Proszę pamiętać: slajdy te w założeniu nie służą jako źródło wiedzy, a są udostępniane, żeby można było zobaczyć zakres materiału.

1 Hasła

1.1 Trzy filary identyfikacji

Trzy filary identyfikacji

- Coś co mam.
- Coś co wiem.
- Coś czym jestem.

Hasła

- Sekret (coś, co wiemy w teorii znany tylko użytkownikowi.
- Używane od starożytności do kontroli dostępu.
- Jedna z niewielu rzeczy, jakie człowiek we współczesnej kryptologii może zrobić bez asystyk komputera.

Odzew

- Ponieważ hasło podaje się temu, kto zapyta, odzew jest potrzebny, żeby się dowiedzieć, czy nie przekazaliśmy hasła wrogowi.
- Hasło łatwo podsłuchać.
- Hasło musi znać ktoś poza podającym.

2 Podstawowe ataki

2.1 Podsłuch

Podsłuchiwanie haseł

- Hasło może być podsłuchane w transmisji lub podczas procesu wprowadzania.
- Jak rozwiązać problem?

Podsłuchiwanie haseł

- Zasłaniamy hasła na monitorze.
- Komunikujemy się bezpiecznym kanałem.
- Przed prośbą o hasło, udowadniamy, że mamy prawo o nie prosić.

2.2 Zgadnięcie hasła

Atak brutalną siłą

- W niektórych sytuacjach, szybkość z jaką możemy zgadywać hasła jest bardzo duża.
- Możemy próbować wszystkie możliwe kombinacje haseł.
- Jaką entropię ma np. 8-znakowe hasło?
- Jak się przed tym zabezpieczyć?

Atak brutalną siłą

- Ograniczanie szybkości zgadywania – captcha, spowolnienie, etc.
- Blokowanie konta umożliwia denial of service.
- Wymuszanie zmian co jakiś czas nie jest skuteczne.

2.3 Zapisane hasła

Zapisywanie haseł

- Żeby sprawdzić, czy użytkownik podaje właściwe hasło, trzeba je z czymś porównać.
- Przechowywane hasła mogą być celem ataku.
- Wyciek haseł powoduje ich ujawnienie – jedna z gorszych rzeczy jaka się może przytrafić.
- Jak się przed tym bronić?

Zapisywanie haseł

- Żeby zweryfikować, czy użytkownik zna hasło, nie musimy go zapisywać!
- Wystarczy znaleźć mechanizm, który wymaga znajomości hasła, żeby je zweryfikować.
- Najczęściej używana metoda – użycie funkcji haszującej.

Hashowanie

- Prosty, narzucający się sposób, to zapisywanie nazwy użytkownika i hashu hasła, np. użytkownik:MD5(hasło)
- Załóżmy, że plik z takimi hasłami udało nam się wykraść. Jak możemy znaleźć hasła użytkowników?

Hashowanie

- MD5 to kiepska funkcja haszująca.
- Możemy testować wszystkie hasła na raz.
- Możemy hasła policzyć zawczasu.

2.4 Rainbow tables

Rainbow table

- Wyliczanie hashów zawczasu wymaga ich zapisania, a to zajmuje masę miejsca.
- Można sobie pomóc, zapisując tylko niektóre hashe.
- Trick podobny do rho Pollarda.

3 Przechowywanie haseł

3.1 Salting

Salting

- Rozwiązaniem większości problemów z atakami offline jest salting.
- Do każdego użytkownika dodajemy jawną wartość, którą zapisujemy w pliku z hasłami, którą dorzucamy do hasła przy liczeniu hashu: MD5(hasło + salt) = zapisana wartość?
- Jak powinien być tworzony dobry salt?

Salting

- Salt powinien być unikalny dla danego serwisu/usługi.
- Salt powinien być unikalny dla każdego użytkownika.
- Nigdy nie używamy powtórnie.

Hashowanie a GPU

- Moc obliczeniowa dostępna za przystępną cenę bardzo się zwiększyła dzięki GPU.
- Dobre karty graficzne potrafią liczyć do 1000 Ghash/s.
- Niezależnie od sprytu przy saltingu, taka moc obliczeniowa pozwala szybko policzyć np. dowolne hasła o długości do 8 znaków.

3.2 Key stretching

Key stretching

- Ze słabego klucza wytwarza dłuższy klucz, z dużej przestrzeni kluczy.
- Proces celowo powolny – bez drogi na skróty.

PBKDF2

- Password Based Key Derivation Function 2 (RFC 2898).
- Rekomendowana przez NIST.
- PBKDF2(PRF, hasło, salt, c, dkLen)
- DK $= T_1 || T_2 || \ldots || T_{\text{dkLen/hLen}}$
- $T_i = F$(hasło, salt, iteracje, i)
- F(hasło, salt, c, i) $= U_1 \oplus U_2 \oplus \ldots \oplus U_c$
- $U_1 = \text{PRF}(\text{hasło}, \text{salt}||\text{INT-msb}(i))$ $\ldots$ $U_c = \text{PRF}(\text{hasło}, U_{c-1})$
- Przykład – WPA2: klucz = PBKDF2(HMAC-SHA1, hasło, ssid, 4096, 256)

bcrypt

- Opiera się na algorytmie Blowfish – szyfruje dane w dokładnie taki sam sposób.
- Różni się w kroku key schedule – ustalaniem kluczy rund i zawartości s-boxów.

scrypt

- Algorytm, który nie tylko spowalnia sprawdzanie hasła, ale i zwiększa wymagania pamięciowe.
- Zabezpiecza przed przetwarzaniem równoległym.
- Zbyt nowy, żeby bezkrytycznie polecać (obecnie draft IETF).

3.3 Rozwiązania radykalne

Dowody z wiedzą zerową

- Umożliwiają przekonanie weryfikującego, że znamy sekret, bez zdradzania żadnej informacji o sekrecie.
- Dużo matematyki – użytkownik musi polegać na urządzeniu.

Zamiana hasła na coś co mam

- Może w ogóle lepiej dać użytkownikom token?
- Niektóre tokeny są w pełni kompatybilne z wprowadzaniem hasła z klawiatury.

HSM

- Hasła można bezpiecznie przechowywać w HSM.
- HSM tylko weryfikuje, czy hasło jest poprawne, ale nigdy go nie zdradzi.

4 Użyszkodnicy a hasła

4.1 Phishing

Phishing

- Użytkownicy wierzą we wszystko.
- Edukacja jest jedynym wyjściem.
- Można próbować np. personalizować okno loginu dla każdego użytkownika.