losowo wybierana lista wyrazów (*diceware*). Metoda ta polega na użyciu zwykłej kostki do gry (ang. *dice, stąd nazwa techniki*) i wyrzuceniu jej pięć razy. Uzyskane wartości tworzą pięciocyfrową liczbę (np. 46231, gdzie cyfry odpowiadają poszczególnym wynikom). Liczba ta służy do wyszukania określonego słowa z wstępnie zdefiniowanej listy. Istnieje 7776 możliwych wartości, zatem lista powinna zawierać tyle właśnie wyrazów. Warto zauważyć, że jest to więcej, niż liczba słów używanych na co dzień przez większość (nawet wykształconych) ludzi. Szersze omówienie (w języku angielskim) oraz przykładowe listy wyrazów z różnych języków dla metody *diceware* można znaleźć pod adresem *http://world.std.com/~reinhold/diceware.html*.

Używając przedstawionego wcześniej wzoru i znając długość hasła oraz liczbę symboli w wybranym zbiorze można oszacować liczbę bitów entropii losowo wygenerowanego hasła.

Entropia haseł tworzonych przez ludzi

Zanim zaczniemy próbować mierzyć poziom entropii hasła, które zostało utworzone przez człowieka, a nie wygenerowane losowo zgodnie ze standardami zabezpieczeń, musimy uświadomić sobie, że obliczenia te nie są trywialne. Zaproponowano wiele różnych metod realizacji tego zadania (NIST, Shannon Entropy, Guessing Entropy itd.), ale wszystkie one wykazują pewne wady i niedociągnięcia.

Metoda Shannona wydaje się zwracać zbyt optymistyczną ocenę bezpieczeństwa hasła (a zarazem nie daje praktycznych wskazówek jego poprawienia), zaś metoda NIST jest niedokładna (choć konserwatywna, czyli zwraca niedoszacowania). Ponieważ zawsze powinniśmy kierować się stronę większego bezpieczeństwa, a nie większego ryzyka, przyjrzyjmy się krótko opracowaniu NIST na temat mierzenia haseł wybieranych przez ludzi, co powinno dać nam dobry punkt wyjścia do dalszych rozważań.

Zgodnie z artykułem NIST oznaczonym jako 800-63-2, jeśli mamy hasło wybrane przez człowieka, możemy zmierzyć jego przybliżoną entropię zgodnie z poniższym algorytmem[6]:

- Entropia pierwszego znaku wynosi 4 bity.
- Entropia kolejnych 7 znaków wynosi po 2 bity na znak (autorzy stwierdzają, że jest to „w przybliżeniu zgodne z oszacowaniem Shannona, że efekty statystyczne dla ciągów mających nie więcej niż 8 znaków mają entropię około 2,3 bitów na znak").
- Znaki od 9 do 20 mają entropię wynoszącą 1,5 bitów na znak.
- Kolejne znaki (21. i dalej do końca hasła) mają entropię 1 bit na znak.
- Dodajemy 6-bitowy bonus do hasła spełniającego reguły wymagające stosowania różnych wielkości liter i znaków niealfabetycznych (to również jest dość pesymistyczne oszacowanie, jako że autorzy publikacji NIST zauważają, że owe znaki specjalne najczęściej pojawiają się na początku lub na końcu hasła, co redukuje całkowitą przestrzeń przeszukiwania).
- Dodatkowy 6-bitowy bonus otrzymują hasła o długości od 1 do 19 znaków, które spełniają warunki zaawansowanego testu słownikowego w celu sprawdzenia, że hasło nie jest wyrazem występującym w (obszernym) słowniku. Powodem, dla którego hasła dłuższe niż 20 znaków nie otrzymują tego bonusu, jest to, że zakłada się, że składają się z one z wielu wyrazów słownikowych, czyli są frazą hasłową.

