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

DEMO BERICHT

Inhalt

DEMO BERICHT

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.

DEMO BERICHT

Zusammengefasst sehen wir folgende Empfehlungen

- Die Schwachstelle mit hoher Priorität sollten sofort behoben werden, die aufgelisteten Schwachstellen mit mittlerer Priorität sind in einem Zeitfenster von ca. drei Monaten zu beheben
- Implementieren einer Input- und Output Validierung
- Implementierung eines korrekten Authentisierungs- und Autorisierungsmechanismus
- Implementierung eines sauberen 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 allgemein gültigen Parameter der Überprüfung. Im nachfolgenden Abschnitt „Resultate & Einschätzung Sicherheitsniveau“ sind sämtliche durchgeführten Tests sichtbar, welche zur Entdeckung einer Schwachstelle 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 Massnahmen entspricht. Protect7 kann Sie durch Ihre Erfahrung in der Softwareentwicklung bei der Umsetzung von Sicherungsmassnahmen unterstützen. Der Fokus dieses Penetration Tests liegt in der Identifikation von Schwachstellen.

2.2 Testumfang und Applikations-Kontext

Im Zusammenhang mit einem Web Application Penetration Test sprechen wir von einer oder mehreren Applikationen und einem Kontext, in welchem getestet wird. Das Präfixat ‚Web‘ bezieht sich auf den Standort des Sicherheitsspezialisten, der den Test durchführt. In diesem Fall wurde der Test aus dem Protect7 HackingLab über das Internet ausgeführt. Der Kontext beschreibt den Applikations-Umfang, welcher in der Regel durch das Erteilen einer Authentisierungs-Stufe erreichbar gemacht wird. Ein Kontext kann sich aber auch aus anderen Merkmalen wie bspw. einer bestimmten Source-IP oder einem definierten User-Agent Header bilden. Wesentliches Merkmal eines Applikations-Kontexts ist das Darstellen oder Verfügbarmachen von Ressourcen aufgrund von spezifischen Benutzer-Attributen.

Der Web Application Penetration Test umfasst die Domain buggybook im Kontext eines anonymen Benutzers und im Kontext eines registrierten Benutzers. Ein Administratoraccount wurde zur Verfügung gestellt, der Administratorkontext wurde jedoch auf Kundenwunsch nicht explizit überprüft.

Folgende User-Accounts standen für die Tests zur Verfügung:

Account	Beschreibung
dbuhrer	Administratoraccount
mstehlik	Benutzeraccount
pdelfuri	Alternativer Benutzeraccount

Tabelle 1 - User-Accounts

COPYRIGHT

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.

DEMO BERICHT

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
I	Information
A	Schwachstelle mit hohem Risiko
B	Schwachstelle mit mittlerem Risiko
C	Schwachstelle mit tiefem Risiko

Tabelle 3 - Legende

3.2 Auflistung der gefundenen Schwachstellen

Ref.	Description	Class.[1]
(2)	Der Parameter „id“ auf der Seite /books/view.html ist im authentisierten und nicht authentisierten Kontext auf einen SQL Injection Angriff anfällig.	A
(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.	A
(4)	[illegible]	A
(6)	[illegible]	A
(7)	[illegible]	A
(8)	[illegible]	A

[1] Bewertung siehe Legende 3.1

DEMO BERICHT

(9)	[illegible]	A
(13)	[illegible]	A
(14)	[illegible]	A
(16)	[illegible]	A
(17)	[illegible]	A
(18)	[illegible]	A
(19)	[illegible]	A
(20)	[illegible]	A
(5)	[illegible]	B
(10)	[illegible]	B
(12)	[illegible]	B
(15)	[illegible]	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

DEMO REPORT

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.

