



## Web Application Penetration Test buggybook

Technisches Management Summary & Einschätzung des Sicherheitsniveaus

Zürich, 01.04.2013

Version 1.0

Erstellt für:

**BuggyBook AG**

**BuggyBook AG**

Demoweg 22  
8000 Zürich

**Herr Hans Muster**

IT-Security Manager  
Tel +41-98-121-12 99  
E-Mail: admin@buggybook

Erstellt von:

**Protect7 GmbH**

Franklinstrasse 7  
8050 Zürich  
Tel +41-44-515-68 68  
Fax +41-44-515-68 69

**Ihr Ansprechpartner**

Tel +41-44-515 68 68  
E-Mail: info@protect7.com

## Inhalt

1	Einleitung .....	3
1.1	Management Summary .....	3
2	Einleitung und Beschreibung Testaufbau .....	5
2.1	Struktur dieses Dokuments .....	5
2.2	Testumfang und Applikations-Kontext.....	5
2.3	Abgrenzung .....	6
3	Resultate & Einschätzung Sicherheitsniveau .....	7
3.1	Legende .....	7
3.2	Auflistung der gefundenen Schwachstellen .....	7
3.3	Ausführung .....	9
3.3.1	A1: Injection .....	9
3.3.2	A2: Cross Site Scripting .....	19
3.3.3	A3: Broken Authentication and Session Management.....	22
3.3.4	A4: Insecure Direct Object References .....	27
3.3.5	A5: Cross Site Request Forgery (CSRF) .....	29
3.3.6	A6: Security Misconfiguration .....	30
3.3.7	A7: Insecure Cryptographic Storage .....	34
3.3.8	A8: Failure to Restrict URL Access .....	35
3.3.9	A9: Insufficient Transport Layer Protection .....	36
3.3.10	A10: Unvalidated Redirects and Forwards.....	37
3.3.11	Nicht kategorisierte Verwundbarkeiten.....	38
4	Empfohlene Massnahmen .....	43
5	Anhang.....	44
5.1	Tabellenverzeichnis .....	44

## 1 Einleitung

### 1.1 Management Summary

Die Firma BuggyBook AG hat Protect7 mit der Sicherheitsüberprüfung der Webapplikation buggybook.p7net betraut. Die Sicherheitsüberprüfung wurde auf einer genauen Kopie der produktiven Systeme durchgeführt. Die gefundenen Schwachstellen entsprechen somit der realen Situation und die gefundenen Schwachstellen können tatsächlich ausgenutzt werden. Die getestete Applikation ist einerseits eine Visitenkarte des Unternehmens, andererseits werden in der Applikation Unternehmensdaten verarbeitet. Aufgrund dieser Betrachtung wird von der Applikation ein entsprechendes Sicherheitsniveau erwartet. Während den Tests wurden diverse Schwachstellen mit hohem Risiko sowie auch mit mittlerem und geringem Risiko erkannt. Die Tests wurden sowohl im authentisierten als auch im anonymen Kontext durchgeführt. Für die Tests standen uns verschiedene Kunden-Accounts mit unterschiedlichen Berechtigungen zur Verfügung.

Aufgrund der Resultate muss der Applikation ein sehr schlechtes Sicherheitsniveau ausgewiesen werden.

#### Schwachstellen der Webapplikation

Bei der Sicherheitsüberprüfung der Webapplikation wurden einige Schwachstellen im authentifizierten Bereich gefunden. Um diese Schwachstellen auszunutzen, muss ein Angreifer an einen gültigen Account kommen. Es gibt jedoch auch kritische Schwachstellen, welche im nicht authentifizierten Bereich liegen und somit auch ohne einen gültigen Account ausgenutzt werden können.

Die gefundenen Cross-Site-Scripting-Schwachstellen (XSS) ermöglichen es einem Angreifer an Informationen anderer Kunden zu gelangen. Er kann dazu das Vertrauen der Kunden in die Firma Buggybook AG missbrauchen und durch eingeschleusten Code, welcher beim Kunden über die Seite Buggybook AG ausgeführt wird, nach kritischen Informationen fragen, bspw. Passwörter oder Kreditdaten und so weiter.

Ein Angreifer kann über SQL und XML Injection sensitive Daten auslesen, die Datenbank manipulieren und eine Web-Shell einrichten.

Durch Datei-Uploads können beliebige Dateien auf den Server geladen werden und eine Web-Shell eingerichtet werden. Dadurch lässt sich das System kompromittieren. Daraus folgend bietet sich vielleicht die Möglichkeit, weitere, allenfalls interne Systeme, über dieses kompromittierte System zu attackieren.

Das Fehlen korrekter Autorisierungsmechanismen in weiten Teilen der Applikation, erlaubt einem Angreifer Administrationsaufgaben wie Benutzer verwalten und Transaktionen verwalten durchzuführen. Es ist für einen Angreifer möglich, einen Root Benutzer zu erstellen und mit diesem jeden beliebigen Benutzer der Applikation zu manipulieren.

Die Applikation fängt interne Fehlermeldungen nicht ab und gibt Angreifern so wertvolle Informationen weiter. Es sollten alle Applikationsfehler intern abgefangen werden und dem Benutzer wenn überhaupt nur generische Fehlermeldungen gezeigt werden.

Eine unsichere direkte Objektreferenz erlaubt es, applikationsinterne Dateien auf dem Server einzusehen.

Die verwendete Version des Struts Frameworks ist veraltet und hat bekannte Sicherheitslücken, die ausgenutzt werden können. So ist es möglich, beliebige Systemkommandos abzusetzen.

Die Funktion zum Zurücksetzen des Passwortes verwendet eine schwache Verschlüsselung, was es ermöglicht, die Accounts von anderen Benutzern zu übernehmen.

Benutzer mit gültigen Vouchers können sich durch Manipulation eines Hidden Fields beliebig grosse Beträge gutschreiben lassen.

## Zusammengefasst sehen wir folgende Empfehlungen

- Die Schwachstelle mit hoher Priorität sollte sofort behoben werden, die aufgetretenen Schwachstellen mit mittlerer Priorität sind in einem Zeitfenster von ca. drei Monaten zu beheben
- Implementieren einer Input- und Output Validierung
- Implementierung eines korrekten Authentifizierungs- und Autorisierungsmechanismus
- Implementierung eines weiteren Error Handling
- Analyse ob der Server bereits kompromittiert wurde

DEMO BERICHT

## 2 Einleitung und Beschreibung Testaufbau

Protect7 wurde von der Firma BuggyBook AG, vertreten durch Herrn Hans Muster, beauftragt, die Webapplikation buggybook einem manuellen Web Penetration Test zu unterziehen. Dieses Kapitel liefert die Informationen, welche für die Nachvollziehbarkeit der Testresultate notwendig sind.

### 2.1 Struktur dieses Dokuments

Der erste Abschnitt „Management Summary“ fasst die gefundenen Schwachstellen zusammen und bringt sie in einen gemeinsamen Kontext mit möglichen Bedrohungen. Das aktuelle Kapitel beschreibt die all-gemein gültigen Parameter der Überprüfung. Im nachfolgenden Abschnitt „Resultate & Einschätzung Sicherheitsniveau“ sind sämtliche durchgeführten Tests sichtbar, welche zur Entdeckung einer Schwach-stelle geführt haben. Während der Überprüfung wurden unzählige Tests durchgeführt, die die Applikation erfolgreich bestanden hat, diese sind in diesem Dokument nicht enthalten.

Bei den gefundenen Schwachstellen stehen die Nachvollziehbarkeit sowie kurze Hinweise auf mögliche Ursachen und Lösungsansätze im Vordergrund. Im Abschnitt „Empfohlene Massnahmen“ werden die empfohlenen Massnahmen, um die Risiken auf ein akzeptables Niveau zu senken, zusammengefasst. Die Umsetzung und genaue Beschreibung der einzelnen Sicherheitsmassnahmen kann komplex und zeitaufwändig sein, weshalb die Zusammenfassung keinesfalls einer Spezifikation der empfohlenen Mas-snahmen entspricht. Protect7 kann Sie durch Ihre Erfahrung in der Softwareentwicklung bei der Umset-zung von Sicherungsmassnahmen unterstützen. Der Fokus dieses Penetration Tests liegt in der Identifi-kation von Schwachstellen.

