Bezpieczeństwo systemów i sieci komputerowych

dr inż. Mirosław Mazurek

Zakład Systemów Złożonych
Bud. F, pok. 305, tel. 17 865 11 04

Czym się różni hashowanie od szyfrowania?

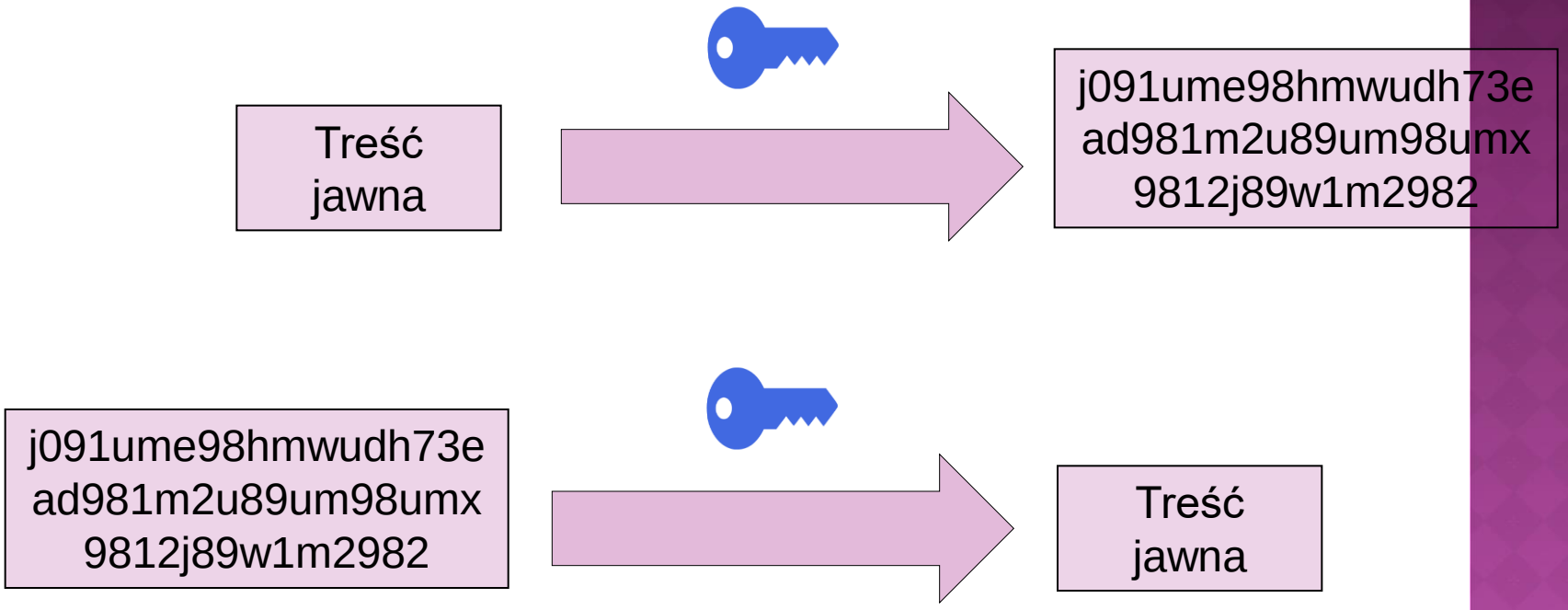
Funkcja skrótu, **funkcja mieszająca** lub **funkcja haszująca** – funkcja przyporządkowująca dowolnie dużej liczbie krótką, zawsze posiadającą stały rozmiar, niespecyficzną, quasi-losową wartość, tzw. *skrót nieodwracalny*.

W informatyce funkcje skrótu pozwalają na ustalenie krótkich i łatwych do weryfikacji sygnatur dla dowolnie dużych zbiorów danych. Sygnatury mogą chronić przed przypadkowymi lub celowo wprowadzonymi modyfikacjami danych (sumy kontrolne), a także mają zastosowania przy optymalizacji dostępu do struktur danych w programach komputerowych (tablice mieszające).

