bWAPP vs. Juice Shop

Unsichere Webservices

Seminararbeit

Ausgewählte Kapitel der IT-Security

Vorgelegt von:
Melanie Kaimer

Personenkennzeichen
1710475082

Abgabe am:
08.01.2020

Abkürzungsverzeichnis

HTTP	HyperTextTransfer Protocol
OWASP	Open Web Application
PHP	Hypertext Preprocessor
SQL	Structured Query Language
WAMP	Windows mit Apache, MySQL(MariaDB) und PHP;
XAMPP	PHP Entwicklungsumgebung
XSS	Cross Site Scripting

Schlüsselbegriffe

bWAPP
Juice Shop
OWASP
Penetrating Testing
SQL-Injection-Attack

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Anzahl der Benutzer*innen von Online-Diensten hat in den letzten Jahren ein exponentielles Wachstum erfahren. Webanwendungen stellen eine fundamentale Säule in der heutigen Welt dar. Somit wachsen auch die Gefahren von Angriffen im Netz und machen sogenannte Penetrationstests nötig. Ein Penetrationstest umfasst eine Reihe von Aktivitäten, mit welchen Sicherheitslücken erfasst und identifiziert werden.

In dieser Seminararbeit werden zwei beabsichtigt unsichere Webservices miteinander in einen Vergleich gebracht. Unterschiede in der Installation, sowie der Architektur werden in theoretischer Ausführung erläutert. Der Schwerpunkt des praktischen Teiles umfasst die Austestung der weltweit am häufigsten verwendeten SQL-Injection-Attacke. Angreifer*innen nutzen dabei gezielt Schwachstellen aus, um Zugänglichkeit zu vertrauten Informationen zu erhalten. Bei der SQL-Injection-Attacke wird eine Variation an Möglichkeiten beschrieben.[6][5]1.1

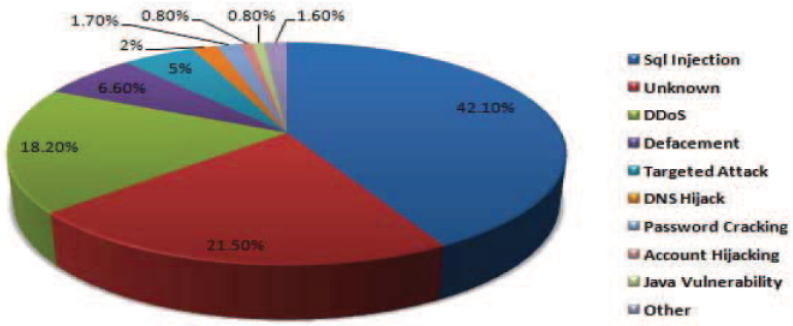


Abbildung 1.1: Statistik der Häufigkeit verschiedener Attacken, verwendet von [6]

1.1 Motivation

Die Bedrohungsrate für Webanwendungen nimmt täglich zu, da der Gebrauch webbasierter Anwendungen steigt und somit immer mehr an Bedeutung gewinnt. Die meisten Softwarefehler sind Resultate unangemessener Eingabevalidierungen. Dies führt

immer häufiger zu einem Angriff auf vertrauliche Informationen. Die Sicherheit von Webanwendungen ist ein wesentlicher Bestandteil webbasierter Anwendungen. Sicherheitslücken wie etwa Cross Site Scripting (XSS), Structured Query Language (SQL) und unangemessene Passwörter stellen ein großes Sicherheitsrisiko für Benutzer*innen dar. XSS-Angriffe und SQL-Injection-Attacken sind bei Hacker*innen sehr beliebt, da sie über einfache Skripte ausgeführt werden.

Das Erkennen dieser Schwachstellen ist oftmals keine einfache Angelegenheit. Viele Softwarefehler in Webanwendungen sind Ergebnisse ungültiger Eingaben. Obwohl der überwiegende Teil von Sicherheitslücken im Web direkt und aus strategischer Distanz passiert, sind zahlreiche Webdesigner überraschenderweise nicht sicherheitsbewusst. Das Testen von Schwachstellen für Webanwendungen ist eine Möglichkeit, um mit derartigen Problemen umzugehen. Tools wie *bWAPP* und *OWASP Juice Shop* bieten Anwender*innen neben Spaß auch die Möglichkeit einige dieser Schwachstellen aufzudecken.[4]

Kapitel 2

Web Security Test-Application

Sicherheit ist ein wichtiger Bestandteil und eines der Hauptprobleme von Informationssystemen. Mithilfe von Penetrationstests werden gezielt Sicherheitslücken identifiziert, um Organisationen und Benutzer*innen mehr Sicherheit vor Datenverlust bieten zu können. Hierfür werden sogenannte unsichere Webservices wie beispielsweise *OWASP Juice Shop* und *bWAPP* genannt. Im folgenden Kapitel werden diese beiden unsicheren Webservices miteinander verglichen sowie die unterschiedlichen Installationswege beschrieben.

2.1 Penetrating Testing

Unter dem Begriff Penetrating Testing -kurz als Pentest bezeichnet- versteht man einen gezielten, erlaubten Versuch in ein IT-System einzudringen. Das Hauptziel besteht darin, Sicherheitslücken aufzudecken und zu beseitigen, um die IT-Sicherheit zu verbessern. Schwachstellen sollen aufgezeigt werden, noch bevor diese entstehen. Hierbei kommen Methoden wie beispielsweise SQL-Injection, XML External Entities (XEE) und Cross-Site Scripting (XSS) zum Einsatz, wie diese auch von Hacker*innen in Verwendung sind.[8]

