Web Security

SQL-Injection:

Erweitern oder Modifizieren einer SQL-Query einer Anwendung über Eingabedaten (z.B. Textfelder, URL). Nutzt mangelnde Überprüfung oder Maskierung von Benutzereingaben aus.

Beispiel 1:

SQL-Query: SELECT * FROM users WHERE username='$username' and password='$password'

Benutzereingabe:

Username: Max
Password: ' OR '1'='1

Ergebnis: SELECT * FROM users WHERE username='Max' and password='' OR '1'='1

Erläuterung: password='' wird zwar in der Regel zu false ausgewertet werden, aber '1'='1' ist immer true. Wenn es einen Benutzer "Max" gibt, kann man sich in seinem Namen einloggen ohne das Passwort zu kennen.

Beispiel 2:

SQL-Query: SELECT * FROM news WHERE news_id='$news_id'

Benutzereingabe:

news_id: 42'; DROP TABLE news; #

Ergebnis: SELECT * FROM news WHERE news_id='42'; DROP TABLE news; #'

Erläuterung: Das # wird genutzt, damit das letzte ' keinen Syntaxerror erzeugt.

Beispiel 3:

SQL-Query: SELECT ProductName, Price FROM Products WHERE ProductName LIKE '$SearchBox'

Benutzereingabe:

anything' UNION SELECT UserName, Password FROM Users --

Ergebnis: SELECT ProductName, Price FROM Products WHERE ProductName LIKE 'anything' UNION SELECT UserName, Password FROM Users --'

Erläuterung: Durch das Vereinigen der 2 Queries werden auch die Benutzername-Passwort Kombinationen aller Nutzer zurückgeliefert.

Gegenmaßnahmen:

1. Benutzung von gebundenen und typisierten Parametern bei parametrisierten Queries (Prepared Statements)

2. Vorsichtige Benutzung von "stored procedures"

3. Nutzung von Accounts mit minimalen Privilegien

Cross-Site-Scripting (XSS):

Ein Angreifer injiziert schädlichen Javascript Code in eine Webseite und wird vom Browser des Opfers ausgeführt.

Persistent vs. Non-Persistent XSS:

Persistent: Der Angreifer injiziert das Script in die Webanwendung, bevor das Opfer diese aufruft. Der schädliche Code wird in der Webanwendung gespeichert.

Non-Persistent: Der Angreifer sendet dem Opfer eine URL mit dem Script. Der schädliche Code wird nicht in der Webanwendung gespeichert.

Gegenmaßnahmen:

Aufbereitung der Benutzereingaben:
htmlspecialchars(string) ersetzt alle Sonderzeichen durch ihre HTML-Codes

Nachteile von Blacklisting:

Ein Zeichen kann sehr viele verschiedene Repräsentationen haben.

Schlechtes Session-Management

Gegenmaßnahmen:

1. Einloggdaten sollten immer geschützt (verschlüsselt) abgespeichert werden

2. Einloggdaten sollten nicht leicht zu erraten sein

3. Session-ID's sollten nicht in der URL übergeben werden

4. Sessions sollten nach einer gewissen Zeit ablaufen (Timeout)

5. Sessions sollten rotieren

Sichere Password-Recovery Mechanismen

1. Abfragen von mehreren harten Daten

2. Überprüfen der Sicherheitsfragen (falls falsch => generische Fehlermeldung)

3. Versenden eines Tokens über einen Seitenkanal (E-Mail, SMS)

4. Erlaube dem Benutzer ein neues Passwort festzulegen

Cross-Site-Request-Forgery (CSRF)

Bringt den Browser des Benutzers dazu eine unautorisierte Anfrage an die Webseite, bei der er gerade eingeloggt ist, zu senden.

Vorgehen:

1. Das Opfer loggt sich bei einer Seite ein

2. Der Angreifer sendet dem Opfer eine URL oder das Opfer besucht die Webseite des Angreifers

3. Das Opfer sendet einen Request zu dem Server, bei dem es sich vorher eingeloggt hat

Gegenmaßnahmen:

1. PIN- oder TAN-Eingabe

2. Nutzung zufälliger Einmal-Tokens

3. Senden eines Tokens sowohl über "form" als auch über einen Cookie

Unsicheres kryptographisches Speichern

- Angreifer brechen i.d.R. keine kryptographischen Funktionen, sondern versuchen z.B. Schlüssel oder Klartextkopien zu finden
- Viele sensitive Daten werden nicht verschlüsselt
- unsichere Schlüsselgenerierung und Speicherung, Schlüssel rotieren nicht, schwacher Algorithmus
- schwache oder "unsalted" Hashes werden verwendet um das Passwort zu beschützen

Gegenmaßnahmen:

1. Gefahren, gegen die man die Daten beschützen möchte

2. Backups werden verschlüssel und Schlüssel werden getrennt gespeichert

3. starke Algorithmen und Schlüssel

4. Passwörter Hashen mit starkem Algorithmus und "salt"

5. Schlüssel und Passwörter werden vor unautorisiertem Zugriff geschützt

Redirects und Forwards

Gegenmaßnahmen:

1. Redirects und Forwards vermeiden

2. keine Parameter zur Bestimmung der Zieladresse verwenden

3. Falls Zielparameter nicht umgangen werden können, muss überprüft werden, dass der Wert für den Benutzer gültig und autorisiert ist