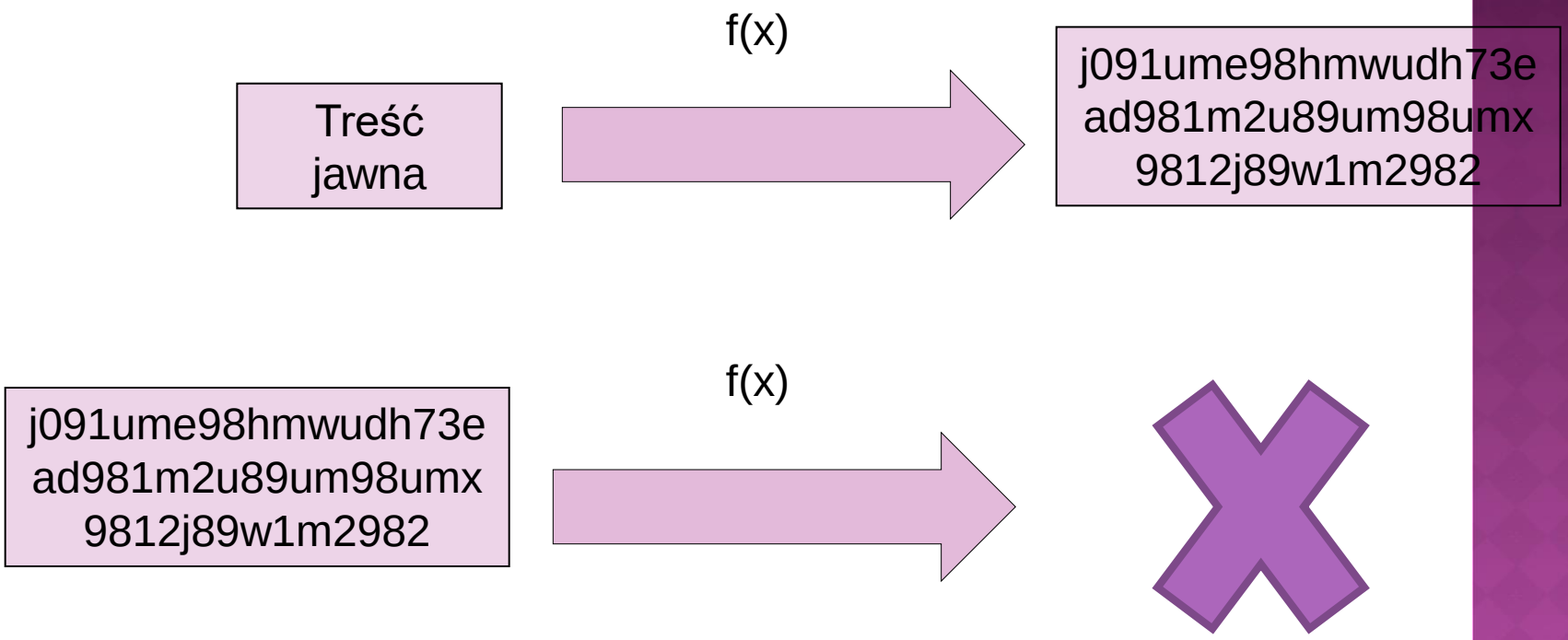
Sól (ang. *salt*) lub **ciąg zaburzający** – dane losowe dodawane do hasła podczas obliczania funkcji skrótu przechowywanej w systemach informatycznych. Celem soli jest ochrona systemowej bazy haseł przed atakami słownikowymi. Jako że sól jest przechowywana jawnie, nie ma ona znaczenia względem ataków brute-force. Podobne znaczenie i zastosowanie ma wektor inicjujący.

Czym się różni hashowanie od szyfrowania?

Szyfrowanie / odszyfrowywanie:



Hashowanie / „odhaszowanie”:



MD5

Nazwa użytkownika	md5(hasło)
kowalski@gmail.com	e10adc3949ba59abbe56e057f20f883e
iksinski@yahoo.com	5f4dcc3b5aa765d61d8327deb882cf99
mozart@opera.org.pl	e10adc3949ba59abbe56e057f20f883e
…	

123456

password

123456

Sól

Nazwa użytkownika	sól	md5(sól.hasło)
kowalski@gmail.com	Mqkr4C49OK	82a8555ddb6cc6398750ee6dcb9e7260
iksinski@yahoo.com	UGSDQcqD1R	2f438208b01cf1fdcfe8701192b21eb1
mozart@opera.org.pl	JmaGHTn5hb	73454a8510402bfba4946ea9f8a73999
…		

123456

password

123456

- OWASP TOP 10

OWASP 10 to dokument poświęcony bezpieczeństwu aplikacji internetowych, będących ogromnym zagrożeniem dla aplikacji sieciowych. Członkowie projektu obejmują wielu ekspertów ds. bezpieczeństwa z całego świata, którzy podzielili się swoją wiedzą na ten temat.