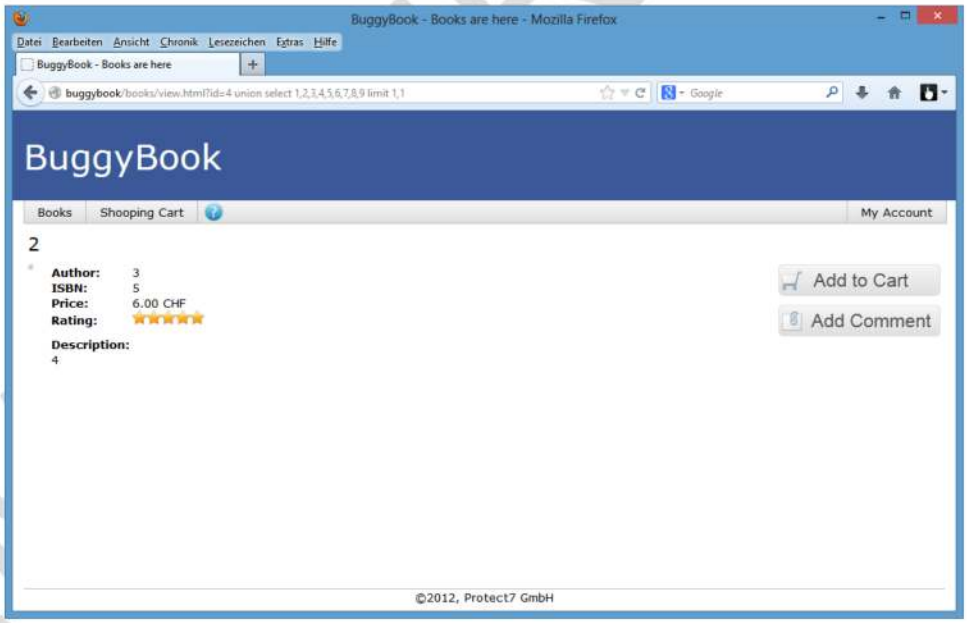
Referenz:

http://de.wikipedia.org/wiki/SQL_Injection
https://www.owasp.org/index.php/SQL_Injection
https://www.owasp.org/index.php/Top_10_2010-A1

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

DEMO BERICHT

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

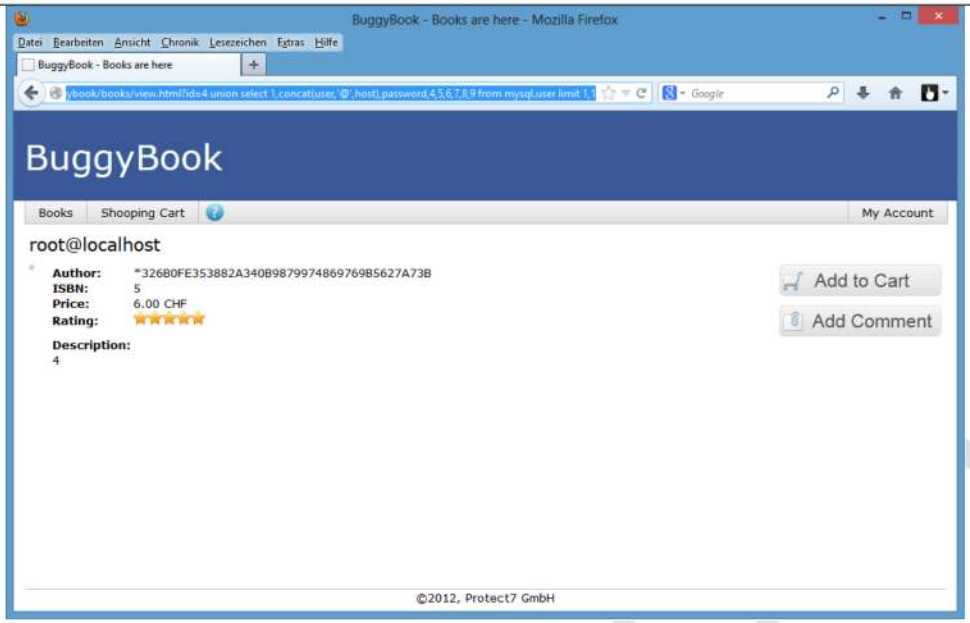


Nun kann Inhalt einer beliebigen sql-Tabelle selektiert und angezeigt werden. Zum Beispiel die Tabelle mysql.user mit dem ersten Benutzer (root).