### 2.2 Testumfang und Applikations-Kontext

Im Zusammenhang mit einem Web Applikation Penetration Test arbeiten wir von einer oder mehreren Applikationen und deren Kontext, in welchem gewährleistet. Das Produkt „Protect7“ basieren sich auf den Standards des Schweizerischen, der gen. TIG Richtlinien. In diesem Fall wurde der Test aus dem Protect7 Tracking ab über das Internet durchgeführt. Die Prüfer beschränkt den Applikations Umfang, welcher in der Regel durch das Erstellen einer Adressenliste (URLs) erreicht gemacht wird. Ein Kontext kann sich aber auch aus anderen Mechanismen ableiten, einer bestimmten Source IP oder einem definierten User Agent Header unter, möglicherweise Nutzung eines Applikations Kontexts in den Diensten oder Verfügbarkeiten von Ressourcen aufgrund von spezifischen Benutzer Attributen.

Der Web Applikation Penetration Test umfasst die Domain buggybook.ch im Kontext eines anonymen Be-nutzers und im Kontext eines privilegierten Benutzers. Ein Administrationskonto wurde zur Verfügung gestellt, der Administratorrechte wurde jedoch auf Kundenbereich nicht explizit überprüf.

Folgende User Accounts standen für die Tests zur Verfügung

Account	Beschreibung
Admin	Administrationskonto
Standard	Benutzerkonto
Privileged	Alternatives Benutzerkonto

Tabelle 1 - User Accounts

## 2.3 Abgrenzung

Die nachfolgende Tabelle listet spezifische Applikations-Teile auf, welche vom Penetration Test ausgeschlossen wurden.

Exkludiert	Beschreibung
buggybook/buggymail/	Das Webmailsystem von BuggyBook AG wurde auf Anweisung des Kunden nicht getestet.
dev.buggybook/	Die Entwicklungsplattform von BuggyBook AG wurde auf Anweisung des Kunden nicht getestet.
buggybook/admin/	Der Administratorbereich wurde nicht explizit untersucht. Es wurde nur geprüft, ob mit Berechtigungen niedriger Stufe Administrationsaufgaben ausgeführt werden können.

**Tabelle 2 - Exklusions-URLs**

Der Application Penetration Test eruiert nur Sicherheitslücken auf Applikationsebene. Die Sicherheit der zugrundeliegenden Systeme wurde vorgängig mit einem Network Penetration Test untersucht. Diesbezüglich sei auf den Bericht „Network Penetration Test buggybook“ verwiesen.

## 3 Resultate & Einschätzung Sicherheitsniveau

Die getestete Applikation weist einige schwerwiegende Sicherheitsmängel auf. Grundlegende Sicherheitsanforderungen wie Input- und Output Validierung, sauberes Error-, Authentication- und Autorisation Handling sind nicht genügend umgesetzt. Die Applikation verfügt an mehreren Stellen über keinen Mechanismus zur Verhinderung von SQL oder XML Injection. Die Applikation weist Schwachstellen gegenüber Cross-Site Scripting Angriffen auf und ermöglicht das Ausbrechen aus dem vorgesehenen Kontext. Darüber hinaus kann das Dateisystem auf dem Server durchsucht und Dateien herunter- und hochgeladen werden. Daraus ergeben sich Risiken für das ganze System und die darin enthaltenen Kundendaten.

Je nach Netzwerkeinbindung könnten mit den entdeckten Schwachstellen Angriffe auf weitere interne Systeme gestartet werden.







### 3.1 Legende

In diesem Bericht wird folgende Bewertung der Schwachstellen angewendet

Symbol	Beschreibung
	Information
	Schwachstelle mit hohem Risiko
	Schwachstelle mit mittlerem Risiko
	Schwachstelle mit tiefem Risiko

Tabelle 3 - Legende

### 3.2 Auflistung der gefundenen Schwachstellen

Ref.	Description	Class. <sup>1</sup>
(2)	Der Parameter „id“ auf der Seite /books/view.html ist im authentisierten und nicht authentisierten Kontext auf einen SQL Injection Angriff anfällig.	
(3)	Der Parameter „sortby“ der „advanced“ Suche auf der Seite books/overview.html ist im authentisierten und nicht authentisierten Kontext auf eine blind SQL Injection anfällig.	
(4)	Der Parameter „id“ auf der Seite books/view.html ist im authentisierten und nicht authentisierten Kontext auf einen Blind SQL Injection Angriff anfällig. Es ist möglich, beliebige SQL Code auszuführen.	
(6)	Das Feld „file“ auf der Seite images/messages/messages.html ist im authentisierten Kontext auf einen persistent Cross Site Scripting Angriff anfällig.	
(7)	Das Feld „file“ auf der Seite images/messages/messages.html ist im authentisierten Kontext auf einen persistent Cross Site Scripting Angriff anfällig.	
(8)	Es ist möglich, eine bestehende Session auf einem anderen Client zu übernehmen, z.B. ein Webbrowser. Prüfung auf verschiedene Merkmale wie IP-Adresse, User-Agent, Header, multiple Sessions oder ähnliches ist.	

<sup>1</sup> Bewertung siehe Legende 3.1

(9)	Das Session Cookie wird nicht mittels HttpOnly Flag geschützt	A
(13)	Die Admin-Rechte werden nur durch einen Cookie (JSESSIONID) bestimmt. Durch Manipulation an den Cookies ist es möglich, Administrationsrechte zu erlangen.	A
(14)	Der Parameter „path“ auf der Seite /help/index.html verweist direkt auf eine File-Resource. Dadurch ist es in autorisierten und nicht autorisierten Kontext möglich, auch Dateien ausserhalb des gewollten Benutzerschnitts auszuliefern.	A
(16)	Die eingesetzte Version 2.3.1.1 des Struts-Frameworks hat eine bekannte Schwachstelle (http://cve.mitre.org/cve/2007/3918)	A
(17)	Die Tokens zum authentifizieren des Benutzerschnitts werden mit einem schwachen Algorithmus erstellt.	A
(18)	Die Applikation verwendet keine Verschlüsselung und überträgt alle Informationen in Klartext.	A
(19)	Die Funktion zum Update eines Profildates auf der Seite /myaccount/profile.html erlaubt die Hochladen von ausführbaren Dateien.	A
(20)	Ein Benutzer mit einem gültigen Token kann sich beliebige Bereiche anschauen.	A
(5)	Die Seite /bookingsmanagement.html ist ein Parameter „path“ auf eine File-Resource an.	B
(10)	Ein Admin-Wireless des Benutzerschnitts auf keine http-SessionID gesetzt.	B
(12)	Session-Protokolle ist möglich. Es kann jede HTTP-Schwachstelle ausgenutzt werden, um dem Opfer eine beliebige SessionID zu geben.	B
(15)	Die Applikation gibt alle internen URLs/URLen an den Benutzer weiter.	B
(11)	Die Session ID wird im Cookie JSESSIONID gespeichert und besitzt eine ausreichende Länge. Die Varianz der generierten SessionIDs ist akzeptabel, hat aber Verbesserungspotential.	C

**Tabelle 4 - Übersicht der gefundenen Schwachstellen**



## 3.3 Ausführung

### 3.3.1 A1: Injection

Injection Schwachstellen wie SQL, OS oder LDAP Injection entstehen, wenn ungefilterte Daten als Teil eines Commands oder Query an einen Interpreter gesendet werden. Daten eines Angreifers können so unbeabsichtigte Kommandos absetzen oder auf unautorisierte Daten zugreifen.

Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.

#### Referenz:

[http://de.wikipedia.org/wiki/SQL\\_Injection](http://de.wikipedia.org/wiki/SQL_Injection)

[https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)

[https://www.owasp.org/index.php/Top\\_10\\_2010-A1](https://www.owasp.org/index.php/Top_10_2010-A1)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(1)	<b>Analyse LDAP Injection</b>	<b>Test:</b> Die Applikation wird hinsichtlich ihrer Verwundbarkeit gegen LDAP Injection Angriffe untersucht. Dabei wird untersucht, ob Daten ausgelesen, gespeichert oder Sicherheitsmechanismen ausgehebelt werden können. <b>Analyse:</b> Es wurden keine LDAP Injection Schwachstellen gefunden. <b>Risiko:</b> - <b>Empfehlung:</b> -	I
(2)	<b>Analyse SQL Injection</b>	<b>Test:</b> Die Applikation wird hinsichtlich ihrer Verwundbarkeit gegen SQL Injection Angriffe untersucht. Dabei wird untersucht, ob Daten aus der Datenbank ausgelesen werden können und/oder das System kompromittiert werden kann. <b>Analyse:</b> Der Parameter „id“ auf der Seite /books/view.html ist im authentisierten und nicht authentisierten Kontext auf einen SQL Injection Angriff anfällig.	A



Durch ein „union select“ kann herausgefunden werden, wie die selektierten Spalten in der Seite zugeordnet sind.