Przyjęcie OWASP Top 10 jest prawdopodobnie najbardziej skutecznym pierwszym krokiem w kierunku zmiany rozwoju kultury w organizacji jako organizacji, która produkuje bezpieczny kod.

• OWASP TOP 10

OWASP Top 10 Application Security Risks - 2017

A1-Injection

Injection flaws, such as SQL, OS, XXE, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2-Broken Authentication and Session Management

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities (temporarily or permanently).

A3-Cross-Site Scripting (XSS)

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

A4-Broken Access Control

Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

A5-Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

A6-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A7-Insufficient Attack Protection

The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts. Application owners also need to be able to deploy patches quickly to protect against attacks.

A8-Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. Such an attack allows the attacker to force a victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A9-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

A10-Underprotected APIs

Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps, that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities.

• OWASP TOP 10: A1 - Injection

- niepoprawna walidacja parametrów
- bezpośrednie odwołanie do bazy danych (SQL injection)
- XML injection

```
GET /login.php HTTP/1.1
login=admin&password=1' or 1='1
```

```
select * from users where login='admin' and pass='1' or 1='1'
```

https://www.owasp.org/index.php/Top_10_2017-A1-Injection

OWASP TOP 10: A2 - Broken Authentication and Session Management

- zarządzanie sesjami i użytkownikami np.
  - hashowanie haseł
  - timeout sesji użytkownika
  - losowość tokenów sesyjnych
  - enumeracja użytkowników
  - omijanie uwierzytelnienia
  - ataki bruteforce

https://www.owasp.org/index.php/Top_10_2017-A2-Broken_Authentication_and_Session_Management

• OWASP TOP 10: A2 - Broken Authentication and Session Management

Scenariusz #1: Aplikacja rezerwacji linii lotniczych obsługuje ponowne zapisywanie adresów URL, umieszczając identyfikatory sesji w adresie URL:

http://example.com/sale/saleitems;
sessionid=268544541&dest=Hawaii

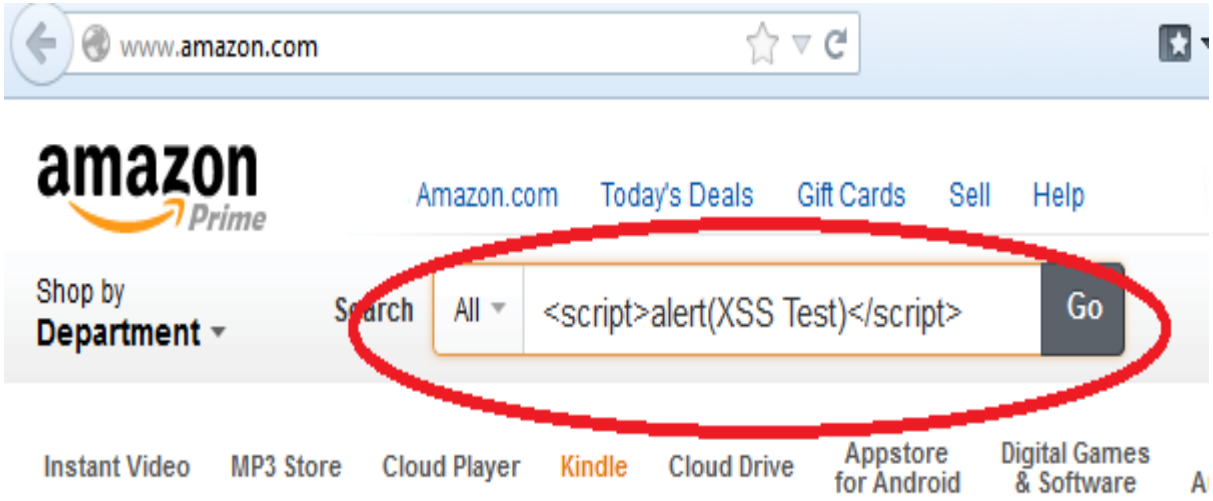
Uwierzytelniony użytkownik witryny chce poinformować znajomych o sprzedaży. Wysyła e-mail z linkiem nie wiedząc, że rozdaje także swój identyfikator sesji. Kiedy znajomi użyją linku, korzystają z sesji użytkownika i karty kredytowej.