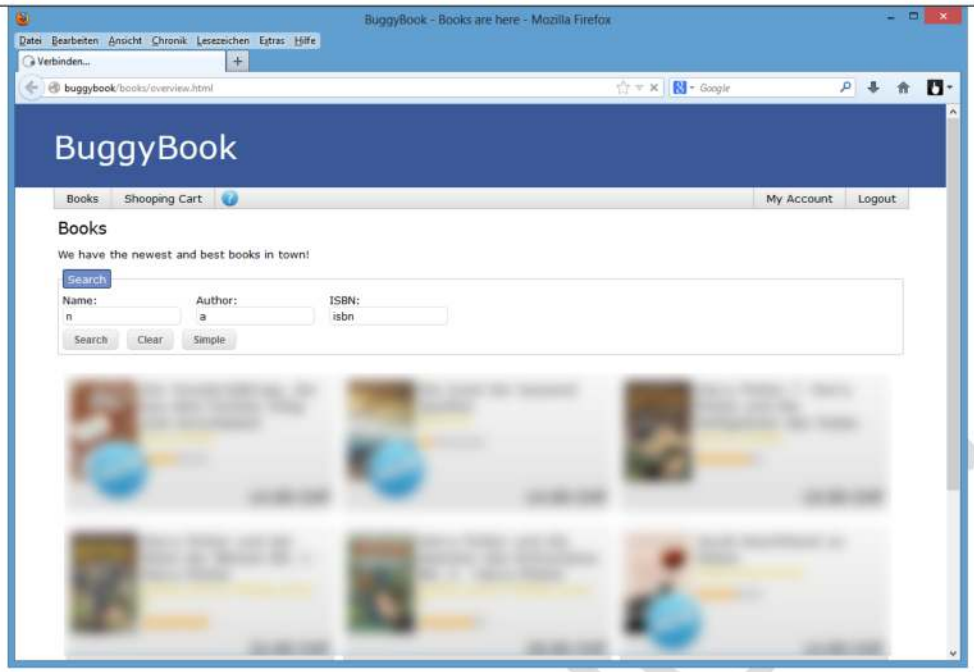
Anfrage:

```
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:

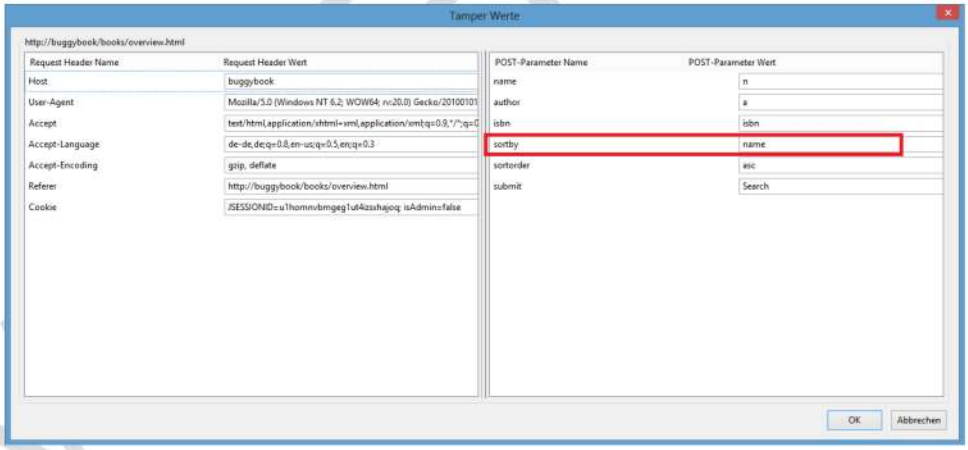
		 **Risiko:** Durch diese Schwachstelle ist es möglich, sensitive Daten auszulesen. **Empfehlung:** Alle Felder müssen saubere Input Validation Richtlinien implementieren, um solche Angriffe zu verhindern. Dazu sollte wenn möglich für alle Datenbankabfragen Prepared Statements eingesetzt werden.	
(3)	**Analyse 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.	A

DEMO BERICHT



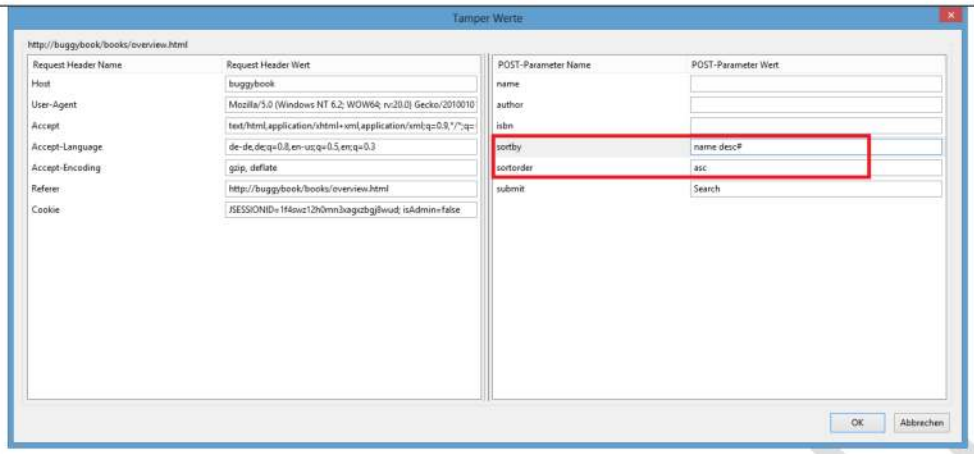
Es ist möglich beliebigen SQL Code einzuschleusen. Diese Funktion steht allen eingeloggten Benutzern zur Verfügung.

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

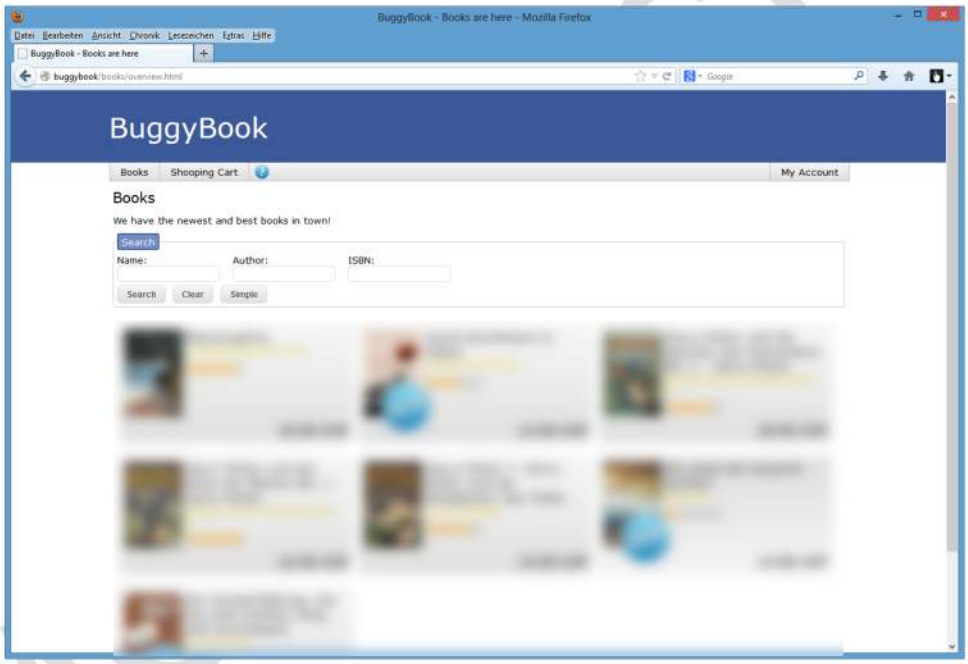


Fügt man dem Parameter “sortby” weitere gültige SQL Kommandos an und kommentiert nachfolgende Kommandos aus, so werden die SQL Kommandos ausgeführt. Im folgenden Beispiel werden die Namen **absteigend** sortiert:

DEMO-EXEMPLAR



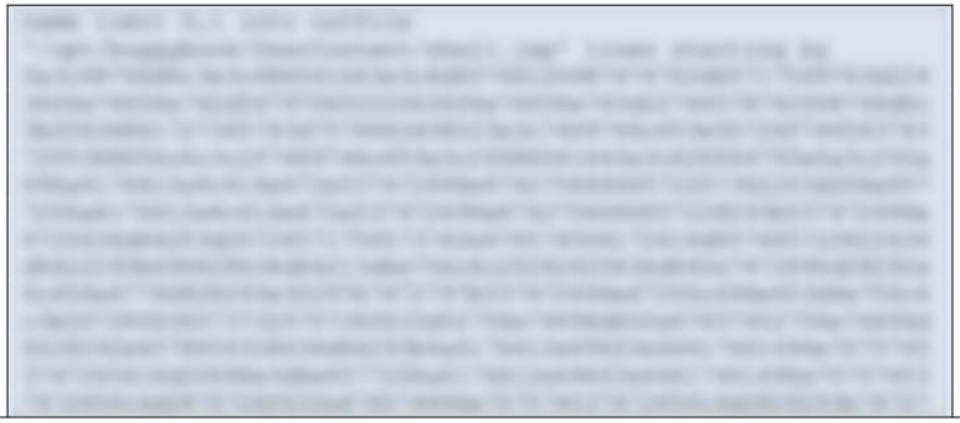
Die Bücher werden wie erwartet in umgekehrter Reihenfolge aufgelistet:



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

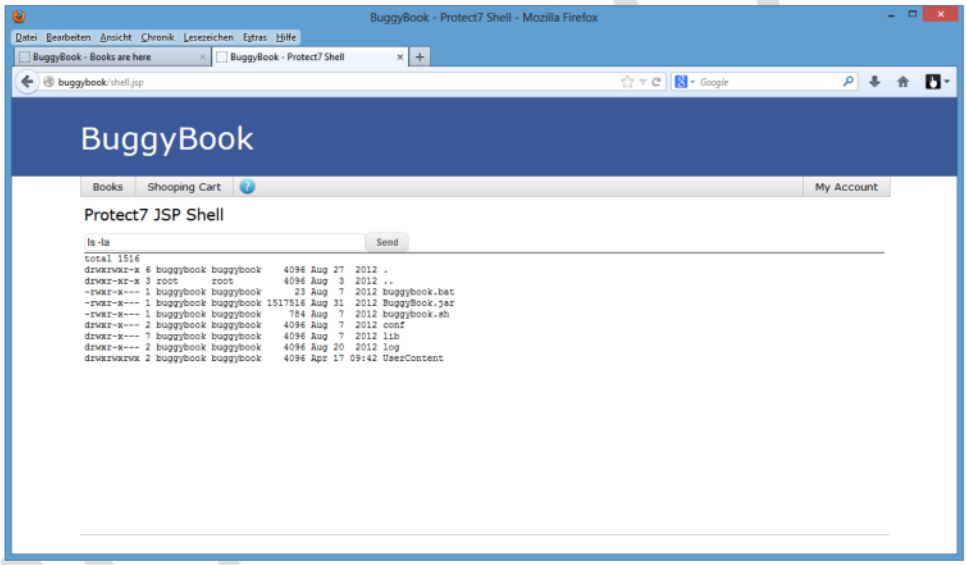
Mit dem Wissen dieser Schwachstelle können wir nun direkt auf dem Server eine Webshell platzieren.

Eingabe in Feld „sortby“:



DEMO

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.

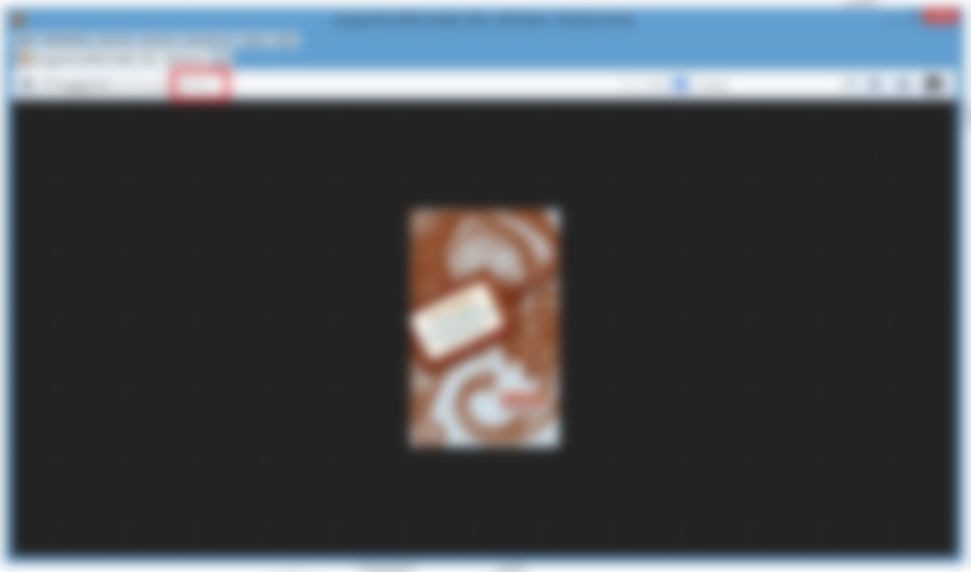
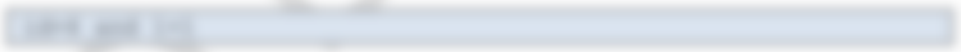
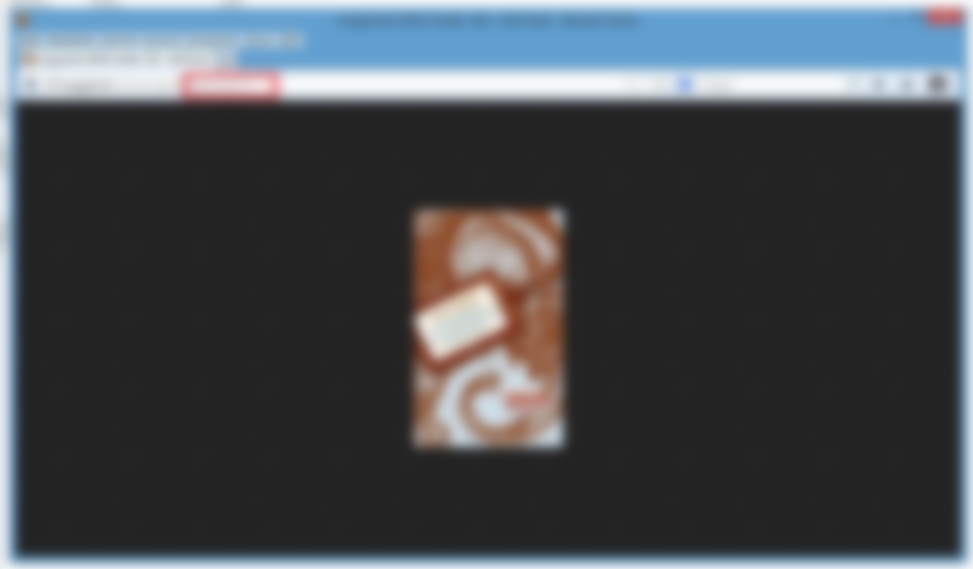
Empfehlung:

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
https://www.owasp.org/index.php/SQL_Injection
https://www.owasp.org/index.php/Top_10_2010-A1
https://www.owasp.org/index.php/Blind_SQL_Injection

DEMO

(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. **Analyse:** Der Parameter „id" auf der Seite /books/image.html ist im authentisierten und nicht authentisierten Kontext auf einen Blind SQL Injection Angriff anfällig. Es ist möglich beliebigen SQL Code einzuschleusen.  Wenn als id ein wahrer SQL Ausdruck eingegeben wird, erscheint das selektierte Bild. Wahrer Ausdruck:  Resultat:  Wenn als id ein falscher Ausdruck eingegeben wird, erscheint die Meldung „Not found". Falscher Ausdruck:	A

