Übung 1
Web Application Security
IT-Sicherheit WS 22/23

Moritz Buhl

24.10.2022 – **30.10.2022 23:59**

Organisation: Übungen

- Fragen: Zulip `https://zulip.in.tum.de/`
- Übung
  - Dienstag 17:00 – 19:00, 00.08.059 **oder**
  - Freitag 10:00 – 12:00, 00.13.036
  - Bei Entfall bitte eine andere Gruppe besuchen!
- Hausaufgaben
  - 14 Übungsblätter mit Hausaufgaben + 2 Bonusblätter
  - In Teams à 2 Studenten / Plagiatsversuche werden geahndet!
  - Programmieraufgaben im „Capture the Flag“ Stil
  - Ausgabe über `https://scoreboard.sec.in.tum.de`
  - Vorstellung von **mind. 2 Hausaufgaben** in Übung für Bonus!

Aufgabe 1: System Setup

1. Public Key auf `https://scoreboard.sec.in.tum.de` hochladen
2. Mit dem im Scoreboard angezeigten Befehl auf dem Sandkasten Rechner einloggen
3. Team auf Scoreboard registrieren

Aufgabe 2: Begrifflichkeiten

a) Grenzen Sie kurz die Begriffe *Safety* und *Security* voneinander ab. Worin unterscheiden sie sich?

- **Safety: Funktions- und Betriebssicherheit**
  - Erkennen und Abwehr von Störungen, die die korrekte Funktionalität (Betriebssicherheit) beinträchtigen
  - Störungen kommen **von Innen**
- **Security: Daten- und Informationssicherheit**
  - Verwundbarkeit von zu schützenden Werten systematisch reduzieren
  - Bewahren eines Systems vor Beeinträchtigung und Missbrauch durch Angriffe
- Es gibt Wechselwirkungen zwischen Security und Safety
  - Aufwändige Zugriffskontrollen (für Security) können im Notfall die Safety beeinträchtigen
  - Redundanz (erhöht Safety) bringt zusätzliche Angriffsmöglichkeiten

Aufgabe 2: Begrifflichkeiten

b) Beschreiben Sie die Begriffe **Schwachstelle**, **Bedrohung** und **Angriffsvektor**

- **Schwachstelle**: ermöglicht es, dass die Sicherheitskontrollen des Systems umgangen oder getäuscht werden können
  - Beispiel: das System erlaubt schwache Passwörter
- **Bedrohung**: Ein Umstand oder Ereignis mit dem **Potenzial**, ein System durch unbefugten Zugriff, Zerstörung, Offenlegung, etc. zu beeinträchtigen.
  - Beispiel: DDoS-Attack
- **Angriffsvektor**: Ein **konkreter** Angriffsweg, der eine oder mehrere Schwachstellen ausnutzt, um die Sicherheit eines Systems zu gefährden
  - Beispiel: Eve snifft Alice' Passwort (Schwachstelle: unverschlüsselte Übertragung) und gibt sich dann als Alice aus

Aufgabe 3: Web Basics

Betrachten Sie die unten stehenden Webseiten genauer. Schauen Sie sich dazu den HTML-Code der Webseite an. Diesen können Sie über die Web Entwickler Konsole in ihrem Browser einsehen (F12 drücken).

a) `http://itsec.sec.in.tum.de:7000/`

b) `http://itsec.sec.in.tum.de:7001/`

c) `http://itsec.sec.in.tum.de:7002/`

Können Sie auf den Webseiten eine Flagge finden?

Aufgabe 4: SQL Injections

- **SQL** (Structured Query Language) ist eine Sprache, die uns erlaubt, CRUD (und weitere Operationen) auf relationalen Datenbanken auszuführen
- Relationale Datenbanken speichern Daten in Tabellen
- Tabellen können auf andere Tabellen verweisen (Relations)
- Das Schema enthält alle Tabellen und deren Attribute
- Um die Tabelleninhalte auszulesen, verwendet man den SELECT-Befehl
- Es gibt viele weitere Befehle - siehe Cheatsheet
- Es gibt verschiedene Extensions für SQL, die jeweils die Features einer proprietären DB abbilden: MySQL, PostgreSQL, SQLite
- Ausführlichere Erklärung: `https://wiki.selfhtml.org/wiki/Datenbank/SQL-Grundlagen`

Aufgabe 4: SQL Injections

a) Erläutern Sie möglichst allgemein, was eine Injection Vulnerability ist, und nennen Sie Beispiele für Stellen, an welchen eine solche Schwachstelle vorkommen kann!

- Nicht validierte Daten werden von einem Interpreter als Bestandteil einer Anfrage verarbeitet, um z.B. **Kommandos auszuführen** oder die **Semantik** zu verändern.
- Beispiele: SQL Injection in SQL-Abfrage, XSS (unsicherer Webserver erlaubt Einfügen von Schadcode in Website)

Aufgabe 4: SQL Injections

b) Schauen Sie sich das folgende Beispiel zu SQL-Injections an. Gehen Sie davon aus, dass es aus einem fiktiven Nutzerverwaltungsprogramm stammt. Wie geht der Angreifer vor und was ist im gegebenen Beispiel sein Ziel?

- Der Server sieht die Query als `SELECT * FROM users WHERE vorname=" OR 1=1 - - AND id > 1000`
- `- - AND id > 1000` ist ein Kommentar, ist also nicht Teil des Filters
- Damit werden alle Tabelleneinträge ausgegeben, da `vorname=" OR 1=1` true ist, unabhängig vom Eintrag

Aufgabe 4: SQL Injections

c) Nehmen Sie an Sie wüssten nicht, welche Tabellen in der Datenbank vorhanden sind. Wie würden Sie dann vorgehen? Können Sie diese Informationen in der gegebenen Aufgabe auch über eine SQL-Injection erhalten?

- Hint: Jede SQL-DB enthält eigene Managementtabellen, die das Schema der Datenbank beschreiben
- Man kann die Infos mittels einer SQL-Union aus der Managementtabelle auslesen

Aufgabe 4: SQL Injections

d) In einigen Fällen werden die Ergebnisse einer verwundbaren Datenbankabfrage überhaupt nicht ausgegeben, sondern man erhält von der verwundbaren Webseite nur eine richtig oder falsch Meldung. Entwickeln Sie eine Methode, um trotzdem Informationen aus einer Webseite zu extrahieren.

- Einzelne Bytes eines Strings können mit SUBSTR extrahiert werden.
- SUBSTR kann ein weiteres SQL Statement als String bekommen.
- Dieses Konstrukt kann durch Injection an eine vorherige Kondition gekettet werden.

`https://www.sqlite.org/lang_corefunc.html#substr`