Beim Pentesting wird zwischen Infrastruktur- und Webanwendungstests unterschieden. Zu den Penetrationstests zählen Black-Box, White-Box und Grey-Box-Penetrationtests [1]. Die angewandten Testmethoden finden extern sowie intern statt. Beim externen Penetrating Testing sind Unternehmen an der Sicherheit ihrer Systeme durch äußere Bedrohungen interessiert. Internes Pentesting handelt von einem größeren Unternehmen mit einer Anzahl von über 50 Mitarbeitern. Hier können Schwerpunkte wie zum Beispiel das Herausfinden eines gehashten Kennwortes gesetzt werden.[7]

Für Webanwendungen bietet das *Open Web Application Security Project* (OWASP) Materialien für Pentests an. OWASP ist unter anderem auch an der Veröffentlichung des Webservice-Tool-Projects namens *Juice Shop* (Abschnitt 2.4) beteiligt. Eine OWASP-Top-10-Liste zeigt die aktuellste Liste der derzeit riskantesten Sicherheitslücken von Webanwendungen an. OWASP bietet neben wertvollen Anleitungen für sichere Webentwicklung auch Abhilfen für Sicherheitslücken.[8]

[1] https://www.itexperst.at/penetrationstest-definition-abgrenzung-ueberblick

2.2 SQL-Injection-Vulnerability

In den Anfängen des Internets war das Erstellen von Webseiten viel unkomplizierter. Es gab weder Skriptsprachen wie JavaScript noch Style-Sheet Sprachen für elektronische Dokumente wie *Cascading Style Sheet*s (CSS) und ebenso sehr wenige Images. Mit zunehmender Popularität des *World Wide Web* gab es einen zunehmenden Bedarf an fortschrittlicher Technologie und dynamischen Webseiten. Serverseitige Skriptsprachen wie Active Server Pages (ASP), Java Servlet Pages (JSP) und Hypertext Preprocessor (PHP) wurden entwickelt. Webseiten machten eine Entwicklung durch und Benutzereingaben wie Webseiteninhalte werden in Datenbanken gespeichert. Jede gängige serverseitige Skriptsprache unterstützt SQL-Datenbanken. Hier haben Hacker*innen Angriffspunkte entdeckt und solange relationale Datenbanken in Webanwendungen verwendet werden, besteht auch die Gefahr von SQL-Injection-Attacken.

Die SQL-Injection-Vulnerability ist eines der gefährlichsten Probleme für die Vertraulichkeit und Integrität in Webanwendungen und wurde somit in die OWASP-Top-10 Liste als eine der häufigsten Sicherheitsanfälligkeiten seit ihrer Einführung beschrieben. Angreifer*innen können durch eine SQL-Injection-Vulberability Malware in ein SQL-Statement einfügen. Die Funktionalität einer Webanwendung generiert eine Zeichenfolge. Das SQL-Statement wird an die Funktion übergeben, welche die Zeichenfolge an die Datenbank sendet. Die Zeichenfolge wird in weiterer Folge analysiert, ausgeführt und das Ergebnis zurückgegeben. Beispielsweise ist es oft problemlos möglich eine SQL-Syntax in eine Anweisung einzufügen, wenn die Eingabe von der Anwendung nicht geschützt wurde.

Der praktische Teil dieser Arbeit befasst sich mit der SQL-Injection-Attacke. Wie ich später im praktischen Teil zeigen werde, ist es sehr einfach möglich mittels dieser Attacke die Kennwortüberprüfung effektiv zu entfernen und einen Datensatz für vorhandene Anwender*innen zurückzugeben - in diesem Fall $'admin'$. Angreifer*innen können sich dadurch mit einem Administratorkonto anmelden, ohne ein Kennwort angeben zu müssen.

Angreifer*innen sind in der Lage eine Reihe von Aktionen auszuführen sobald eine SQL-Injection auf eine anfällige Seite angewandt wird. Durch Ausnutzen dieser Sicherheitslücke, können Inhalte der Datenbank hinzugefügt, bearbeitet, gelöscht oder gelesen werden. Ausschlaggebend sind hier die Kenntnisse der Angreifer*innen, wobei die Ausnutzung einer SQL-Injection-Vulnerability zu einer vollständigen Übernahme der Datenbank und des Webservers führen kann. Es ist sinnvoll, den Zugriff auf die eigenen Daten einzuschränken um mögliche Schäden durch Hackerangriffe zu vermeiden.[3]

2.3 bWAPP

Der Begriff bWAPP steht für *buggy Web Applications*. bWAPP gehört dem $'ITSEC\ Games\text{-}Project'$ an und beschreibt eine bewusst extrem fehlerhafte Webanwendung. Entworfen wurde bWAPP von Malik Mesellem mit dem Ziel der Gewährleistung der IT-Sicherheit. Des Weiteren hat es einen Gaming-Charakter und soll neben dem Training auch als Spaßfaktor dienen. Diese bewusst unsichere Webanwendung umfasst alle bekannten Sicherheitslücken. Sie bietet Sicherheitsexpert*innen, Entwickler*innen und

Student*innen die Möglichkeit, Probleme zu entdecken und zu verhindern. Zusätzlich bereitet bWAPP auf erfolgreiche Penetrationstests und ethische Hacking-Projekte vor. bWAPP ist einfach in der Anwendung und sehr verständlich aufgebaut. Es gibt 3 Sicherheitsstufen von Challenges: leicht, mittel und schwer. Über 100 Web-Schwachstellen sind vorhanden und somit werden alle bekannten Web-Bugs abgedeckt. Der Fokus liegt nicht auf einem bestimmten Thema. Dokumentationen zur Hilfestellung sind vorhanden sowie auf der Webseite von *'itsecgames'* ausführlich beschrieben.[1]