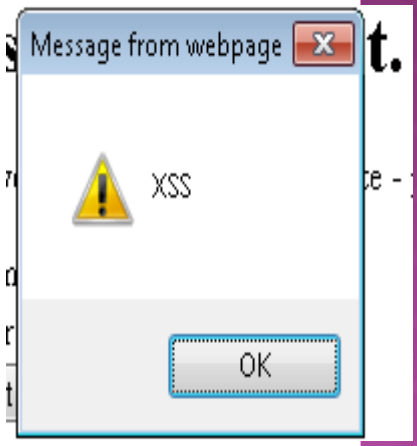
Scenariusz #2: Limity czasu aplikacji są nieprawidłowe. Użytkownik uzyskuje dostęp do witryny przez komputer publiczny. Zamiast wybierać "logout" użytkownik po prostu zamyka kartę przeglądarki i odchodzi. Osoba atakująca korzysta z tej samej przeglądarki później, a ta wyszukiwarka jest nadal uwierzytelniona.

Scenariusz #3: Wewnętrzny hacker uzyskuje dostęp do bazy danych haseł systemu. Hasła użytkownika nie są poprawnie mieszane i solone, narażając hasła każdego użytkownika.

- OWASP TOP 10: A3 - Cross-Site Scripting (XSS)

- wykonanie zewnętrznego kodu w ramach przeglądarki

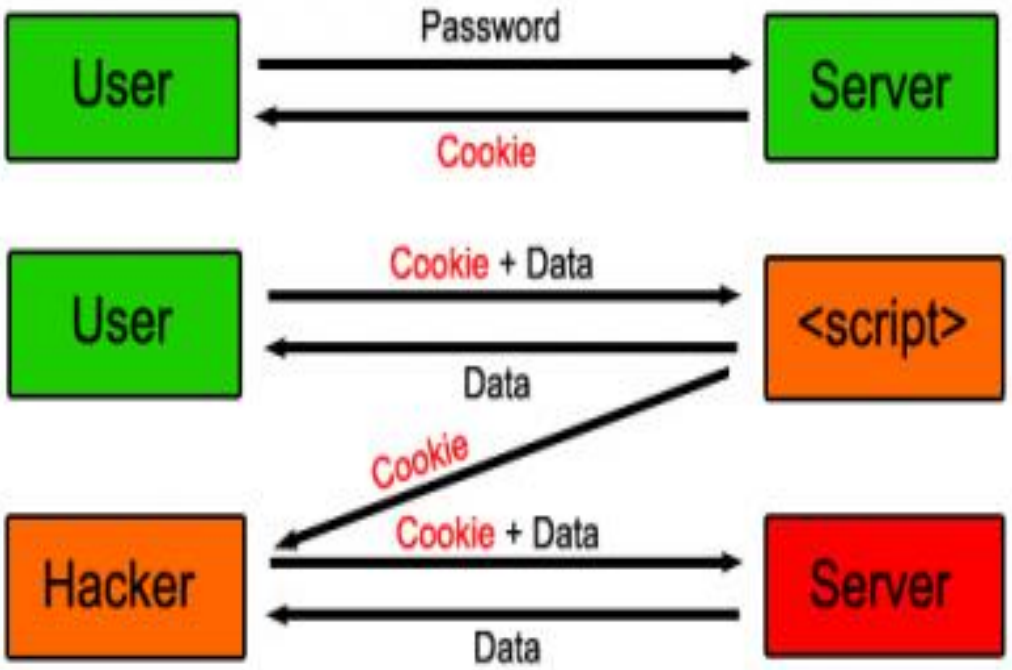




https://www.owasp.org/index.php/Top_10_2017-A3-Cross-Site_Scripting_(XSS)

- OWASP TOP 10: A3 - Cross-Site Scripting (XSS)

- wykonanie zewnętrznego kodu w ramach przeglądarki



OWASP TOP 10: A4 - **Broken Access Control**

Scenariusz #1: Aplikacja używa niezweryfikowanych danych w wywołaniu SQL, które uzyskuje dostęp do informacji o koncie:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Osoba atakująca po prostu modyfikuje parametr 'acct' w przeglądarce w celu wysłania dowolnego numeru konta. Jeśli nie zostanie poprawnie sprawdzony, osoba atakująca może uzyskać dostęp do dowolnego konta użytkownika.