Spróbujmy zastosować te reguły do obliczenia entropii kilku przykładowych haseł:

- ***monkey (6 znaków) = 14 bitów entropii*** (4 bity za pierwszy znak, 10 za pozostałych 5 znaków)
- ***Monkey1 (7 znaków) = 22 bity entropii*** (4 bity za pierwszy znak, 12 za pozostałych 6 znaków, bonus 6 bitów za użycie wielkiej litery i znaku niealfabetycznego)
- ***tvMD128!Rrsa (12 znaków) = 36 bitów entropii*** (4 bity za pierwszy znak, 14 bitów za kolejnych 7 znaków, 6 za następne 4 znaki, bonus 6 bitów za użycie wielkich liter i znaków niealfabetycznych, 6 bitów za niesłownikowy ciąg w pierwszych 19 znakach)
- ***tvMD128!aihdfo#Jh43 (19 znaków) = 46,5 bitów entropii*** (4 bity za pierwszy znak, 14 bitów za kolejnych 7 znaków, 16,5 za kolejnych 11 znaków, bonus 6 bitów za użycie wielkich liter i znaków niealfabetycznych, 6 bitów za niesłownikowy ciąg w pierwszych 19 znakach)
- ***tvMD128!aihdfo#Jh432 (20 znaków) = 42 bity entropii*** (4 bity za pierwszy znak, 14 bitów za kolejnych 7 znaków, 18 za kolejnych 12 znaków, bonus 6 bitów za użycie wielkich liter i znaków niealfabetycznych)

6 *http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf*

Można tu zauważyć luki w założeniach poczynionych przez autorów opracowania NIST. Pokazują to dwa ostatnie przykłady, gdzie jeden dodatkowy znak spowodował utratę bonusu 6 bitów entropii z powodu przypuszczenia, że hasło o tak dużej długości nie może być złożonym ciągiem, ale jest zbudowane ze sklejonych ze sobą słów pochodzących ze słownika, na przykład „drzewomanicurekuratornado". Zgodnie z wzorem NIST hasło to powinno dać 41 bitów entropii. Hasło to jest przykładem zastosowania techniki *diceware* i zgodnie z wcześniejszym podrozdziałem ma entropię 12,9 x 4, czyli 51,6 bitów.

W miarę zagłębiania się w to zagadnienie można zauważyć, dlaczego ustalenie siły (bezpieczeństwa) hasła tworzonego przez człowieka może być trudne, a wynika to z faktu, że ludzie są nieprzewidywalni. Jeśli do wymagań bezpieczeństwa dołączymy system haseł generowanych przez komputer i będziemy przechowywać te hasła w aplikacji zarządzania hasłami, takiej jak 1Password, KeePass lub LastPass, uzyskamy bardzo przewidywalne środowisko. To właśnie z tego powodu zazwyczaj wybieramy jedno z dwóch podejść (niekiedy obydwa) zabezpieczania tożsamości w projektach sieciowych:

1. Przy tworzeniu hasła przez użytkownika stawiamy mu wymagania dotyczące siły zabezpieczenia. Mogą one dotyczyć długości, użycia znaków niealfabetycznych, małych i wielkich liter, wyrazów niesłownikowych itp. Z oczywistych względów takie rozwiązanie jest niewygodne i może zniechęcać wielu użytkowników, ale poziom bezpieczeństwa wzrasta. Typowy problem polega na tym, że wraz z utrudnianiem tworzenia haseł zwiększamy prawdopodobieństwo, że użytkownik je zapomni, co wymaga zastosowania procedury odzyskiwania hasła (którą musimy odrębnie przygotować i która sama w sobie jest elementem osłabiającym bezpieczeństwo systemu).
2. Próbujemy zabezpieczyć dane jak najlepiej „z naszej strony" (w sposób niewidoczny lub mało angażujący użytkowników). Obejmuje to zazwyczaj szyfrowanie, solenie (dołączanie ciągów zaburzających) i wzmacnianie kluczy, aby powstrzymać złamanie słabych haseł (wszystkimi tymi koncepcjami zajmiemy się w rozdziale 2). Przy stosowaniu takiego rozwiązania zazwyczaj spotykamy również mechanizm pozwalający na tylko kilka prób nieudanego logowania, zanim konto zostanie zablokowane, aby wykluczyć potencjalny atak siłowy. To rozwiązanie jest lepsze z punktu widzenia użytkowalności, gdyż użytkownicy mogą w praktyce wybrać dowolne hasło, ale zmniejsza ogólne bezpieczeństwo ich kont.

Tak więc wróciliśmy do problemu „użytkowalność kontra bezpieczeństwo" i prawda jest taka, że w scenariuszu idealnym właściwe rozwiązanie leży gdzieś pośrodku. Pamiętajmy, te dwa zagadnienia wykluczają się wzajemnie.