2.4 OWASP Juice Shop

Auf den ersten Blick wirkt der *OWASP Juice Shop* wie ein unscheinbarer Online-Shop für Fruchtsäfte. Der *Juice Shop* wurde im Jahre 2014 von Björn Kimminich entworfen und ist ein sogenannter 'Online-Saftladen' für Sicherheitstrainings. Zwei Jahre nach der Gründung wurde *Juice Shop* als OWASP-Tool-Project eingereicht und angenommen. Dieser Schritt hat den *Juice Shop* innerhalb kürzester Zeit für eine große Community von Nutzer*innen zugänglich gemacht. Beschäftigt man sich genauer mit diesem Webservice, fällt auf dass es hier mehrere Sicherheitslücken gibt. Beispielsweise sind Bestellungen mit negativem Gesamtpreis möglich. *OWASP Juice Shop* ist natürlich absichtlich ein unsicheres Tool, um damit zu üben, sich selbst auszuprobieren und daraus zu lernen. Es soll als Motivation dienen, sich mit dem Hacken auseinanderzusetzen und um Schwachstellen von Webanwendungen vorzuführen. Die Schwachstellen und Sicherheitslücken von *OWASP Juice Shop* sind vielfältiger Natur. OWASP führt Listen der bekanntesten dieser Vulnerabilities. Angeboten werden Challenges zu SQL-Injection, Cross Site Scripting-Attacken (XXS), Cross-Site Request Forgery-Attacken (CSRF), eingesetze kryptografische Funktionen wie Rabatt-Coupons und weitere eingebaute Schwachstellen. An der Zahl besteht der *OWASP Juice Shop* aus 28 eingebauten Modelle von Schwachstellen. Diese Sicherheitslücken wurden beabsichtigt genau zu einem bestimmten Zweck in die Anwendung eingepflanzt, und zwar auf eine Weise, die auch bei der echten Webentwicklung tatsächlich vorkommt. Dokumentationen sind auch bei *OWASP Juice Shop* zahlreich auf der offiziellen Homepage vorhanden.[2]

2.5 Installation von bWAPP und Juice Shop

Sowohl Installations-Anleitungen als auch ausführliche Erklärungen sind für beide Webanwendungen vorhanden und leicht verständlich.

bWAPP lässt sich auf zwei verschiedene Arten installieren. Diese unsichere Webanwendung wird von Wampserver[2] (WAMP) und XAMPP[3] unterstützt und ist auf ebenso auf einem eigenen Linux-Betriebssystem Ubuntu 64Bit zu finden. Linux hat

[2]WampServer ist eine Webentwicklungsplattform unter Windows, mit welcher die Möglichkeit besteht, dynamische Webanwendungen mit Apache2, PHP, MySQL und MariaDB zu erstellen, https://sourceforge.net/projects/wampserver/

[3]XAMPP ist ein sehr einfach zu installierendes Anwendungspaket für Linux, Solaris, Windows und Mac OS X. Das Paket enthält den Apache-Webserver, MySQL, PHP, Perl, einen FTP-Server und phpMyAdmin, https://sourceforge.net/projects/xampp/

bereits alle wichtigen Bestandteile von *bWAPP* vorinstalliert. Für die Umsetzung des praktischen Teils wird VirtualBox von Oracle sowie BeeBox 1.6.7 verwendet.

Um *OWASP Juice Shop* zu installieren, gibt es mehrere Methoden. NodeJs oder DockerContainer sind zu installieren. Der Einstieg direkt auf der Plattform ist ebenfalls eine Möglichkeit. OWASP bietet eine ausführliche Installations- und Anwendungsdokumentation an.

Kapitel 3

Vergleich und Architektur von bWAPP und OWASP Juice Shop

Die Hauptidee dieser beiden Vulnerable Web Applications besteht darin, den Anwender*innen verschiedene Sicherheitsaspekte beizubringen und Hacking-Herausforderungen auf unterschiedlichen Levels zu meistern. In folgendem Abschnitt wird die Architektur von *bWAPP* und *OWASP Juice Shop* beschrieben.

3.1 Vergleich der beiden Vulnerable Web Applications

Das folgende Diagramm zeigt die allgemeinen Kommunikationspfade zwischen Client, Server und Datenschichten:3.1

3.2 Architektur von bWAPP und OWASP Juice Shop

1.OWASP Juice Shop

OWASP Juice Shop ist eine reine Webanwendung, welche in JavaScript und TypeScript implementiert ist. Im Frontend wird Angular Framework verwendet, um eine sogenannte Single Page Application zu erstellen. Das Layout der Benutzeroberfläche ist die Implementierung von Googles Material Design unter der Verwendung von Angular Material Komponenten. Zur Erreichung der Reaktionsfähigkeit wird das Angular Flex-Layout verwendet. Von der Font Awesome library stammen alle verwendeten Symbole.

Javascript wird ebenso im Backend als Programmiersprache verwendet. Eine Express Anwendung wird auf einem Node.js Server gehostet und liefert den clientseitigen Code an den Browser. Auch wird dem Client die erforderliche Backend-Funktionalität über eine RESTful-API zur Verfügung gestellt. Als Datenbank wurde SQLite ausgewählt. Dies macht es einfach, die Datenbank programmgesteuert von Grund auf neu zu erstellen, ohne einen dedizierten Server. Sequelize und Finale-Rest werden als

	bWAPP	**OWASP Juice** Shop
Bugs(Modelle)	over 100	28
Security levels	3	6
Challenges	91	73
Language	English	Multi-Language Support
Enviroment	Linux,	Linux
	Windows	Windows
	Mac	
Compatibility	VMware Player	NodeJS,
	VMware Fusion	DockerContainer
	Oracle VirtualBox	Virtualbox by Vagrant
		Platform
Fuzzing possibilities	Yes	Yes
Cheat Sheet	Available	Available
Architecture	Open Source PHP application	JavaScript
	Backend MySQL database	NodeJS BackEnd
	Hosted On Linux/Apache/IIS	Angular7 FrondEnd
	Supported on WAMP or XAMPP	CTFd-Server

Tabelle 3.1: bWAPP und Juice Shop im direkten Vergleich, reproduziert von [1][2]

Abstraktionsebene aus der Datenbank benutzt. Dies ermöglicht die Verwendung dynamisch erstellter API-Endpunkte für einfache Interaktionen (d.H.CRUD-Operationen) mit Datenbankressourcen, wobei es weiterhin möglich ist, benutzerdefiniertes SQL für komplexere Abfragen auszuführen.