DEMO

		[redacted] **Resultat**  **Risiko:** [redacted] **Empfehlung:** [redacted] **Referenz:** [redacted]	
(5)	**Analyse XML Injection**	Test: [redacted] **Analyse:** [redacted] [redacted]	B

DEMO

Resultat

Es kann auch getestet werden, ob z.B. Port 443 offen ist mit der Eingabe:

Resultat

Aus der Meldung „Malformed XML - Connection refused" weiss ein Angreifer ob der Port auf dem entsprechenden System aktiv ist.

Risiko:

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

Empfehlung:

XML Parser hardening implementieren.

Dazu gehört mindestens: Schema Validierung aktivieren, Unterbindung von Ein-

DEMO

		bezug externer Entitäten. Unterbindung der Auflösung externer Entitäten. Einschränkung der Anzahl Knoten. **Referenz:** https://www.owasp.org/index.php/Testing_for_XML_Injection_(OWASP-DV-008)	

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
http://de.wikipedia.org/wiki/Cross-Site_Scripting

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(6)	**Analyse Cross Site Scripting**	[redacted] 	A

DEMO

Risiko:

Grundsätzlich können XSS-Schwachstellen verwendet werden, um eigenen Code auf dem Client-Browser auszuführen.

Empfehlung:

Alle Felder müssen saubere Input- und Output Encoding Richtlinien implementiert werden, um solche Angriffe zu verhindern.

(7)	**Analyse Cross Site Scripting**	**Test:** Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Cross Site Scripting untersucht. Es wird versucht, an verschiedenen Stellen JavaScript Code einzuschleusen, welcher für einen Angriff verwendet werden könnte. Es wird erwartet dass kein JavaScript Code eingeschleust werden kann. **Analyse:** Das Feld *Titel* auf der Seite *myaccountmessageschloss.html* ist im authentisierten Kontext auf einen persistent Cross Site Scripting Angriff anfällig.  Um den XSS Code einzuschleusen, wird auf der Seite	A