Nazwa użytkownika i hasło – analiza

Kolejnym ważnym krokiem na drodze do zrozumienia koncepcji systemu identyfikacji jest stwierdzenie, czym są nazwa użytkownika i hasło. Mówiąc w uproszczeniu, jest to identyfikacja osoby (nazwa użytkownika lub klucz publiczny) oraz weryfikacja tego faktu przy użyciu czegoś, co tylko ta osoba powinna wiedzieć (hasło lub klucz prywatny).

Mając to na uwadze, mamy do wyboru dwie drogi myślenia o zarządzaniu danymi w systemie uwierzytelniającym:

Wzmocnić system Ten przypadek występuje, gdy mamy istniejący (lub nowy) system zbudowany na tradycyjnym rozwiązaniu z nazwą użytkownika i hasłem i chcemy zwiększyć jego bezpieczeństwo.

Usunąć nazwę użytkownika i hasło W nowych i innowacyjnych rozwiązaniach technicznych koncepcje nazwy i hasła nadal są obecne, ale są realizowane w odmienny sposób.

W miarę lektury kolejnych rozdziałów okaże się, że nasze główne cele sprowadzają się do wykorzystania jednej z tych koncepcji i skupieniu się na wzmocnieniu systemu albo znalezieniu nowej metodologii budowania mechanizmów ochrony tożsamości i danych przy użyciu nowych narzędzi i technik.

Zabezpieczanie istniejących standardów dla ochrony tożsamości

Zwiększenie bezpieczeństwa istniejącego systemu jest zwykle zadaniem większości z nas, jako że zazwyczaj budujemy nowe rozwiązania na bazie już istniejących. Również w przypadku zupełnie nowych produktów użycie nazwy użytkownika i hasła może być preferowanym mechanizmem logowania ze względu na nawyki użytkowników.

Jak pokazaliśmy wcześniej w tym rozdziale, użytkownicy są zazwyczaj ostatnimi osobami, którym należałoby powierzyć odpowiedzialność za ich własne bezpieczeństwo. Znakomita większość będzie wybierać hasła, które łatwo zapamiętać, co zazwyczaj stanowi przeciwieństwo tego, co tradycyjnie uważamy za „bezpieczne hasło".

Z poprzednich podrozdziałów wiemy już, jak oszacować przewidywalność hasła i że powinniśmy zawsze konstruować system zabezpieczeń pod kątem najsłabszego elementu łańcucha, a nie przeciętnego.

Mając to na uwadze, można wymienić określone standardowe mechanizmy, których możemy używać dla zapewnienia bezpieczeństwa kont oraz takie, których należy unikać.

Dobre i złe algorytmy zabezpieczeń

Nie wszystkie algorytmy szyfrowania są równe, gdy chodzi o bezpieczeństwo naszych danych i poufnych informacji użytkowników. Niektóre stworzono pod kątem prędkości, aby szybko i dokładnie szyfrować i rozszyfrowywać wielkie ilości danych, zaś inne zaprojektowano z premedytacją, aby były powolne. Przyjmijmy, że nasza baza danych z milionem zaszyfrowanych rekordów użytkowników została skradziona i napastnik próbuje złamać szyfrowanie, na przykład wypróbowując każdy wyraz ze słownika, aby odzyskać dane. Czy będziemy preferowali, aby można było zrealizować to tak szybko, jak się da, czy przeciwnie, jak najwolniej? Poprawna odpowiedź brzmi, że będziemy chcieli, aby proces ten był tak powolny, jak to możliwe.

Przy stosowaniu zwykłych funkcji haszujących napastnik może testować setki milionów haseł na sekundę. W przypadku algorytmów haszujących dla haseł, zależnie od konfiguracji, napastnik może być w stanie odgadnąć tylko kilkaset lub kilka tysięcy haseł w tym samym czasie, co czyni ogromną różnicę.

Dobre

Poniższa (niekompletna) lista przedstawia algorytmy haszujące, które zaprojektowano pod kątem zabezpieczania haseł i które świadomie zbudowano jako powolne, aby utrudnić łamanie szyfrowania.