Als zusätzlicher Datenspeicher ist eine MarsDB Bestandteil des *OWASP Juice Shops*. Es ist ein JavaScript Derivat von der weit verbreiteten MongoDB NoSQL Datenbank und kompatibel mit den meisten Abfrage- und Änderungsoperationen.

Push-Benachrichtigungen, welche angezeigt werden, wenn eine Herausforderung erfolgreich gehackt wurde, sind implementiert über WebSocket Protocol. Die Anwendung bietet auch komfortable Benutzer*innen eine Registrierung über OAuth 2.0[1], damit sich Benutzer mit ihren Google-Konten anmelden können.[2]3.1

2.bWAPP

bWAPP ist eine Open-Source-PHP-Anwendung, welche eine MySQL-Datenbank verwendet. Es kann unter Linux oder Windows mit Apache oder IIS und MySQL gehostet werden. Außerdem kann es auch mit WAMP oder XAMPP installiert werden. Eine weitere Möglichkeit ist das Herunterladen von bee-box, einer mit *bWAPP* vorinstallierten virtuellen Maschine.[1]

[1]OAuth 2.0 ist ein Protokoll, bei welchem Benutzer*innen in der Lage sind einem anderen Standort eingeschränkten Zugriff auf seine Ressourcen zu erteilen, ohne Anmeldeinformationen preisgeben zu müssen, https://auth0.com/docs/protocols/oauth2

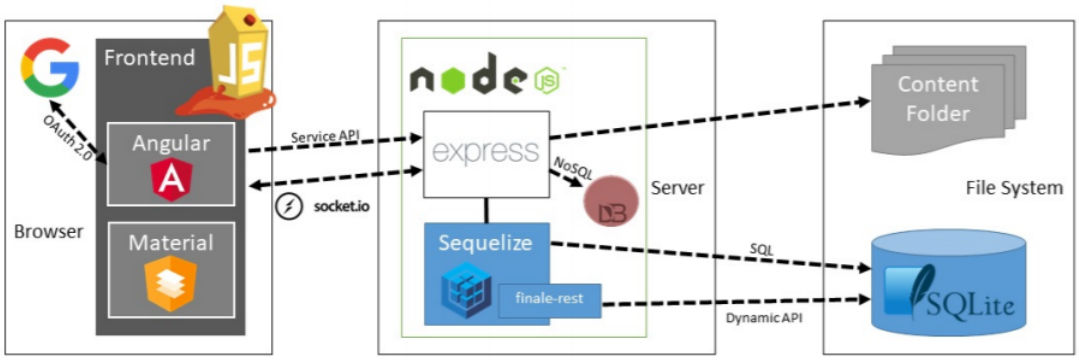


Abbildung 3.1: OWASP Juice Shop: Allgemeinen Kommunikationspfade zwischen Client, Server und Datenschichten, verwendet von [2]

3.3 Praktische Demonstration von bWAPP und Juice Shop

Bei den praktischen Demonstrationen wird SQL-Injection gewählt. Diese Hacking-Methode wird auf beiden unsicheren Webanwendungen ausprobiert, um einen eindeutigen Vergleich zu erhalten. SQL-Injection gehört zur häufigsten Methode auf der Top-10-Liste von OWASP und ist somit als sehr große Bedrohung anzusehen.

1.bWAPP Hier wird zuerst die Challenge ausgewählt. Anschließend findet eine Weiterleitung zur nächsten Webseite statt, um die Challenge ausführen zu können.3.23.3

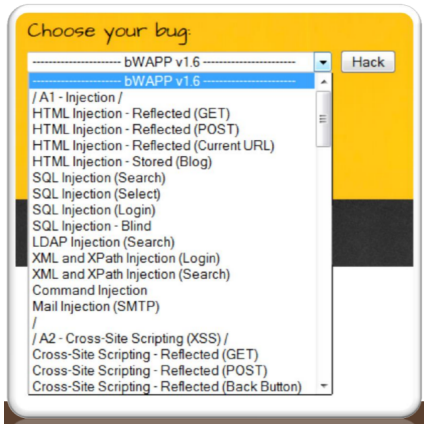


Abbildung 3.2: bWAPP Hacking-Möglichkeiten, verwendet von [1]

Challenge1 Um eine Challenge auswählen zu können, gibt es eine Registrierungspflicht. Benutzer*innen haben jedoch die Möglichkeit die Registrierung mit beliebigen Daten durchzuführen. Anschließend kann aus unterschiedlichen 'bugs' eine SQL-Injection-Aufgabe gewählt werden. Bei der ausgewählten (GET/SEARCH) SQL-Injection wird bei Eingabe eines Filmtitels, Informationen über verfügbare Filme in einer Tabelle angezeigt. Hier besteht die Möglichkeit nach einem oder mehreren Filmen zu suchen. Als Ergebnis der Suche werden die Filmdetails angezeigt. Wird die

Abbildung 3.3: Eine beliebe Challenge kann ausgewählt werden, verwendet von [1]

Eingabeschaltfläche angeklickt ohne einen Suchbegriff einzugeben, werden alle Filme angezeigt. Das gesetzte Ziel besteht darin, ein User-Passwort durch die eingesetzte Pentesting-Methode zu erhalten.3.43.5