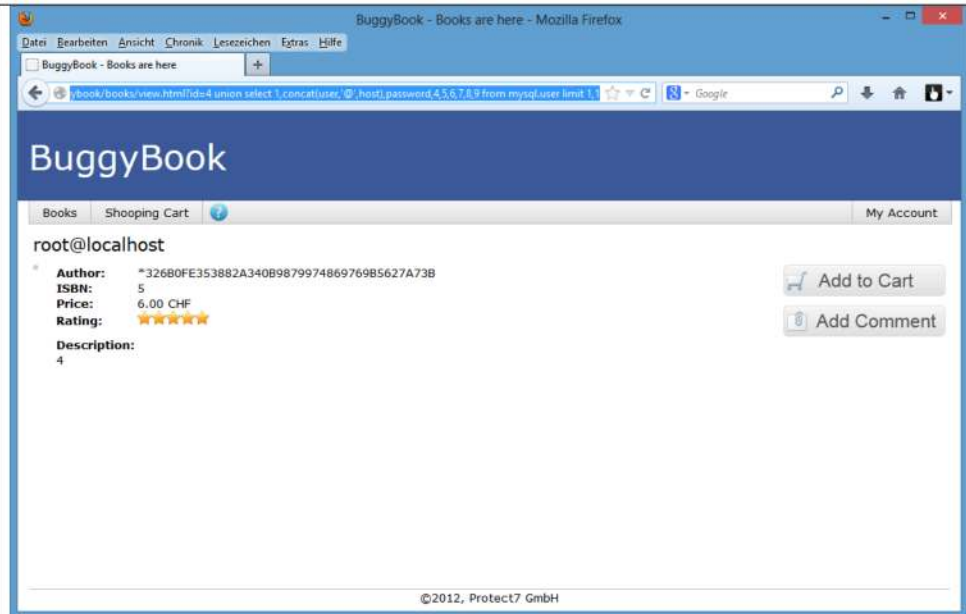


Nun kann Inhalt einer beliebigen sql-Tabelle selektiert und angezeigt werden. Zum Beispiel die Tabelle mysql.user mit dem ersten Benutzer (root).

Anfrage:

```
http://buggybook/books/view.html?id=4%20union%20select%201,concat%28user,%27@%27,host%29,password,4,5,6,7,8,9%20from%20mysql.user%20limit%201,1
```

Resultat:



**Risiko:**

Durch diese Schwachstelle ist es möglich, sensitive Daten auszulesen.

**Empfehlung:**

Alle Felder müssen saubere Input Validation Richtlinien implementieren, um solche Angriffe zu verhindern. Dazu sollte wenn möglich für alle Datenbankabfragen Prepared Statements eingesetzt werden.

(3) **Analyse SQL Injection**

**Test:**

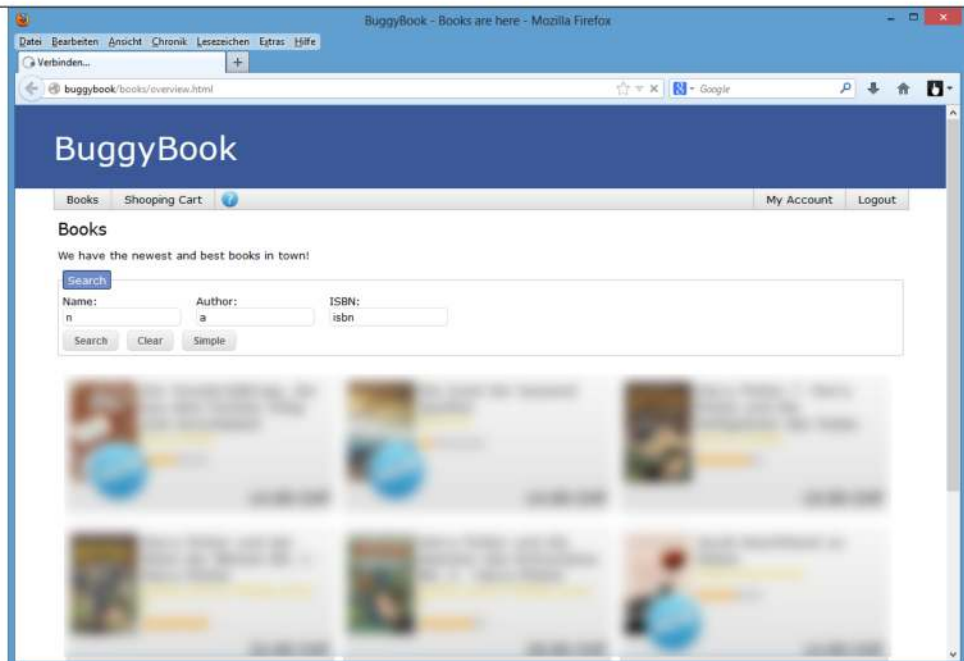
Die Applikation wird hinsichtlich ihrer Verwundbarkeit gegen SQL Injection Angriffe untersucht. Dabei wird untersucht, ob Daten aus der Datenbank ausgelesen werden können und/oder das System kompromittiert werden kann.

**Analyse:**

Der Parameter „sortby“ der „advanced“ Suche auf der Seite books/overview.html ist im authentisierten und nicht authentisierten Kontext auf eine blind SQL Injection anfällig.

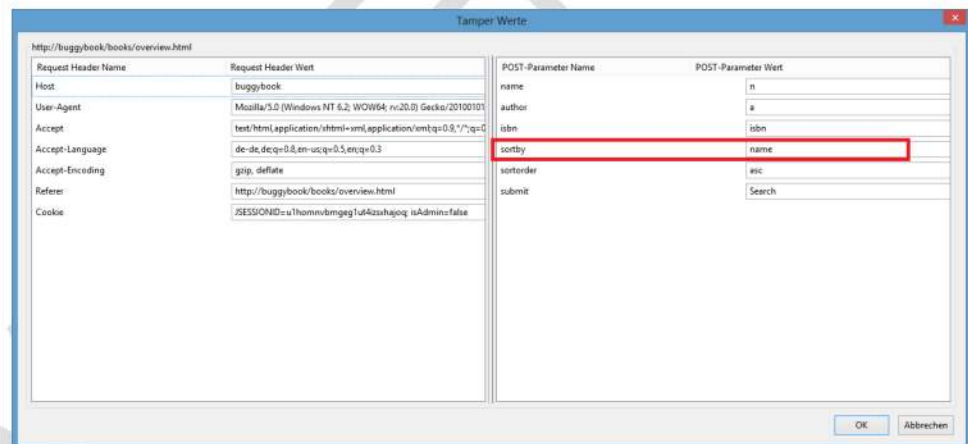
Die folgende Seite ist auf einen SQL Injection Angriff anfällig.



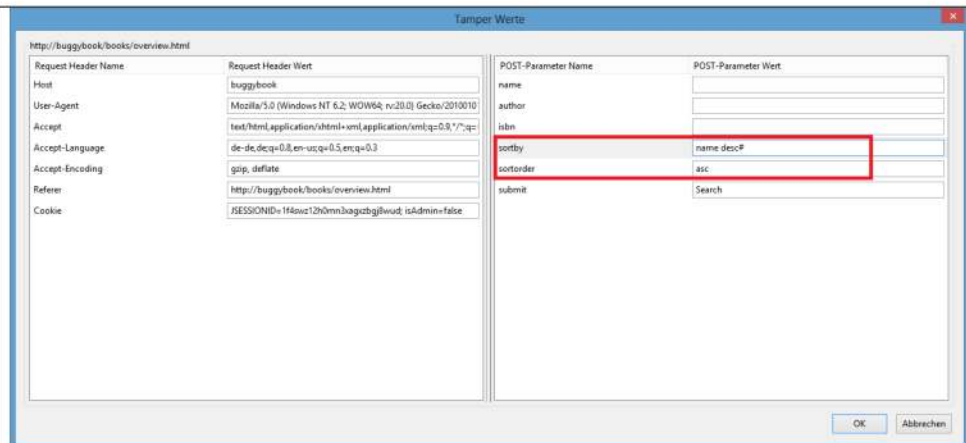


Es ist möglich beliebigen SQL Code einzuschleusen. Diese Funktion steht allen eingeloggten Benutzern zur Verfügung.