DEMO

		myaccountmessageshew.html im Feld Title das Script eingefügt. Um ein rudimentäres Escaping der <Script> Tags zu umgehen, müssen die Tags verschachtelt werden. **Beispieleingabe:** [redacted] Wenn der Empfänger der Nachricht die Nachrichtendetails anzeigt, wird das Script ausgeführt.  **Risiko:** Grundsätzlich können XSS-Schwachstellen verwendet werden, um eigenen Code auf dem Client-Browser auszuführen. **Empfehlung:** Alle Parcbit müssen saubere Input- und Output Encoding Richtlinien implementiert werden, um solche Angriffe zu verhindern.	

DEMO

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:

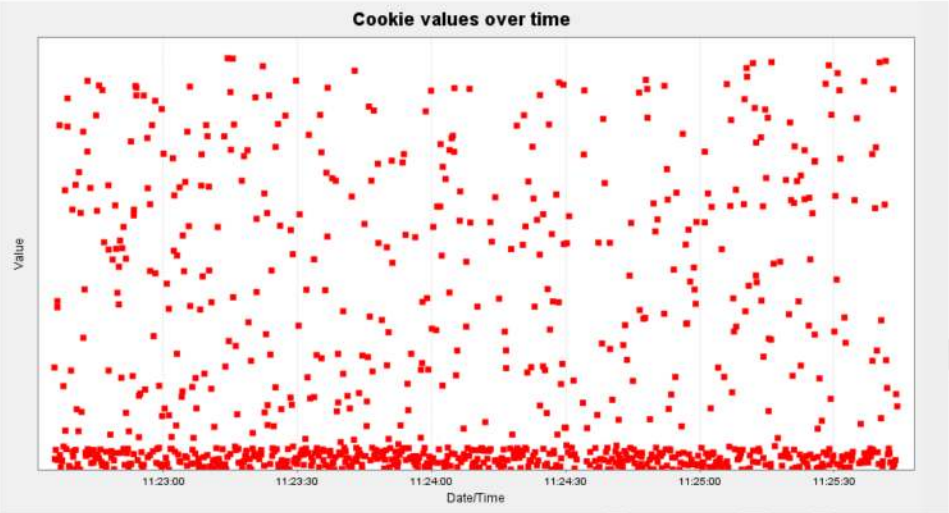
https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(8)	**Analyse Broken Authentication and Session Management**	[illegible]	A
(9)	**Analyse Broken Authentication and Session Management**	[illegible]	A

DEMO

		[redacted]	
(10)	**Analyse Broken Authentication and Session Management**	[redacted]	B
(11)	**Analyse Broken Authentication and Session Management**	**Test:** Die Applikation wird hinsichtlich Ihrer Anfälligkeit auf Broken Authentication und Session Management Schwachstellen untersucht. Die Session ID, welche von der Applikation dem Kunden vergeben wird, wird analysiert. Es wird erwartet, dass dies eine Zufällige, nicht nachvollziehbare Zahlenfolge mit einer entsprechenden Länge ist. **Analyse:** Die Session ID wird im Cookie JSESSIONID gespeichert und besitzt eine aus-	C

DEMO BERICHT

		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**	[illegible]	B

DEMOPERIOD

Risiko:

Ein Angreifer kann selber wählen, mit welcher SessionID ein Opfer arbeitet. Dadurch kann er im Kontext des Opfers agieren. Da die SessionID bei einem Kontowechsel nicht neu generiert wird, kann er auch die Rechte des Opfers nach dem Login wahrnehmen.

Empfehlung:

Das HttpOnly Flag bei den Cookies sollte gesetzt werden. Die SessionIDs sollten bei einem Kontowechsel neu generiert werden. Es sollten keine XSS Schwachstellen vorhanden sein.