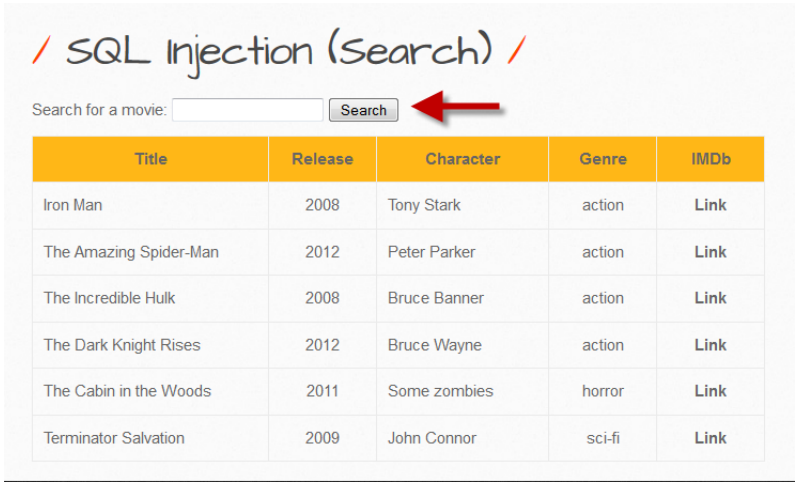


Abbildung 3.4: Suchfenster für Filme ohne Eingabe, verwendet von [1]

Abbildung 3.5: Suchfenster für Filme mit Eingabe, verwendet von [1]

Bei einer beliebigen Eingabe eines Suchbegriffes wie zum Beispiel *'iron'* werden alle Daten wie folgt angezeigt.3.6

Abbildung 3.6: Eingabe eines beliebigen Suchbegriffes, reproduziert von [1]

Eine falsche Eingabe wie etwa 1′ führt zu einem Syntax-Fehler. Als Ergebnis ist zu erkennen, dass der URL-Parameter 1′ definitiv anfällig für eine SQL-Injection ist:3.7

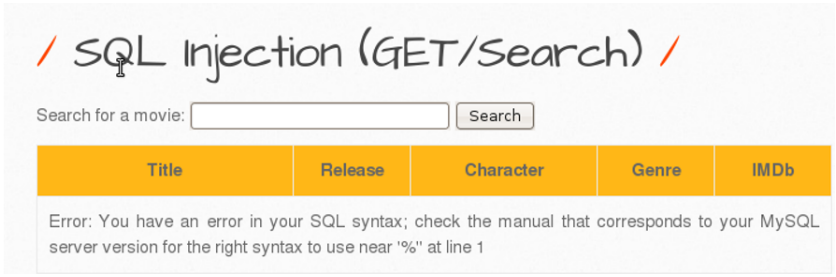


Abbildung 3.7: SQL-Injection mit einfachem Standardbefehl, reproduziert von [1]

Nach Erkennen des Syntax-Fehlers wird die URL bearbeitet. *action=search* wird gelöscht und durch *order by* 1 −− − ersetzt.3.83.9

Abbildung 3.8: Teile aus der URL werden gelöscht und ersetzt, reproduziert von [1]

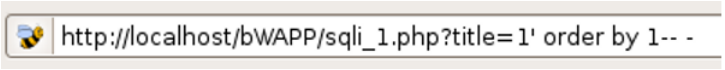


Abbildung 3.9: Formen einer neuen URL mit einem SQL-Statement, reproduziert von [1]

An der Ausgabe ist zu erkennen, das die URL-Parameter definitiv anfällig für eine SQL-Injection ist.3.10

Durchprobieren der Zahlen, bis eine sogenannte *out of table* Meldung angezeigt wird.Nach kurzer Zeit des Ausprobierens -indem die Zahl immer wieder um 1 erhöht wird bis zu einer Fehlermeldung- wird erkannt, dass es 7 Spalten gibt. Im vorgezeigten Beispielfall ist es ein leichtes Spiel, da es sich um 7 Spalten handelt und nicht um beispielsweise 600.3.113.12

Der Versuch vertrauliche Informationen anzeigen zu lassen, beginnt mit einer SQL-Union-Anweisung. Mit besagter Anweisung ist es möglich, Datenbanktabellen zusammenzuführen. Zunächst muss sichergestellt werden, dass die gleiche Anzahl der Spalten verwendet wird, wie bei der ursprünglichen SQL-Anweisung.3.13

Abbildung 3.10: Die Aussage lässt eine SQL-Anfälligkeit erkennen, reproduziert von [1]

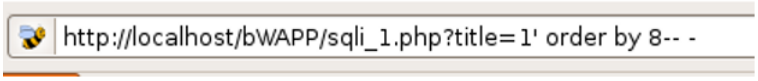


Abbildung 3.11: Herausfinden der Anzahl von Spalten durch Ausprobieren, reproduziert von [1]

Abbildung 3.12: zeigt, dass es eine Anzahl von Spalten ¡ 8 gibt, reproduziert von [1]

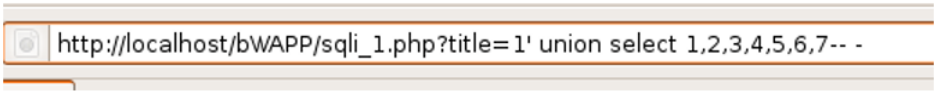


Abbildung 3.13: SQL Union-Anweisung um Datenbanktabellen zusammenführen, reproduziert von [1]

Im Ergebnis ist ablesbar, dass Spalte 5 einem Character entspricht.3.14

Abbildung 3.14: zeigt die Anzahl verschiedener Spalten einer Tabelle, reproduziert von [1]

Erneut wird die URL verändert, um sich an das gewünschte Kennwort heranzutasten.3.15 Das Ergebnis wirft den *Character bWAPP* aus.3.16

Ab diesem Zeitpunkt besteht die Möglichkeit, die aktuelle Datenbankversion zu

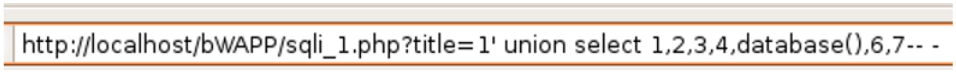


Abbildung 3.15: zeigt die Anpassung der URL, um den Character anzuzeigen, reproduziert von [1]