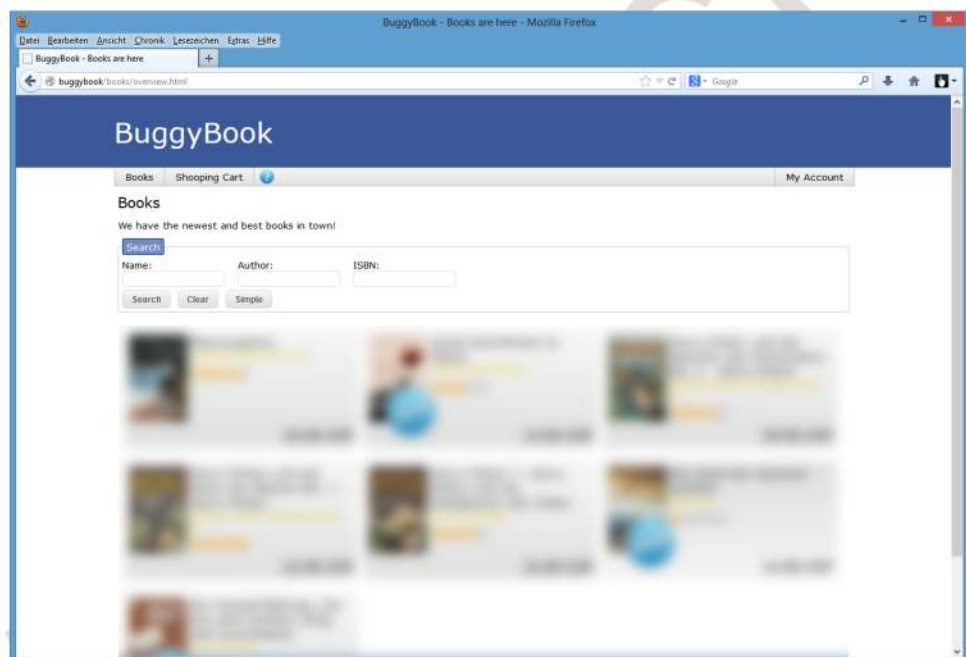
Der Parameter „orderby“ wird als POST-Variable übergeben:



Fügt man dem Parameter "orderby" weitere gültige SQL Kommandos an und kommentiert nachfolgende Kommandos aus, so werden die SQL Kommandos ausgeführt. Im folgenden Beispiel werden die Namen **absteigend** sortiert:



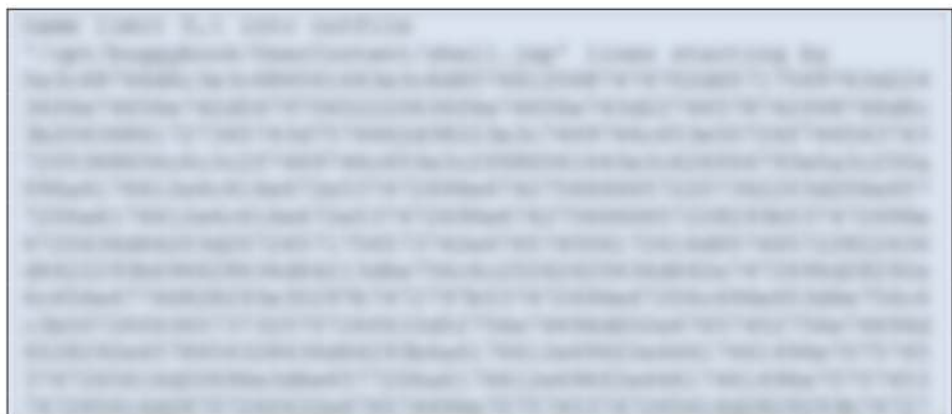
Die Bücher werden wie erwartet in umgekehrter Reihenfolge aufgelistet:



Damit wurde gezeigt, dass der Parameter „sortby“ auf Blind SQL Injections anfällig ist.

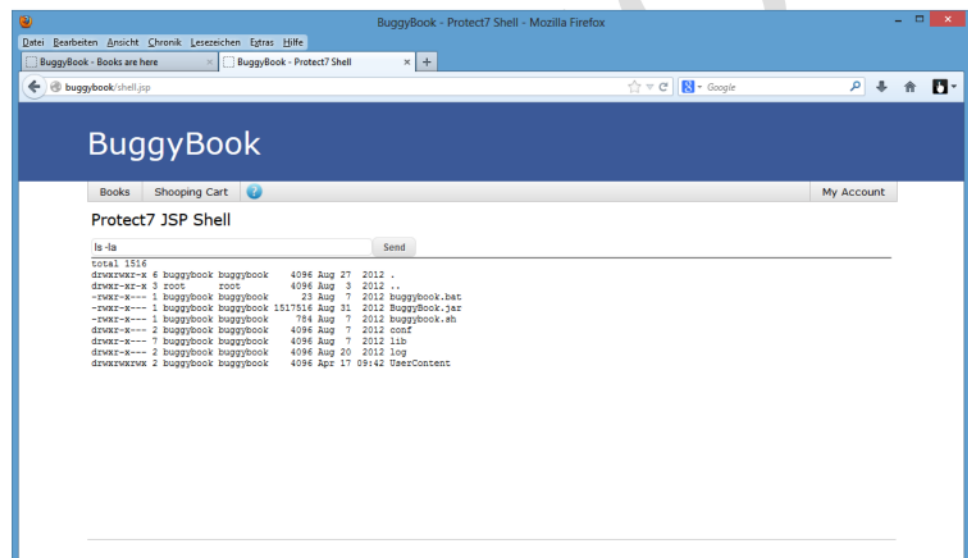
Mit dem Wissen dieser Schwachstelle können wir nun direkt auf dem Server eine Webshell platzieren.

Eingabe in Feld „sortby“:





Die Webshell mit den Rechten des Webservers:



**Risiko:**

Durch diese Schwachstelle ist es möglich, sensitive Daten auszulesen oder unter bestimmten Umständen den ganzen Server zu kompromittieren.

Mit einer Blind SQL Injection können SQL Kommandos ausgeführt werden und es kann durch (Script-unterstütztes) Ausprobieren der Inhalt der Datenbank inklusive Root-Accounts ausgelesen werden.

**Empfehlung:**

Alle Felder müssen saubere Input Validation Richtlinien implementieren, um solche Angriffe zu verhindern. Dazu sollte wenn möglich für alle Datenbankabfragen Prepared Statements eingesetzt werden und dort wo das nicht möglich ist beispielsweise explizites Whitelisting von erlaubten Eingaben.

**Referenz:**

- [http://de.wikipedia.org/wiki/SQL\\_Injection](http://de.wikipedia.org/wiki/SQL_Injection)
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- [https://www.owasp.org/index.php/Top\\_10\\_2010-A1](https://www.owasp.org/index.php/Top_10_2010-A1)
- [https://www.owasp.org/index.php/Blind\\_SQL\\_Injection](https://www.owasp.org/index.php/Blind_SQL_Injection)

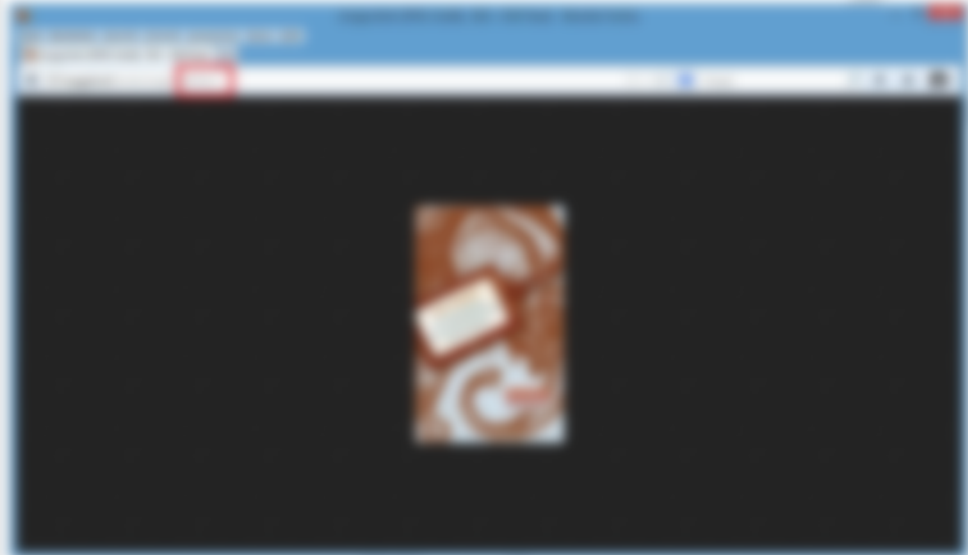
(4) Analyse SQL Injection

**Test:**

Die Applikation wird hinsichtlich ihrer Verwundbarkeit gegen SQL-Injection-Angriffe untersucht. Dabei wird untersucht, ob Daten aus der Datenbank ausgelesen werden können und/oder das System kompromittiert werden kann.