(13)	**Analyse Broken Authentication and Session Management**	**Test:** Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Broken Authentication und Session Management Schwachstellen untersucht. **Analyse:** Die Admin-Rechte werden nur durch einen Cookie Value bestimmt. Durch Manipulation des Cookies ist es möglich, Administratorrechte zu erlangen.	A

DEMO

Nach dem Setzen von isAdmin=true kann der Angreifer auf alle Admin-Ressourcen zugreifen.

Risiko:

Ein Angreifer kann alle Rechte des Administrators wahrnehmen.

Empfehlung:

Sensitive Informationen sollten nie clientseitig gespeichert werden. Autorisationen müssen immer serverseitig verwaltet werden.

DEMO

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 Zugriffschutz 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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(14)	**Analyse Insecure Direct Object References**	[redacted] 	A

DEMO COPYRIGHT

		Risiko: Ein Angreifer kann das komplette Dateisystem durchsuchen und so an sensitive Daten gelangen. **Empfehlung:** Die Pfadangaben sollten strikte gefiltert und validiert werden, bevor sie in der Applikation verwendet werden.	
(15)	**Analyse Insecure Direct Object References**	**Test:** Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Insecure Direct Object References Schwachstellen untersucht. **Analyse:** Die Applikation wird hinsichtlich „Directory Traversal" überprüft. Dabei geht es darum, Verzeichnisse zu finden deren Inhalt direkt mit dem Browser angezeigt werden können. Dabei wurden keine Verzeichnisse gefunden, die gegen Directory Traversal anfällig sind. **Risiko:** - **Empfehlung:** -	I

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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(16)	**Analyse Cross Site Request Forgery**	[illegible]	[illegible]

DEMO BERICHT

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 Umsysteme 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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(17)	**Analyse Security Misconfiguration**	[redacted]	B

DEMOBERICHT