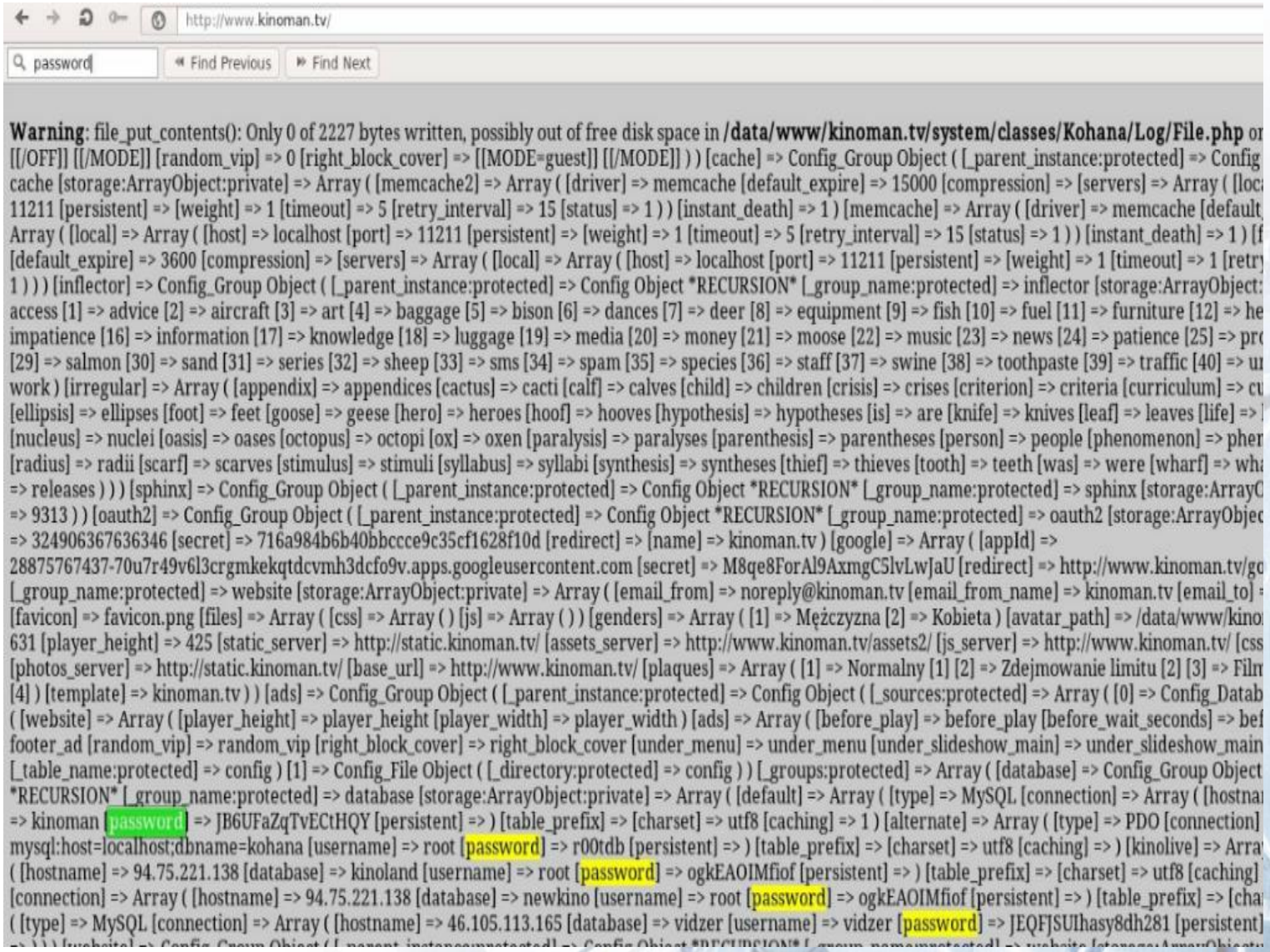
http://example.com/app/accountInfo?acct=notmyacct

Scenariusz #2: Osoba atakująca po prostu wymusza przeglądanie adresów URL. Prawa administratora są również wymagane w celu uzyskania dostępu do strony administratora.