**Analysen:**

Der Parameter „id“ auf der Seite Suchergebnisse.html ist ein autorisierter und nicht autorisierter Kontext auf einen Wert SQL-Injection-Angriff möglich. Es ist möglich beliebigen SQL-Code einzuschleusen.

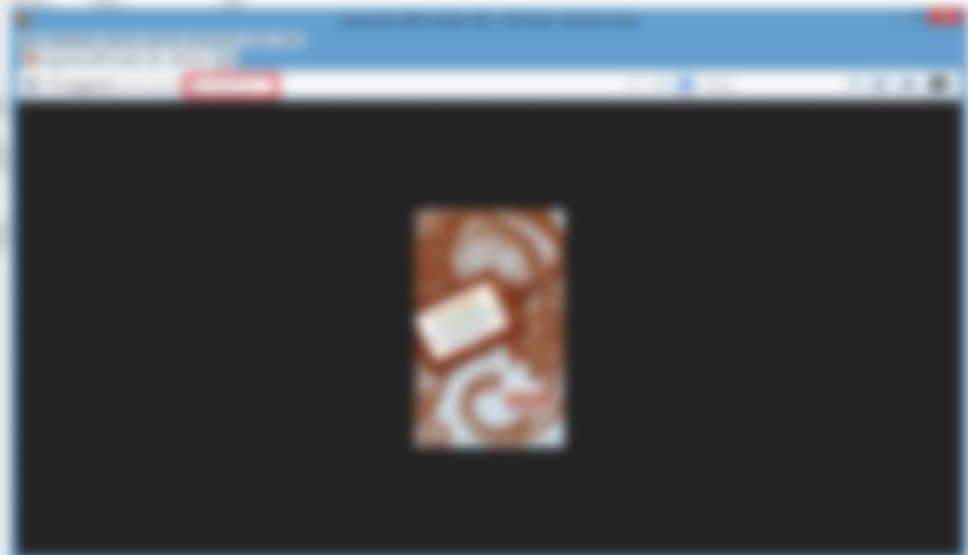


Wenn als id ein gültiger SQL-Ausdruck eingegeben wird, erscheint die vollständige Site.

Gültiger Ausdruck

id=1 UNION SELECT \* FROM users

Resultat



Wenn als id ein falscher Ausdruck eingegeben wird, erscheint die Meldung „Not found“.

Falscher Ausdruck

	 <p><b>Risiko:</b> Durch diese Schwachstelle wird möglich, sensitive Daten auslesen oder unter bestimmten Umständen den Service zu manipulieren. Als einer über SQL Injection möglich SQL-Kommandos ausgeführt werden und es kann durch diese Manipulationen Informationen der Inhalt der Datenbank in diese Form angezeigt verfügbar werden.</p> <p><b>Empfehlung:</b> Als Programmierer werden diese Schwachstellen identifizieren implementieren, um auf die Angriffe zu verhindern. Dies sollte wenn möglich für alle Datenbankabfragen. Parameter-Sensitivität angepasst werden und dort wo das nicht möglich ist bei Abfragen spezielle Filterung von unerwünschten Eingaben.</p> <p><b>Referenzen:</b>  <ul style="list-style-type: none"> <li>http://www.w3schools.com/php/php_mysql_injection.asp</li> <li>http://www.w3schools.com/php/php_mysql_injection.asp</li> <li>http://www.w3schools.com/php/php_mysql_injection.asp</li> <li>http://www.w3schools.com/php/php_mysql_injection.asp</li> </ul> </p>	
(5) Analyse XML Injection	<p><b>Risk:</b> Die Applikation wird hinsichtlich ihrer Verwundbarkeit gegen XML Injection Angriffe untersucht. Dabei wird untersucht, ob Informationen ausgelesen werden können, welche das System kompromittiert werden kann.</p> <p><b>Analysen:</b> Die Seite <code>test.php</code> wird mit dem Parameter <code>param=1</code> auf eine XML Injection geprüft. Zum Beispiel kann der Inhalt der Konfigurationsdatei auf dem lokalen Filesystem ausgelesen werden, mit der Eingabe:</p>	B



https://www.ubuntu.com/

### Risiko:



Es kann auch genutzt werden, um ein System über ein Netzwerk mit der Eingabe

```
ssh user@ip-address
```

### Risiko:



Aus der Meldung 'Connection refused' sollte ein Angriff auf den Port auf dem entsprechenden System ableitbar sein.

### Risiko:

Es können beliebige Dateien ausgelesen und Informationen über das System oder andere interne Systeme gewonnen werden.

### Empfehlung:

SSH, Firewall Hardening implementieren.

Dies gehört mindestens, Schema Hardening aktivieren, Unterbindung von Br.

		<p>Bezug externer Quellen, Unterstützung der Erfüllung externer Quellen, Ein-          arbeitung der Anzahl Kunden.</p> <p><b>Referenz</b></p> <p>https://www.unity-engine.de/unity-testing-for-ARM-architecture-201404-01-000</p>	
--	--	--	--

DEMO BERICHT


### 3.3.2 A2: Cross Site Scripting

Cross Site Scripting Schwachstellen treten dann auf, wenn die Applikation ungefilterte Daten und ohne korrektes Enkodieren zurück an den Web Browser schickt. Ein Angreifer kann so Skripte im Web Browser des Opfers ausführen. Session Hijacking, Website Defacing, Ausführung von Exploitcode etc. können die Folge sein.

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

**Referenz:**

- [https://www.owasp.org/index.php/Top\\_10\\_2010-A2](https://www.owasp.org/index.php/Top_10_2010-A2)
- [http://de.wikipedia.org/wiki/Cross-Site\\_Scripting](http://de.wikipedia.org/wiki/Cross-Site_Scripting)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(6)	<b>Analyse Cross Site Scripting</b>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Anfälligkeit für Cross Site Scripting untersucht. Es wird versucht, an unsanctionierte Stellen Applikations Code einzuschleusen, welcher für einen Angriff verwendet werden könnte. Es wird versucht das kein Applikations Code eingeschleust werden kann.</p> <p><b>Analyse</b></p> <p>Das Test „Titel auf der Seite“ zeigt, dass die Applikation keine Filter hat, um sicherzustellen, dass kein JavaScript Code eingeschleust werden kann. Es wird versucht, ein JavaScript Code in das Formularfeld einzuschleusen, welches die Seite defaced.</p>  <p>Um den XSS Code einzuschleusen, wird auf der Seite Inputformulare eingegeben, die den Titel der Seite defaced.</p> <p><b>Empfehlung</b></p> <p>Die Applikation sollte sicherstellen, dass alle Daten, die an den Webbrowser geschickt werden, korrekt escaped sind.</p>	A

		 <p><b>Risiko:</b> Grundlegend können XSS-Schwachstellen ausgenutzt werden, um eigenen Code auf dem Client-Browser auszuführen.</p> <p><b>Empfehlung:</b> Alle Front-End-Entwickler müssen sichere Input- und Output-Encoding-Richtlinien implementiert werden, um solche Angriffe zu verhindern.</p>	
(7)	<p><b>Analyse Cross Site Scripting</b></p>	<p><b>Ziel:</b> Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Cross Site Scripting untersucht. Es wird versucht, in ungeschützten Stellen JavaScript Code einzuschleusen, welcher für einen Angriff verwendet werden könnte. Es wird erwartet dass kein JavaScript Code eingeschleust werden kann.</p> <p><b>Beispiel:</b> Das Front-End auf der Seite https://www.example.com/ ist so inoffiziell, sodass es auf einen persistenten Cross Site Scripting Angriff anfällig ist.</p>  <p>Um den XSS Code einzuschleusen, wird auf der Seite</p>	<p style="text-align: center;">A</p>

Input- und Output-Streams sind im Fall TLS des Scripts angelegt. Um ein vollständiges Escaping der Script-Tags zu umgehen, müssen die Tags verschlüsselt werden.

#### Empfehlungen

Das Script sollte in einem dedizierten Container (z.B. Docker) ausgeführt werden.

Wenn die Ergebnisse der Nachweise die Nachschreibendatei erzeugen, wird das Script ausgeführt.



#### Risiko

Grundfunktionen können über Substraktionen umgangen werden, um eigenen Code auf dem Client-Endpoint auszuführen.

#### Empfehlung

Als Front-End werden mehrere Input- und Output-Encoding-Methoden implementiert werden, um solche Angriffe zu verhindern.

### 3.3.3 A3: Broken Authentication and Session Management

Applikationsfunktionen, welche Authentisierung und Session Management behandeln, sind oft nicht korrekt implementiert. Dadurch kann ein Angreifer Passwörter, Session Tokens und Keys kompromittieren, sowie andere Implementierungsfehler ausnutzen, um an Benutzeridentifikationen zu gelangen.