So sieht man in der Fehlerbeschreibung Applikationspfade und kann vermuten, dass Jetty und Struts eingesetzt werden.

Risiko:

Einem potentiellen Angreifer werden Informationen preisgegeben, welche für einen erfolgreichen Angriff sehr nützlich sind.

Empfehlung:

Die Applikation sollte Fehler abfangen können und dem Kunden dann eine allgemeine Fehlerseite/Fehlerinformation anzeigen und keine Informationen ausgeben.

Referenz:

http://de.wikipedia.org/wiki/Information_Disclosure

(18)	**Analyse Security Misconfiguration**	**Test:** Die Applikation wird hinsichtlich ihrer Anfälligkeit auf Security Misconfiguration Schwachstellen untersucht. Dabei wird untersucht, ob die Applikation Fehler einem Angreifer preisgibt. **Analyse:** Es wurde geprüft, ob im Http-Header unnötig Informationen preisgegeben werden. Http-Header:	I

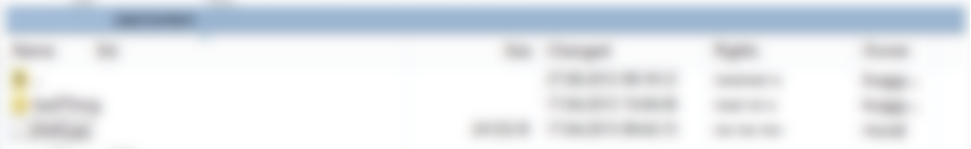
DEMO

		[redacted]	
(19)	**Analyse Security Misconfiguration**	[redacted]	A

DEMO EXEMPLAR

Resultat

Als Resultat erscheint zunächst einmal eine Fehlermeldung. Allerdings wurde der Befehl ausgeführt und das Verzeichnis erstellt.



Dadurch lassen kann durch die Schwachstelle auch ein Programm ausgeführt werden mit den Rechten des Webservers.

Risiko:

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

Empfehlung:

Die Applikation sollte eine aktuelle Version des Struts-Frameworks verwenden.

DEMO

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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(20)	**Analyse Insecure Cryptographic Storage**	[illegible]	A

DEMO COPYRIGHT

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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(21)	**Analyse Failure to Restrict URL Access**	[illegible]	I

DEMO BERICHT

3.3.9 A9: Insufficient Transport Layer Protection

Um die Vertraulichkeit und Integrität der Daten zu gewährleisten, sollte sensitiver 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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(22)	**Analyse Insufficient Transport Layer Protection**	[illegible]	A