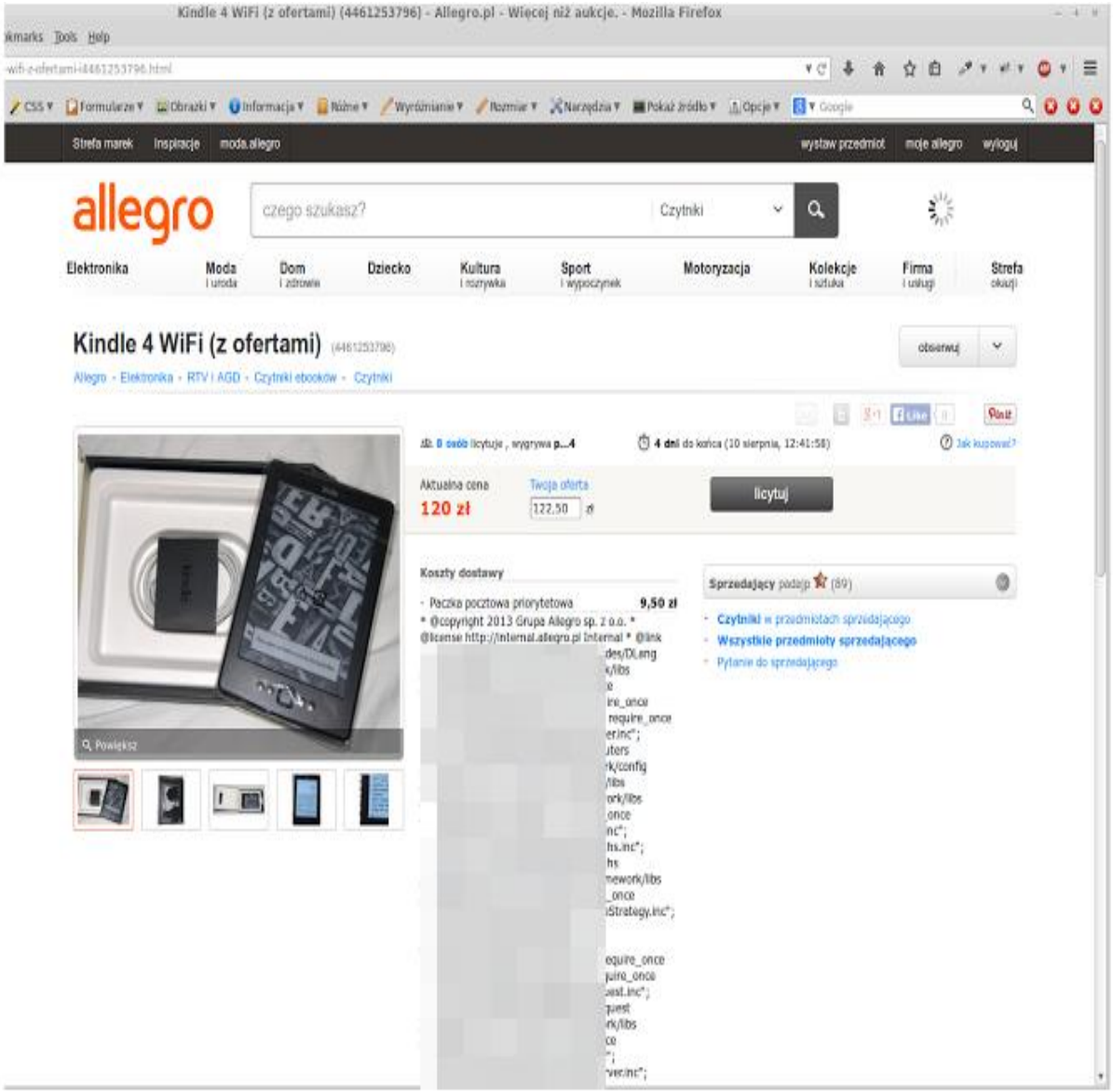
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo

Jeśli nieupoważniony użytkownik ma dostęp do jednej strony, to jest zagrożenie. Jeśli nie-admin może uzyskać dostęp do strony administratora, jest to również błąd.