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.

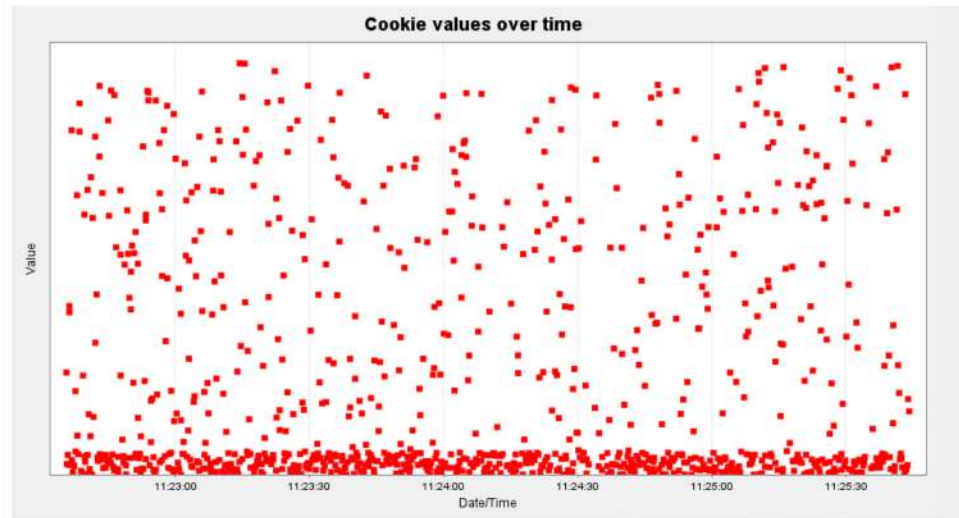
**Referenz:**

[https://www.owasp.org/index.php/Broken\\_Authentication\\_and\\_Session\\_Management](https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(8)	<b>Analyse Broken Authentication and Session Management</b>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Broken Authentication und Session Management Schwachstellen untersucht.</p> <p><b>Analyse</b></p> <p>Es wird analysiert, ob die Session-Cookies/Token auf sensible Attribute wie IP-Adresse, User-Agent, usw. gefährlicher ist, oder ob ein Angreifer die Session ohne weiteres übernehmen kann.</p> <p>Es ist möglich, eine legitime Session auf einem anderen Client zu übernehmen, z.B. ein falsches IP-Adresse, User-Agent, usw. gefährlicher ist, oder ob ein Angreifer die Session ohne weiteres übernehmen kann.</p> <p>Es ist möglich, eine legitime Session auf einem anderen Client zu übernehmen, z.B. ein falsches IP-Adresse, User-Agent, usw. gefährlicher ist, oder ob ein Angreifer die Session ohne weiteres übernehmen kann.</p> <p><b>Risiko</b></p> <p>Die Session kann grundsätzlich von einem Angreifer gestohlen und übernommen werden. Die Session-Cookies werden als sensibles Authentifizierungsmittel mit dem Client und der Web-Applikation verwendet.</p> <p><b>Empfehlung</b></p> <p>Die Abrechnung sollte ein Binding der Session auf überlebige Werte wie IP-Adresse, User-Agent, usw.</p>	A
(9)	<b>Analyse Broken Authentication and Session Management</b>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Broken Authentication und Session Management Schwachstellen untersucht.</p> <p><b>Analyse</b></p> <p>Es wird analysiert, ob die Session-Cookies vor unzulässigem Zugriff geschützt sind. Das Session-Cookie wird nicht mittels HttpOnly Flag geschützt.</p>	A



reichende Länge. Die Varianz der generierten SessionIDs ist akzeptabel, hat aber Verbesserungspotential.



Die SessionIDs sind nicht einfach vorhersagbar, konzentrieren sich aber zu sehr bei niedrigen Werten.

**Risiko:**

Wenn die generierten SessionIDs zu wenig zufällig sind, kann ein Angreifer zukünftige SessionIDs erraten und sich allenfalls in deren Kontext bewegen.

**Empfehlung:**

Einen besseren Zufallsgenerator zur Konstruktion von SessionIDs verwenden.

(12) **Analyse Broken Authentication and Session Management**

**Ziel:**  
Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Broken Authentication und Session Management Schwachstellen untersucht.

**Erwartung:** Es wird geprüft, ob Session-Funktionen richtig im Web-Session-Framework der Applikation definiert sind. SessionIDs sind sicher und zufällig generiert.

**Methodik:**  
Web-Session-Funktionen werden getestet. Es kann eine HTTP-Schwachstelle ausgenutzt werden, um ungültige oder falsche SessionIDs zu generieren.

**Wichtiges Info:**  
[Blauer Balken mit Text: 'Die Schwachstelle ist nicht auf dem Client des Testers ausgeglichen']

**Wird auf dem Client des Testers ausgeglichen:**

B



		 <p><b>Risiko:</b> Ein Angreifer kann selbst wählen, mit welcher SessionID ein Nutzer arbeitet. Dadurch kann er im Kontext des Nutzers agieren. Da die SessionID bei einem Namestransfer nicht neu generiert wird, kann er auch die Rechte des Nutzers mit dem Login übernehmen.</p> <p><b>Empfehlung:</b> Das httpOnly Flag bei den Cookies sollte gesetzt werden. Die SessionIDs sollten bei einem Kontextwechsel neu generiert werden. Es sollten keine XSS-Schwachstellen vorhanden sein.</p>	
(13)	<b>Analyse Broken Authentication and Session Management</b>	<p><b>Ziel:</b> Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Broken Authentication und Session Management Schwachstellen untersucht.</p> <p><b>Realisier:</b> Die Admin-Rechte werden nur durch einen Cookie-Wert festgelegt. Durch Manipulation des Cookies ist es möglich, Administrationsrechte zu erlangen.</p>	



Nach dem Setzen von Administrator kann der Angreifer auf alle Admin-Ressourcen zugreifen.



**Risiko:**

Ein Angreifer kann alle Rechte des Administrators übernehmen.

**Empfehlung:**

Sensitive Informationen sollten nie dauerhaft gespeichert werden. Administratoren müssen immer separat erstellt werden.


### 3.3.4 A4: Insecure Direct Object References

Ein direkter Objektverweis tritt dann auf, wenn ein Entwickler einen Verweis auf ein internes Objekt implementiert, zum Beispiel auf eine Datei, auf ein Verzeichnis oder auf Datenbank-Spalte. Ohne einen Zugriffsschutz kann ein Angreifer diese manipulieren und auf unerlaubte Daten zugreifen.

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

**Referenz:**

[https://www.owasp.org/index.php/Top\\_10\\_2007-Insecure\\_Direct\\_Object\\_Reference](https://www.owasp.org/index.php/Top_10_2007-Insecure_Direct_Object_Reference)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(14)	<b>Analyse Insecure Direct Object References</b>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Anfälligkeit für Insecure Direct Object Reference Schwachstellen untersucht.</p> <p><b>Analyse</b></p> <p>Der Parameter „id“ auf der URL <code>http://www.example.com/show.php?id=1</code> ist eine ID-Referenz. Durch es es ist aufzufälligerweise nicht aufzufälligerweise geschützt, auch Daten ausführend für gewöhnlich Benutzerkonten ausgeben.</p> <p><b>Empfehlung</b></p> <p>Implementieren Sie Zugriffskontrollen für ID-Referenzen.</p> <p><b>Code</b></p> <pre> // Beispiel für eine Insecure Direct Object Reference // http://www.example.com/show.php?id=1 </pre> 	A

		<p><b>Risiko</b>          Ein Angreifer kann die komplette Datenkopie durchsuchen und so an sensible Daten gelangen.</p> <p><b>Empfehlung</b>          Die Datenkopie sollte verschlüsselt und geschützt werden, bevor sie in die Applikation verwendet werden.</p>	
(15)	<p><b>Analyse Insecure Direct Object References</b></p>	<p><b>Risk</b>          Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Insecure Direct Object References Schwachstellen untersucht.</p> <p><b>Findings</b>          Die Applikation wird hinsichtlich 'Directory Traversal' überprüft. Dabei gibt es keine Verzeichnisse zu finden, deren Inhalt direkt mit dem Pfad manipuliert werden können.</p> <p>Dabei wurden keine Verzeichnisse gefunden, die gegen Directory Traversal anfällig sind.</p> <p><b>Risk</b>          -</p> <p><b>Empfehlung</b>          -</p>	<p>1</p>

DEMO BERICHT

### 3.3.5 A5: Cross Site Request Forgery (CSRF)

Ein CSRF Angriff zwingt den Browser eines Benutzers, http Request zu senden, die der Benutzer nicht beabsichtigt. Da diese Request durch den Browser abgesetzt werden, werden alle nötigen Daten inklusiv Session Cookies mitgesendet. Dies ermöglicht einem Angreifer Requests zu senden, welche die verwundbare Applikation als legitim erachtet. Ist eine Applikation nicht gegen solche Angriffe geschützt und der Benutzer angemeldet, kann ein Angreifer so zum Beispiel neue Benutzer erstellen, Konfigurationen ändern, etc.