Abbildung 3.16: Character mit dem Namen *bWAPP* wird angezeigt, reproduziert von [1]

visualisieren:3.173.18

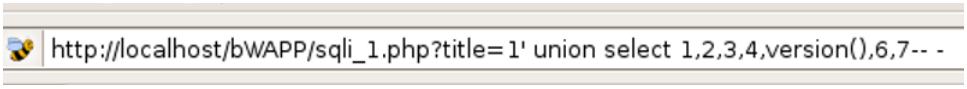


Abbildung 3.17: zeigt die Anpassung der URL, um die Version auszugeben, reproduziert von [1]

Abbildung 3.18: Nach Veränderung der URL wird die Version angezeigt, reproduziert von [1]

Tabellennamen sollen ausgegeben werden:3.193.20

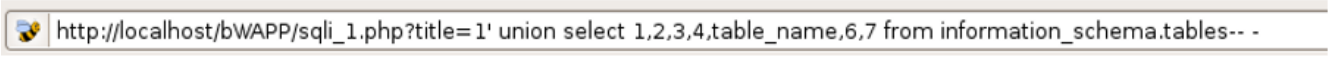
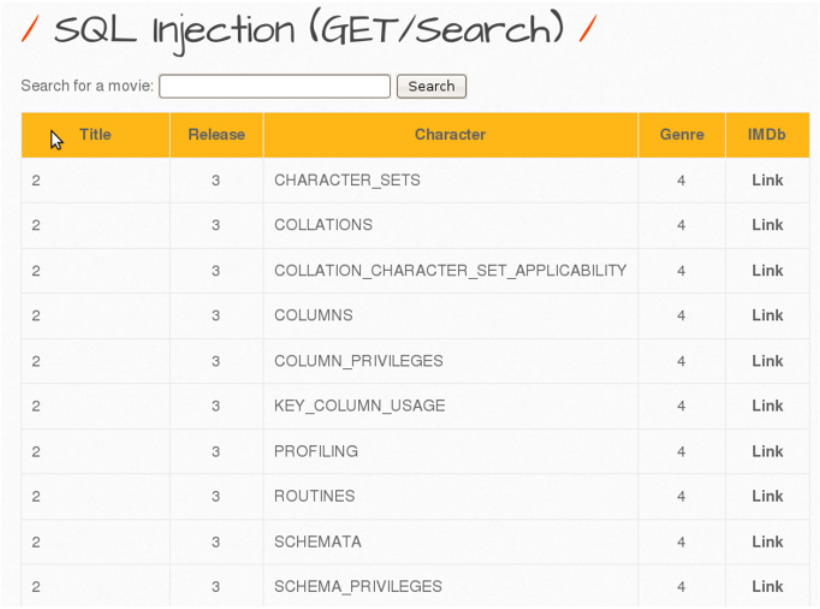


Abbildung 3.19: zeigt die URL-Anpassung zur Ausgabe aller Tabellennamen, reproduziert von [1]

Bestimmte Character der Tabelle sollen jetzt ausgegeben werden.3.213.22

Verknüpfen von Tabellennamen aus der Datenbank3.233.24

In folgendem Schritt wird der Inhalt der Spalten vom *'User'* ausgegeben.3.253.26



Title	Release	Character	Genre	IMDb
2	3	CHARACTER_SETS	4	Link
2	3	COLLATIONS	4	Link
2	3	COLLATION_CHARACTER_SET_APPLICABILITY	4	Link
2	3	COLUMNS	4	Link
2	3	COLUMN_PRIVILEGES	4	Link
2	3	KEY_COLUMN_USAGE	4	Link
2	3	PROFILING	4	Link
2	3	ROUTINES	4	Link
2	3	SCHEMATA	4	Link
2	3	SCHEMA_PRIVILEGES	4	Link

Abbildung 3.20: Ergebnis der Ausgabe aller Tabellennamen, reproduziert von [1]

Abbildung 3.21: zeigt die Ausgabe von Character aus der Datenbank, reproduziert von [1]

Title	Release	Character	Genre	IMDb
2	3	blog	4	Link
2	3	heroes	4	Link
2	3	movies	4	Link
2	3	users	4	Link
2	3	visitors	4	Link

Abbildung 3.22: Vorhandene *Character* werden aufgelistet, reproduziert von [1]

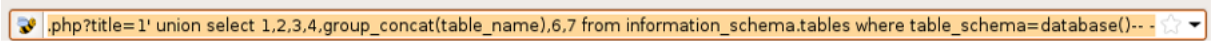


Abbildung 3.23: Verknüpfen von Tabellennamen aus der Datenbank, reproduziert von [1]

Ziel ist es an das Passwort heranzukommen, daher wird die URL so verändert dass bestenfalls Login und Passwort des Users ausgegeben werden. Das Resultat ergibt einige interessante Werte. Die Datenbank wird ausgenutzt, indem vertrauliche Daten

Abbildung 3.24: zeigt Tabellenverknüpfungen aus der Datenbank, reproduziert von [1]

Abbildung 3.25: URL-Anpassung, um 'User'-Daten anzeigen zu lassen, reproduziert von [1]

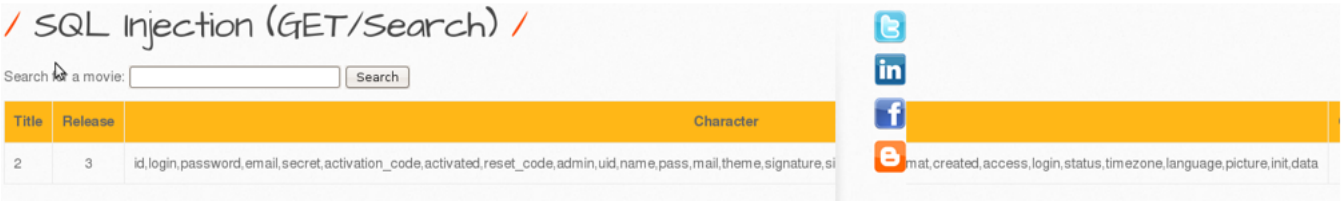


Abbildung 3.26: zeigt den Inhalt der Spalte 'User' an, reproduziert von [1]

abgerufen wurden. Der Wert des Kennwortes ist in einem Hash-Zustand gespeichert und kann noch nicht abgerufen werden.3.273.28