- OWASP TOP 10: A5 - Security Misconfiguration



- OWASP TOP 10: A5 - Security Misconfiguration



• OWASP TOP 10: A5 - Security Misconfiguration

Scenariusz #1: Konsola administracyjna serwera aplikacji jest automatycznie instalowana i nie usuwana. Konta domyślne nie są zmieniane. Haker wykrywa standardowe strony administracyjne znajdujące się na serwerze, loguje się z domyślnymi hasłami i przejmuje.

Scenariusz #2: Lista katalogów nie jest wyłączona na serwerze WWW. Osoba atakująca odkrywa, że ma listę katalogów, aby znaleźć dowolny plik. Osoba atakująca wyszukuje i pobiera wszystkie skompilowane klasy Javy, które dekompilują i wykonują inżynierię odwrotną, aby uzyskać cały własny kod niestandardowy. Napastnik dostrzega w aplikacji poważne braki kontroli dostępu.

Scenariusz #3: Konfiguracja serwera aplikacji umożliwia użytkownikom zwracanie stosu śladów (stack traces), potencjalnie narażając się na błędy, takie jak wersje frameworków, które są znane jako podatne na zagrożenia.

Scenariusz#4: Serwer aplikacji zawiera aplikacje przykładowe, które nie są usuwane z serwera produkcyjnego. Te przykładowe aplikacje mają dobrze znane luki w zabezpieczeniach, których napastnicy mogą wykorzystać do złamania serwera.

• OWASP TOP 10: A6 - Sensitive Data Exposure

- hashowanie haseł ☺
- Brak SSL
- Podatny SSL
- Przekazywanie wrażliwych danych

• OWASP TOP 10: A6 - Sensitive Data Exposure

Scenariusz #1: Aplikacja szyfruje numery kart kredytowych w bazie danych przy użyciu automatycznego szyfrowania bazy danych. Jednak dane te są automatycznie odszyfrowywane podczas pobierania, co umożliwia luka SQL injection w celu uzyskania numerów kart kredytowych w jawnym tekście. Aby zabezpieczyć się nie przechowywać numerów kart kredytowych, używać tokenów lub stosować szyfrowanie.

Scenariusz #2: Witryna po prostu nie używa TLS dla wszystkich uwierzytelnionych stron. Osoba atakująca po prostu monitoruje ruch sieciowy (np. Otwartą sieć bezprzewodową) i przechwytuje plik cookie sesji użytkownika. Następnie osoba atakująca ponownie odtwarza ten plik cookie i porywa sesję użytkownika, uzyskując dostęp do prywatnych danych użytkownika.

Scenariusz #3: Baza haseł korzysta z nieposolonych haseł do przechowywania haseł każdego użytkownika. Luka w przesyłaniu plików umożliwia napastnikowi pobranie bazy danych haseł. Wszystkie nieposolone hasze mogą być narażone na atak z wykorzystaniem tablic tęczowych.

OWASP TOP 10: A7 - **Insufficient Attack Protection**

Scenariusz #1: Osoba atakująca używa zautomatyzowanego narzędzia, takiego jak OWASP ZAP lub SQLMap w celu wykrycia luk w zabezpieczeniach i ich wykorzystania. Detekcja ataków powinna rozpoznawać, że aplikacja jest skierowana na nietypowe żądania. Automatyczne skanowanie powinno być łatwe do odróżnienia od normalnego ruchu.

Scenariusz #2: Wykwalifikowany hacker starannie sprawdza potencjalne luki w zabezpieczeniach, w końcu odkrywając niejasną wadę systemu. Trudniejsze jest wykrycie tego ataku, ale wciąż wymaga żądań normalnego użytkownika, takiego jak dane niedozwolone przez interfejs użytkownika. Śledzenie tego napastnika może wymagać zdefiniowania przypadku, który wykazuje złośliwy zamiar.

Scenariusz #3: Atak rozpoczyna wykorzystywanie luki w zabezpieczeniach aplikacji, której obecna ochrona przed atakami nie blokuje. Jak szybko można wdrożyć prawdziwą lub wirtualną łatę w celu zablokowania dalszej eksploatacji tej luki?