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

**Referenz:**

[http://de.wikipedia.org/wiki/Cross-Site\\_Request\\_Forgery](http://de.wikipedia.org/wiki/Cross-Site_Request_Forgery)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(16)	<b>Analyse Cross Site Request Forgery</b>	<p><b>Ziel</b> Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Cross Site Request Forgery Schwachstellen untersucht.</p> <p><b>Analyse</b> Die Applikation wird mittels formaler spezifizierter Testfälle um die Anfälligkeit gegenüber Cross Site Request Forgery - Angriffen zu prüfen. Die im jeweiligen Testfall definierten Testdaten generiert werden, welche den Testfall erfüllt gefunden werden.</p> <p><b>Werte</b> Sollte die Applikation von Cross Site Request Forgery - Angriffen, kann ein Angreifer im Kontext des angemeldeten Benutzers Aktionen ausführen. Z.B kann ein Angreifer so einen neuen Post-Benutzer anlegen.</p> <p><b>Empfehlung</b> Bei der Weiterentwicklung der Applikation ist zu prüfen, inwiefern grundsätzlich sensitive Requests mit Token absichern.</p>	I

### 3.3.6 A6: Security Misconfiguration



Gute Sicherheit verlangt, dass auch alle Konfigurationen, welche definiert und deployed sind, sicher sind. Dies sind Konfigurationen für die Applikation selber wie auch Frameworks, Application Server, Web Server, Database Server und sonstige Umssysteme und Betriebssysteme. Alle diese Einstellungen sollten definiert, implementiert und gewartet werden, da die Standardkonfigurationen meist keine hohen Sicherheitseinstellungen haben. Dies beinhaltet auch, dass die eingesetzten Applikationen und Bibliotheken immer auf einem aktuellen Stand sind.

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.

#### Referenz:

[https://www.owasp.org/index.php/A10\\_2004\\_Insecure\\_Configuration\\_Management](https://www.owasp.org/index.php/A10_2004_Insecure_Configuration_Management)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(17)	<b>Analyse Security Misconfiguration</b>	<p><b>Ziel</b> Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Security Misconfiguration Schwachstellen untersucht. Dabei wird untersucht, ob die Applikation Fehler einem Angreifer ermöglicht.</p> <p><b>Ansatz</b> Die Applikation gibt alle Parameter aus, welche in der Konfiguration definiert sind. Diese Parameter werden mit den Standardwerten verglichen. Wenn es möglich ist, werden die Parameter mit den Standardwerten verglichen. Wenn es möglich ist, werden die Parameter mit den Standardwerten verglichen.</p> <p><b>Ergebn</b> Die Applikation gibt alle Parameter aus, welche in der Konfiguration definiert sind. Diese Parameter werden mit den Standardwerten verglichen. Wenn es möglich ist, werden die Parameter mit den Standardwerten verglichen.</p>	B

		 <p>Es wurde nur in der Fernbehandlung Applikationspfade und keine weiteren Daten identifiziert und diese analysiert werden.</p> <p><b>Risiko:</b> Eventuelle sensible Angabesätze werden Informationen preisgegeben, welche für einen unbefugten Zugriff sehr nützlich sind.</p> <p><b>Begründung:</b> Die Applikation sollte Fehler anzeigen können und dem Kunden dann eine detaillierte Fehlerbeschreibung anzeigen und keine Informationen preisgeben.</p> <p><b>Maßnahmen:</b> Die im Header angegebene sensible Informationen, die keine</p>	
(18)	<b>Analyse Security Misconfiguration</b>	<p><b>Test:</b> Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Security Misconfiguration Schwachstellen untersucht. Dabei wird untersucht, ob die Applikation Fehler anzeigen kann.</p> <p><b>Ergebnis:</b> Es wurde geprüft, ob im HTTP-Header sensible Informationen preisgegeben werden.</p> <p><b>HTTP-Header:</b></p>	







Die Anwendung greift über die API auf die Daten zu. Die Daten werden über die API abgerufen und die Daten werden in der Anwendung verarbeitet.

id	name	description	status
1	...	...	...
2	...	...	...
3	...	...	...

Die Daten werden über die API abgerufen und die Daten werden in der Anwendung verarbeitet.

**Rolle**

Ein Angestellter kann auf dem Server mit den Rechten des Servers beliebigen Code ausführen.

**Empfehlung**

Die Applikation sollte eine aktuelle Version des Struts Frameworks verwenden.

### 3.3.7 A7: Insecure Cryptographic Storage

Sensible Daten wie Kreditkarten, SSNs oder Authentisierungsangaben werden oft nicht korrekt verschlüsselt oder gehasht abgespeichert. Angreifer können solche ungenügend geschützte Daten möglicherweise stehlen oder modifizieren und weitere kriminelle Handlungen ausüben.

Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.

#### Referenz:

[https://www.owasp.org/index.php/Top\\_10\\_2007-Insecure\\_Cryptographic\\_Storage](https://www.owasp.org/index.php/Top_10_2007-Insecure_Cryptographic_Storage)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(20)	<b>Analyse Insecure Cryptographic Storage</b>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Angewandtheit auf verschlüsselte Cryptographic Storage Schwachstellen untersucht.</p> <p><b>Ergebnis</b></p> <p>Die Daten zum Authentifizieren des Benutzers werden mit einem schwachen Algorithmus verschlüsselt.</p> <p>Ein Angreifer kann mit geeigneten Passwort-Berechnungen und dem generierten Token sein authentisches Passwort verschlüsseln. Einmal ein Token generiert, ist dies eine kurze Zahl mit einem <math>z</math> und einer anderen Zahl generiert werden. Es kann sich herausstellen, dass die erste Zahl die Benutzer ID und die zweite Zahl die aktuelle Sitzung ID darstellen. Ein Angreifer kann durch die Funktion zum Hashing, Nachvollziehbarkeit der Benutzer ID von anderen Benutzern heranzufinden und so alle Token generieren mit dem er deren Passwort ändern kann.</p> <p><b>Risiko</b></p> <p>Ein Angreifer kann die Passwörter von anderen Benutzern ändern und so deren Account übernehmen.</p> <p><b>Empfehlung</b></p> <p>Die Daten von Token sollten starke Zufallsalgorithmen oder starke Hashes verwendet werden.</p>	<div style="border: 1px solid black; padding: 2px; text-align: center; width: 20px; height: 20px; background-color: #800000; color: white; margin: auto;">A</div>

### 3.3.8 A8: Failure to Restrict URL Access

Applikationen prüfen oft die Rechte des Benutzers bevor Verweise und Knöpfe dargestellt werden. Applikationen müssen aber auch zwingend die Berechtigungen des Benutzers prüfen, wenn eine Ressource aufgerufen wird. Tut dies eine Applikation nicht, kann ein Angreifer eine versteckte Ressource trotzdem aufrufen.

Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.

**Referenz:**

[https://www.owasp.org/index.php/Top\\_10\\_2007-Failure\\_to\\_Restrict\\_URL\\_Access](https://www.owasp.org/index.php/Top_10_2007-Failure_to_Restrict_URL_Access)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(21)	<b>Analyse Failure to Restrict URL Access</b>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Ausgabeseiten auf versteckte Ressourcen (URLs, Aufrufe) untersucht.</p> <p><b>Ergebnis</b></p> <p>Es wird untersucht, ob durch gezieltes Aufrufen von URLs Seiten angezeigt werden, die nicht für den aktuellen Benutzer zugänglich sein sollten. Dies wird durch das Erstellen von URLs erreicht, die nicht in der URL-Liste der Applikation aufgeführt sind.</p> <p>Es wurde keine Fragestellung festgestellt, die eine Schwachstelle darstellt.</p> <p><b>Risiko</b></p> <p>...</p> <p><b>Empfehlung</b></p> <p>...</p>	I

### 3.3.9 A9: Insufficient Transport Layer Protection

Um die Vertraulichkeit und Integrität der Daten zu gewährleisten, sollte sensibler Netzwerkverkehr korrekt geschützt sein. Oft werden solche Daten nicht oder nur ungenügend geschützt, dies beinhaltet schwache Algorithmen, die Verwendung von Abgelaufenen Zertifikaten oder generell falsche Implementierungen.

Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.

**Referenz:**

[https://www.owasp.org/index.php/Top\\_10\\_2007-Insecure\\_Communications](https://www.owasp.org/index.php/Top_10_2007-Insecure_Communications)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(22)	<b>Analyse Insufficient Transport Layer Protection</b>	<p><b>Test</b> Die Applikation wird hinsichtlich ihrer Anfälligkeit für möglichen Transport Layer Protection Schwachstellen untersucht.</p> <p><b>Ergebnis</b> Die Applikation verwendet keine Verschlüsselung und überträgt alle Informationen in Klartext.</p> <p><b>Risiko</b> Die Applikation ist anfällig für unbefugten Zugriff auf die übertragenen Daten. Alle übermittelten Informationen können leicht abgelesen und manipuliert werden.</p> <p><b>Empfehlung</b> Sensiblen Informationen sind Passwörter sollten verschlüsselt übertragen werden.</p>	<div style="border: 1px solid black; padding: 2px; display: inline-block; background-color: #800000; color: white; width: 20px; height: 20px; text-align: center; line-height: 20px;">A</div>

### 3.3.10 A10: Unvalidated Redirects and Forwards

Applikationen leiten Benutzer unter bestimmten Umständen auf eine andere Seite weiter. Die Seite, auf welche weitergeleitet werden soll, wird oft aus Benutzerdaten zusammengesetzt, welche nicht korrekt validiert wurden. Werden diese Daten nicht korrekt gefiltert und validiert, kann ein Angreifer seine Opfer auf böswillige Seiten umleiten.

Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

#### Referenz:

[https://www.owasp.org/index.php/Top\\_10\\_2010-A10-Unvalidated\\_Redirects\\_and\\_Forwards](https://www.owasp.org/index.php/Top_10_2010-A10-Unvalidated_Redirects_and_Forwards)

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(23)	<b>Analyse Unvalidated Redirects and Forwards</b>	<p><b>Test</b></p> <p>The Application and its associated files implement all possible Redirects and Forwards. Schwachstellen erkannt.</p> <p><b>Analysen</b></p> <p>Es wurden keine weiteren Schwachstellen gefunden.</p> <p><b>Risiko</b></p> <p>...</p> <p><b>Empfehlung</b></p> <p>...</p>	<b>I</b>

### 3.3.11 Nicht kategorisierte Verwundbarkeiten

In diesem Kapitel werden Schwachstellen aufgeführt, welche nicht unter die OWASP TOP 10 Kategorien fallen.

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(24)	<p>Analysiere Unverschlüsselte File Upload</p>	<p><b>Test</b></p> <p>Die Applikation wird hinsichtlich ihrer Verwundbarkeit gegen Unverschlüsselte Uploads Angriff untersucht. Dabei wird untersucht, ob gezielt Daten auf den Server geschickt werden können.</p> <p><b>Analyse</b></p> <p>Die Funktion zum Upload eines Profildbildes auf der Seite <code>index.php</code> erlaubt es, die Methoden von ausführbaren Dateien.</p> <p>Zuerst findet eine Prüfung des Content Types und der Magic Bytes statt und für das Bild wird ein neuer, zufälliger Name erzeugt.</p> <p>Abhängig von der generierten statischen URL, der Serverkonfiguration sowie im Quellcode werden angepasst, wodurch die Unverschlüsselung möglich ist.</p> <p>Der erlaubte Content type <code>image/jpeg</code> ist in <code>upload.php</code> und <code>upload.php</code> hinterlegt. Zudem werden von PHP Compiler erzeugte Magic Bytes generiert, um die Applikation die Übertragung von <code>image/jpeg</code> zu ermöglichen, was es erlaubt, eine eigene <code>image/jpeg</code> Datei auf den Server zu laden und auszuführen.</p> <p><b>PHP Code, die hochgeladene File</b></p> <pre> &lt;code&gt; &lt;/code&gt; </pre>	<p><b>A</b></p>

```
1 #include <string>
2 #include <vector>
3 #include <algorithm>
4 #include <string_view>
5 #include <string_view_literals>
6 #include <string_view_literals>
7 #include <string_view_literals>
8 #include <string_view_literals>
9 #include <string_view_literals>
10 #include <string_view_literals>
11 #include <string_view_literals>
12 #include <string_view_literals>
13 #include <string_view_literals>
14 #include <string_view_literals>
15 #include <string_view_literals>
16 #include <string_view_literals>
17 #include <string_view_literals>
18 #include <string_view_literals>
19 #include <string_view_literals>
20 #include <string_view_literals>
```

Der Header-Datei wird der Content-Type gefügt



Das Hauptprogramm wird abgepfiffen



Im Quelltext findet man die Adresse des vermeintlichen Bildes

DEA

```

// ...
// ...
// ...
// ...
// ...

```

Beim Aufruf des gefälschten Bildes wird der Code ausgeführt:



### Risiko

Durch das Hochladen von gefälschten Bildern kann die Server-Sicherheit gefährdet werden.

### Empfehlung

Diese Schwachstelle sollte geschlossen werden, indem nur die Daten hochgeladen werden können, welche bereits existiert sind. Eine Prüfung der Datenherkunft ist notwendig. Weiter sollte die Datenherkunft auf nicht direkt aufrufbar werden können.

(25) Web Parameter Tampering

**Risk**  
Es wird versucht, sensitive Informationen auf der Clientseite entfernt werden.  
**Impact**  
Die Seite ist unbrauchbar und kann durch Web Parameter Tampering sein



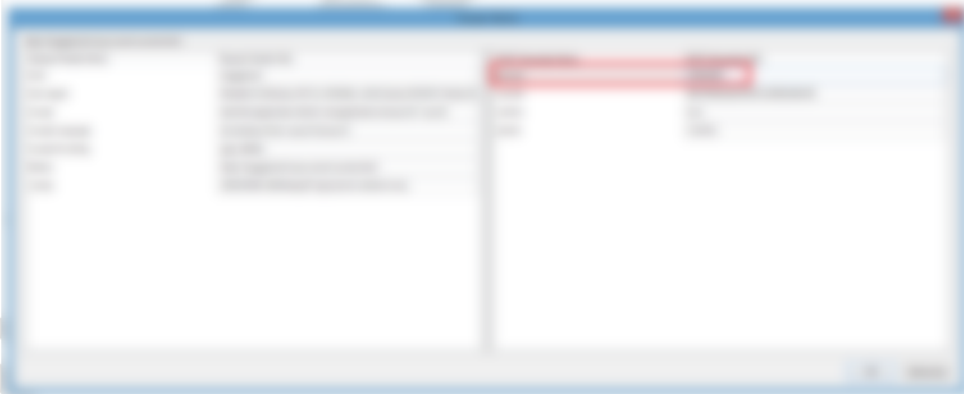




Im Qualitätsbericht der Seite wird nun, dass der Wert des resultierenden PCIDF des Formulars als Hidden Field mitgegeben wird.

Das bedeutet, dass die Werte der PCIDF-Felder im Formular als Hidden Field mitgegeben werden.

Diese Werte sind nun für den Server zugänglich.



Die Verbindung über ein Hidden Field mit dem Server ist ebenfalls gegeben.



**Risiko**

Ein Benutzer kann sich beliebige Benutzernamen aussuchen.

**Empfehlung**

Der Wert eines Benutzernamens sollte mittels einer komplexen Funktion werden.

**Referenzen**

Open Source Security Project (OSSAP), Parameter\_Tampering

Open Source Security Project (OSSAP), Hidden\_Fields

DEMO

## 4 Empfohlene Massnahmen

In Zusammenhang mit den gefundenen Schwachstellen empfehlen wir folgende Massnahmen zu planen und umzusetzen:

- Die Konfiguration des Webanwenders sollte hinsichtlich der Sicherheit überprüft und verbessert werden
- Implementieren einer Input- und Output Validierung um folgende Typen von Schwachstellen zu verhindern:
  - Cross Site Scripting
  - Injection Schwachstellen
  - Unrestricted File Upload
  - Path Traversal vermeiden
- Implementieren einer sicheren Error Handling
- Implementierung einer korrekten Authentisierung und Autorisierung
- Prozess um die Erhaltung der analysierten Themen auch in zukünftigen Releases sicher zu stellen

## 5 Anhang

### 5.1 Tabellenverzeichnis

Tabelle 1 - User-Accounts .....	5
Tabelle 2 - Exklusions-URLs .....	6
Tabelle 3 - Legende .....	7
Tabelle 4 - Übersicht der gefundenen Schwachstellen .....	8

DEMO BERICHT