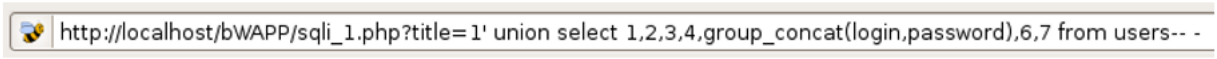


Abbildung 3.27: Die URL wird angepasst, um die Passwortdaten des Characters bWAPP anzuzeigen, reproduziert von [1]

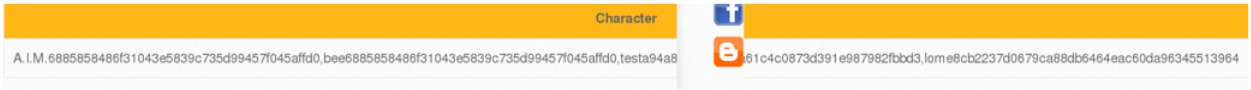


Abbildung 3.28: Das gehashte Passwort wird angezeig, reproduziert von [1]

Um das gehashte Passwort zu knacken, kann im weiteren Schritt mittels einer Passwort-Cracker Software wie beispielsweise *John the Ripper*[2] das Kennwort in Klartext ausgegeben werden[3].3.29

2.OWASP Juice Shop

[2]John the Ripper ist eine populäre Open Source Software zum Knacken von Kennwörtern. Eine Reihe von Passwort-Crackern werden in einem Paket kombiniert. Passwort-Hash-Typen werden automatisch erkannt, https://www.openwall.com/john/

[3]http://itsecgames.blogspot.com/

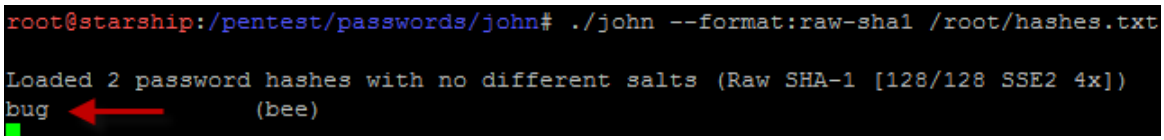


Abbildung 3.29: sqlmap mittels Jack The Ripper, reproduziert von [1]

Beim *OWASP Juice Shop* gibt es eine eigene Webseite, bei welcher die *Challenges* aufgelistet werden. Hier muss im gesamten Shop genauer nach der Ausführungsmöglichkeit dieser *Challenges* gesucht werden. War eine *Challenge* erfolgreich, wird am Bildschirm eine Benachrichtigung eingeblendet.3.30

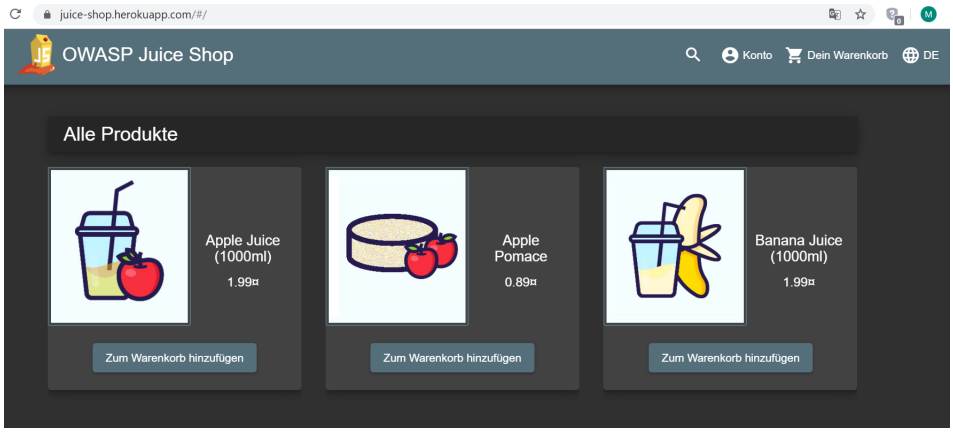


Abbildung 3.30: JuiceShop auf der Online-Plattform, verwendet von [2]

Vorgehensweise Challenge 1:
Für den Versuch sich als *Admin* einzuloggen, wird folgender Befehl verwendet: '*or true* −−. Innerhalb einer kurzen Zeitspanne war es möglich mittels diesem SQL-Befehl und einem zufällig gewählten Passwort als *Admin* eingeloggt zu werden. *True* als SQL-Befehl bedeutet, dass dieses Ergebnis immer der Wahrheit entspricht. Die *double hyphen* danach bedeuten, dass jegliche Zeichen welche nach dem *true* kommen einem Kommentar entsprechen. In beschriebenem Fall hat das *Login* unkompliziert und schnell funktioniert.3.313.32

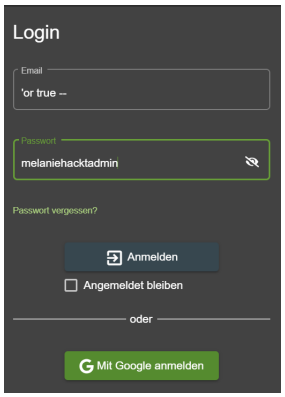


Abbildung 3.31: Login-Versuch als Admin, reproduziert von [2]