DEMO BERICHT

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

Nr.	Fragestellung/Test	Analyse/Empfehlung	Klass.
(23)	**Analyse Unvalidated Redirects and Forwards**	[illegible]	I

DEMO BERICHT

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)	[redacted]	[redacted]	A

DEMO

Vor dem Hochladen wird der Content-Type gelöscht

Das manipulierte Bild wird akzeptiert

Im Quelltext findet man die Adresse des vermeintlichen Bildes

DEMO

		 Beim Aufruf des gelöschten Bildes wird der Code ausgeführt:  **Risiko:** Durch das Hochladen von eigenen Dateien kann der Server kompromittiert werden. **Empfehlung:** Datei-Uploads müssen so gestaltet werden, dass nur die Dateien hochgeladen werden können, welche benötigt werden. Eine Prüfung der Dateiendungen ist notwendig. Weiter sollte das Dateiverzeichnis darf nicht direkt aufgerufen werden können.	
(25)	**Web Parameter Tampering**	**Test:** Es wird geprüft, ob sensitive Informationen auf der Clientseite editiert werden können. **Analyse:** Die Seite myaccountvoucher.html lässt Web Parameter Tampering zu.	A

DEMO

Im Quellcode der Seite sieht man, dass der Wert des Vouchers beim POST des Formulars als Hidden Field mitgesendet wird.

Dieser Wert wird vor dem Absenden manipuliert.

Der veränderte Wert im Hidden Field wird dem Benutzer tatsächlich gutgeschrieben.

DEMO

Risiko:

Ein Benutzer kann sich beliebige Belege gutschreiben.

Empfehlung:

Der Wert eines Vouchers muss mehrfach serverseitig bestimmt werden.

Referenzen:

https://www.owasp.org/index.php/WAS_Parameter_Tampering

https://www.owasp.org/index.php/Category:WASS_Hidden_Fields

DEMO PROTOCOL

4 Empfohlene Massnahmen

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

- Die Konfiguration des Webservers 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 eines sauberen Error-Handling
- Implementierung einer korrekten Authentisierung und Autorisierung
- Prozess um die Einhaltung der erwähnten Themen auch in zukünftigen Releases sicher zu stellen

DEMO BERICHT

5 Anhang

5.1 Tabellenverzeichnis

DEMO BERICHT