PBKDF2 Akronim ten oznacza „Password-Based Key Derivation Function 2" (funkcja generowania klucza oparta na haśle nr 2). Algorytm powstał w RSA Laboratories. Stosuje on pseudolosową funkcję, taką jak skrót, szyfr lub HMAC, do danych wejściowych (hasła) wraz z ciągiem zaburzającym (solą). Proces jest powtarzany wielokrotnie w celu utworzenia klucza pochodnego.

bcrypt Utworzony przez Nielsa Provosa i Davida Mazièresa, bcrypt powstał w ramach projektu „Belgian Fundamental Research in Cryptology and Information Security". Jest to funkcja tworzenia klucza pochodnego oparta na szyfrowaniu Blowfish. W procesie tworzenia klucza używana jest sól, a ponadto występuje tu interesująca funkcjonalność adaptacyjna. Z upływem czasu licznik iteracji może być zwiększany, aby spowolnić działanie, dzięki czemu algorytm pozostaje odporny na atak siłowy.

scrypt Utworzony przez Colina Percivala scrypt jest inną funkcją budowania klucza zaprojektowaną w celu zapewnienia odporności na wielkoskalowe ataki sprzętowe poprzez wymaganie wielkich ilości pamięci, a tym samym spowolnienie obliczeń.

Złe (dla haseł)

Są to standardowe algorytmy haszujące, które z założenia powinny być szybkie. W przypadku haseł nie jest to właściwe podejście, jako że spowolnienie algorytmu sprawia, że napastnikowi znacznie trudniej jest złamać dane.

MD5 MD5, inaczej „Message-digest algorithm", został zaprojektowany przez Roberta Rivesta w roku 1991 i tworzy 128-bitową wartość skrótu, zazwyczaj przedstawianą jako 32-cyfrową liczbę szesnastkową.

SHA-1 SHA oznacza „Secure Hash Algorithm". Został zaprojektowany przez NSA (Narodową Agencję Bezpieczeństwa Stanów Zjednoczonych). SHA-1 tworzy wartość skrótu o długości 160 bitów (20 bajtów). Wartość ta jest zazwyczaj przedstawiana jako 40-cyfrowa liczba szesnastkowa.

SHA-2 Również zaprojektowany przez NSA, SHA-2 jest następcą SHA-1 i składa się z sześciu funkcji haszujących, zwracających wartości skrótu o długości 224, 256, 384 lub 512 bitów (odpowiednio SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256).

Jakie dane powinny być chronione?

Pewne sugestie, jak należy na to odpowiedzieć, przedstawiliśmy już kilkukrotnie w tym rozdziale. Jeśli wprost postawimy sobie pytanie „Jakie informacje bezwzględnie muszą być szyfrowane?", odpowiedź jest bardzo prosta: wszystko, co umożliwia identyfikację osoby (dane tożsamości, informacje osobiste, szczegóły płatności) oraz cokolwiek, co jest niezbędne w naszym systemie, czego ujawnienie mogłoby otworzyć dodatkowe luki lub wycieki w naszej architekturze.

Mechanizmy odzyskiwania kont a socjotechnika

Po ustaleniu tego, jakie szczegóły są warte ochrony, powinniśmy wziąć tę wiedzę pod uwagę przy analizowaniu mechanizmów odzyskiwania. Słabo zaprojektowane mechanizmy odzyskiwania w połączeniu z socjotechniką często prowadzą do ujawnienia informacji – nawet wtedy, gdy zaimplementowane zostały mechanizmy mające chronić przed właśnie takim zdarzeniem. Jeśli Czytelnik zna to zagadnienia, może pominąć ten fragment rozdziału.

Popularne przykłady obejmują sytuacje, gdy pomoc techniczna udostępniała komuś szczegóły konta, których nie powinna nikomu przekazywać, a także źle zaprojektowane procedury odzyskiwania (resetowania) hasła. Przejęte konto poczty elektronicznej może zapewnić łatwy dostęp do konta użytkownika. Zabezpieczenie naszych użytkowników poprzez stosowanie „pytań bezpieczeństwa" i wymaganie podania właściwych odpowiedzi może pomóc w obniżeniu ryzyka wycieków informacji.