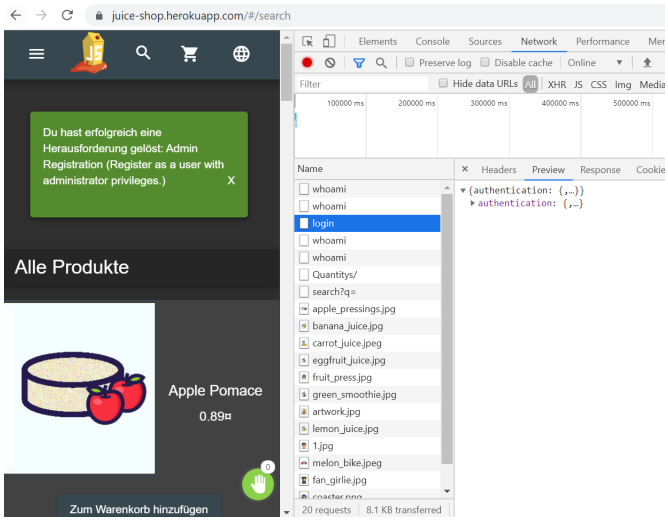


Abbildung 3.32: *Admin* Registrierung mittels simpler SQL-Injection, reproduziert von [2]

Kapitel 4

Fazit

Tools wie *bWAPP* und *OWASP Juice Shop* sind sehr lehrreich und interessant, um spielerisch Hacking-Aufgaben in verschiedenen Schwierigkeitsgraden auszuprobieren. An einem SQL-Injection Beispiel (GET/Search) wird aufgezeigt, dass innerhalb sehr kurzer Zeit Kennwörter beliebiger Benutzer*innen geknackt werden können. Mit der unaufhaltsamen Weiterentwicklung der Technik und heutigen Fülle an Informationen ist es umso wichtiger, sehr gut über die Gefahren aus dem *World Wide Web* informiert zu sein. Wie aus verschiedenen Studien hervorgeht, kann die Wahl eines Kennwortes nachweislich positive Auswirkungen auf die Datensicherheit der Benutzer*innen haben[9]. Viele Informationen und Daten, welche nicht für mich bestimmt waren, konnten abgerufen, und somit mittels *Brute Force Attacke*[1] ein Passwort geknackt werden. In weiteren Schritten besteht die Möglichkeit mittels einem Tool namens *sqlmap*[2] die Datenbank des Betriebssystems zu übernehmen.

4.1 Zusammenfassung

Die beiden unsicheren Webservices *bWAPP* und *OWASP Juice Shop* wurden vorgestellt und miteinander hinsichtlich Installation, Architektur sowie in der Anwendung verglichen. Für die Durchführung des praktischen Teils wurde SQL-Injection (GET/SEARCH) ausgewählt. Es zeigt sich, dass Penetration Testing sinnvoll ist, um Schwachstellen von Organisationen zeitnah aufzudecken. Unsichere Webservices zielen darauf ab, sich mit dem Thema des präventiven Risikomanagement zu beschäftigen. Neben der Anreicherung von praktischen Wissen über viele verschiedene Schwachstellen kommt hier auch der Spaßfaktor nicht zu kurz. Die Gestaltung beider Webservices wirkt sehr freundlich und übersichtlich. Durch die verschiedenen Schwierigkeitsgrade in den Challenges, regelmäßige Anpassungen sowie limitierte Challenges an Weihnachten und Ostern sind Benutzer*innen in der Lage ihre Fähigkeiten weiter auszubauen und zu verbessern.

[1] Mittels Trial-and-Error-Prinzip wird versucht an vertrauliche Daten heranzukommen.

[2] sqlmap ist ein Open-Source-Pentest-Tool, welche mittels SQL-Injection automatisiert zu einer Übernahme der Datenbank führen kann, http://sqlmap.org/

Abbildungsverzeichnis

Literaturverzeichnis

[1] Malik Mesellem, *LaTeX: What is bWAPP? Introducing an extremely buggy web application* , MME BVBA , Menen, Belgien 1st edition, 2014. https://www.mmebvba.com/sites/default/files/downloads/ 5, 8, 9, 10, 11, 12, 13, 14, 15, 16

[2] Björn Kimminich, *LaTeX: OWASP Juice Shop*, OWASP, Hamburg, Deutschland 1st edition, 2019. https://www2.owasp.org/www-project-juice-shop/ 5, 8, 9, 16, 17

[3] Netsparker, *LaTeX: What is the SQL Injection Vulnerability?*, Kalpa Computing, Austin, USA 1st edition, 2019. file=https://www.netsparker.com/blog/web-security/sql-injection-vulnerability/ 4

[4] Priyank Bhojak and Vatsal Shah and Kanu Patel and Deven Gol, *LaTeX: Automated Web Application Vulnerability Detection With Peneteration Testing*, Kalpa Computing, Austin, USA 2nd edition, 2017. 2

[5] P.N.Joshi and N.Ravishankar and M.B.Raju and N.C., *LaTeX: Encountering SQL Injection in Web Applications*, India pages= 257-261, 2018. 1

[6] N.Singh and M.Dayal and R.S.Rawnand S.Kumar, *LaTeX: SQL injection: Types, methodology, attack queries and prevention*, Ambedkar Institute of Advanced Communication Technologies and Research,New Delhi, India pages= 2872-2876, 2016. 1

[7] Bacudio, Aileen and Yuan, Xiaohong and Chu, Bei and Jones, Monique, *LaTeX: An Overview of Penetration Testing*, International Journal of Network Security and Its Applications, Carolina, USA pages= 19-38, 2011. 3

[8] OWASP, *LaTeX: OWASP-TOP-10 2017?*, OWASP, Deutschland, 2017. file=https://www.owasp.org/ 3

[9] N.Ibrahim and H.Sellahewa *LaTeX: Touch gesture-based authentication: A security analysis of Pattern Unlock (IEEE)*, Department of Applied Computing University of Buckingham, Buckingham, United Kingdom pages= 1-8, 2017. 18