• OWASP TOP 10: A8 - Cross-Site Request Forgery (CSRF)

- Wykonanie akcji na innej sesji przeglądarki przez inną stronę
- Wymaga interakcji

Aplikacja pozwala użytkownikowi na złożenie żądania zmiany stanu, który nie zawiera niczego tajnego. Na przykład:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

Więc osoba atakująca konstruuje żądanie, które przenosi pieniądze z konta ofiary na konto osoby atakującej, a następnie osadza ten atak w żądaniu obrazu lub iframe przechowywanym w różnych witrynach pod kontrolą osoby atakującej:

****

Jeśli ofiara odwiedzi dowolną witrynę osoby atakującej, która została już uwierzytelniona na example.com, te fałszywe żądania automatycznie zawierają informacje o sesji użytkownika, autoryzując żądanie osoby atakującej.

• OWASP TOP 10: A9 - **Using Components with Known Vulnerabilities**

Komponenty prawie zawsze działają z pełnym prawem dostępui, więc błędy w dowolnym składniku mogą powodować poważne konsekwencje. Takie wady mogą być przypadkowe (np. Błąd kodujący) lub celowe (np. Backdoor w składniku). Oto niektóre przykładowe luki w zabezpieczeniach:

Apache CXF Authentication Bypass – nie dostarczając identyfikatora tożsamości, napastnicy mogą powoływać się na dowolną usługę sieciową z pełną zgodą. (Apache CXF jest ramą usług, a nie mylić z Apache Application Server.)

Struts 2 Remote Code Execution – Wysyłanie ataku w nagłówku Content-Type powoduje, że zawartość tego nagłówka zostanie oceniona jako wyrażenie OGNL, co umożliwia wykonanie dowolnego kodu na serwerze. Aplikacje korzystające z wrażliwej wersji dowolnego składnika są podatne na ataki, ponieważ oba składniki są bezpośrednio dostępne dla użytkowników aplikacji. Inne słabsze biblioteki, używane głębiej w aplikacji, mogą być trudniejsze do wykorzystania.

OWASP TOP 10: A10 - **Underprotected APIs**

Scenariusz #1: Wyobraźmy sobie mobilną aplikację bankową, która łączy się z interfejsem XML API w banku w celu uzyskania informacji o koncie i wykonywania transakcji. Osoba atakująca używa inżynierii odwrotnej i odkrywa, że numer konta użytkownika jest przekazywany jako część żądania uwierzytelnienia do serwera wraz z nazwą użytkownika i hasłem. Osoba atakująca wysyła legalne poświadczenia, ale numer konta innego użytkownika, uzyskując pełny dostęp do konta innego użytkownika.

Scenariusz #2 Wyobraźmy sobie publiczne API oferowane przez Internet do automatycznego wysyłania wiadomości tekstowych. Interfejs API akceptuje wiadomości JSON, które zawierają pole "transactionid". Interfejs API analizuje tę wartość "transactionid" jako ciąg i łączy go w kwerendę SQL bez escapingu lub parametryzacji. Jak widać, API jest tak samo podatne na wstrzyknięcie SQL jak każdy inny rodzaj aplikacji.
W każdym z tych przypadków dostawca może nie dostarczyć interfejsu WWW do korzystania z tych usług, co utrudnia testowanie zabezpieczeń.

• Missing Function Level Access Control

- Wykonanie funkcji z prawami innego użytkownika
- Dostęp do panelu administratora po zalogowaniu
  http://domena.pl/admin
- Ominięcie kroku:

http://sklep.com/order.php?step=1

http://sklep.com/order.php?step=2

http://sklep.com/order.php?step=4

Unvalidated Redirects and Forwards

- Przekierowania do niezaufanych stron

```
http://www.example.com/redirect?url=evil.com
```

- OWASP TOP 10: Insecure Direct Object References

http://sklep.com/portal/pobierz_fakture.php?id=268544541

http://sklep.com/portal/pobierz_fakture.php?id=2685445442

https://www.olx.pl/mojolx/geteinvoice/?id=xxxxxxx

Unvalidated Redirects and Forwards

- Phishing:

```
http://cgi4.ebay.com/ws/eBayISAPI.dll?FMfcISAPICommand%3DRedirectToDomain%26DomainUrl%3Dhttp%3A%2F%2Fexamp

le%2FUpdateCenter%2FLogin%2F
```