> Inżynieria społeczna (socjotechnika) to nietechniczna metoda włamania używana przez hackerów, która polega głównie na interakcji z ludźmi i skłanianiu ich do naruszenia zwykłych procedur bezpieczeństwa. Jest to jedno z największych zagrożeń, przed którym stoją dzisiejsze organizacje.[7]
>
> – *TechTarget SearchSecurity*

Problem pytań bezpieczeństwa

Podczas gdy ogólna wiedza i świadomość ważności bezpiecznych haseł stale wzrasta, inny wrażliwy obszar – pytania bezpieczeństwa – jest często ignorowany. Zamiast zaproponowania użytkownikowi szerokiego wyboru osobistych pytań lub wręcz pozwalania im na zdefiniowanie własnego pytania bezpieczeństwa, wiele rozwiązań zawiera proste, typowe zwroty, na których odpowiedzi można łatwo znaleźć, na przykład przeglądając profile użytkownika w mediach społecznościowych.

Pytania bezpieczeństwa często się powtarzają, a niekiedy stanowią niezamierzenie komediowe zestawienia, na które niełatwo odpowiedzieć i trudno zapamiętać („Jakie było twoje ulubione danie w dzieciństwie?", „Jaka jest twoja ulubiona książka?"). Soheil Rezayazdi opublikował w magazynie McSweeney's Internet Tendency listę „Nihilistycznych pytań bezpieczeństwa", która powinna wywołać przynajmniej lekki uśmiech – oto nasza osobista „pierwsza piątka" [8]:

1. Kiedy przestałeś próbować?
2. W którym roku porzuciłeś swoje marzenia?
3. W jakim byłeś wieku, gdy uciekł twój ukochany zwierzak?
4. Jak nazywał się twój ulubiony bezpłatny staż?
5. Jak brzmi imię twojego najmniej ulubionego dziecka?

Mówiąc poważnie jednak, wpływ socjotechniki jest często zupełnie niedoceniany lub wręcz ignorowany. Bardzo często łatwiejsze jest wejście przez bramę, niż jej obchodzenie i wyłamywanie barier. Działania inżynierii społecznej mogą rozciągać się od przeglądania faktów dostępnych online na temat jakiejś osoby, aż po wśliźnięcie się do budynku biurowego; choć to ostatnie brzmi przesadnie (i nie zdarza się zbyt często), sensowne wydaje się przygotowanie personelu na taką okoliczność.

Jeśli ktoś szuka więcej informacji na ten temat, doskonałym źródłem dotyczącym socjotechniki są książki Kevina Mitnicka „Duch w sieci", „Sztuka inflitracji" oraz „Sztuka podstępu"[9].

---

7 *http://searchsecurity.techtarget.com/definition/social-engineering*

8 *http://www.mcsweeneys.net/articles/nihilistic-password-security-questions*

9 Kevin Mitnick zdobył ogólny rozgłos, hackując takie firmy, jak Nokia i Pacific Bell. Obecnie jest aktywnym konsultantem w dziedzinie bezpieczeństwa sieci.

Co dalej?

Teraz, gdy poznaliśmy już wszystkie koncepcje, których będziemy używać i omawiać w dalszej części książki, możemy przejść do następnego rozdziału, w którym zajmiemy się tym, jak haszowanie, solenie i szyfrowanie danych można włączyć do naszych systemów.

Rozdział 2

Hasła: szyfrowanie, haszowanie i solenie

Jonathan LeBlanc

W pierwszym rozdziale poznaliśmy podstawowe koncepcje bezpieczeństwa haseł i bieżący stan standardów wykorzystywanych w branży informatycznej. Zacznijmy teraz stosować tę wiedzę w praktyce, przyglądając się realnemu stosowaniu szyfrowania haseł i zabezpieczeń.

Aby rozpocząć taką analizę z punktu widzenia wdrożeń, zastanówmy się najpierw nad różnymi technikami przesyłania i przechowywania danych, które powinniśmy wziąć pod uwagę.

Dane w spoczynku kontra dane w ruchu

Na początku badania zabezpieczania danych musimy uwzględnić dwie ważne koncepcje, którymi trzeba się zająć: dane w ruchu oraz dane w spoczynku.

Gdy mówimy o danych w spoczynku, rozumiemy przez to nieaktywne (spoczywające) dane cyfrowe przechowywane na naszych serwerach, takie jak bazy danych, których używamy do przechowywania haseł, informacji profilowych i dowolnych innych szczegółów potrzebnych w naszej aplikacji.

Gdy mowa o danych w ruchu, mamy na myśli dowolne dane w trakcie przesyłania, wysyłane tam i z powrotem pomiędzy aplikacją a bazą danych lub pomiędzy witrynami a zewnętrznymi źródłami danych.