

FACHHOCHSCHULE WEDEL

SEMINARARBEIT

Wintersemester 2016 /17

Seminar: IT Sicherheit

Thema:

Web Application Testing

Eingereicht von:	Janos Bastian Frank E-Mail: janos.frank.jf@gmail.com
Erarbeitet im:	7. Semester
Abgegeben am:	20.12.2016
Betreuer:	Prof. Dr. Gerd Beuster Fachhochschule Wedel Feldstraße 143 22880 Wedel

Inhaltsverzeichnis

1. Einführung	1
1.1 Problemstellung	1
1.2. Gang der Untersuchung	1
1.3. Was definiert eine Web-Applikation?	2
2. Burp-Suite	3
2.1. Kommunikation mit der Web-Applikation	3
2.2. Burp-Suite Proxy	3
2.3. Arbeiten mit Burp-Suite	4
3. SQL-Injection	5
3.1. Definition des Problems	5
3.2. Funktionsweise SQL-Injection	5
3.3. Arten und Beispiele SQL-Injection	6
4. XPath Injection	7
4.1. Definition des Problems	7
4.2. Funktionsweise XPath Injection	7
5. Command Execution	9
5.1. Definition des Problems	9
5.2. Funktionsweise Command Execution	9
6. Cross-Site Scripting	11
6.1. Definition des Problems	11
6.2. Reflected Cross Site Scripting	11
6.3. Stored Cross Site Scripting	12
6.4. Beispiel Codeeingaben	12
7. Web Application Scanning	13
7.1. Beschreibung Vorgehen der Tools	13
7.2. Webanwendung scannen mit Openvas	13
8. Exkurs Konfiguration Webserver	16
8.1. Definition des Problems	16
8.2. Apache Härtungsguide als Gegenmaßnahme	16
9. Abschlussbewertung	18
9.1. Fazit	18
9.2. Ausblick	18
Literaturverzeichnis und Quellenverzeichnis	19

Abbildungsverzeichnis

Abbildung 1: Kommunikation User / Server	2
Abbildung 2: Proxy Einstellung	3
Abbildung 3: Request	4
Abbildung 4: Ping Device	9
Abbildung 5: Ping Device Umleitung (ohne Ausgabe)	9
Abbildung 6: Beispiel XSS	12
Abbildung 7: Ergebnis Scan	15

Tabellenverzeichnis

Tabelle 1: Checkliste Webserversecurity	16
---	----

1. Einführung

1.1 Problemstellung

Glaubt man statistischen Erhebungen, werden aus dem Internet verfügbare IT-Systeme alle 40 Minuten durch Cyber-Kriminelle gescannt. Schwachstellen werden ausgenutzt, um IT-Systeme zu kompromittieren. Speziell Webanwendungen sind ein beliebtes Angriffsziel. Die Möglichkeit wirtschaftlichen Schaden anzurichten und verwertbare Information zu entwenden ist hier vielfältig. Da Usability bei den meisten Webanwendungen im Vordergrund steht, ist es für die betreuenden Fachkräfte meist sehr schwer ihre Systeme im Zusammenspiel mit den Benutzern so zu schützen, dass ein Angriff extrem erschwert wird. Hinzu kommt, dass Webanwendungen extrem komplex sein können, sowohl was die Architektur, also auch Funktionsweise angeht. Vor allem die Architektur, Zusammenspiel aus Webserver Betriebssystem und Komponenten der Webanwendung wie JavaScript, Flash und Datenbanken, eröffnen einem Hacker eine Vielzahl von Angriffsvektoren. Im Folgenden werden einige der wichtigsten Angriffstechniken und die dafür entwickelten Tools vorgestellt.

1.2. Gang der Untersuchung

Diese Seminararbeit soll einen Überblick über bestehende Angriffstechniken gegen Webanwendungen geben. Als Leitfaden wurde Kapitel 14, Web Application Testing, aus dem Buch „Penetration Testing“ von Georgia Weidman vorgegeben. Die einzelnen Techniken werden als erstes kurz definiert und im Anschluss im Detail erläutert. Es werden insgesamt fünf Techniken beschrieben. Zusätzlich werden zwei Tools erläutert, die beim Aufzeigen und Anwenden der Angriffstechniken bevorzugt benutzt werden. Der volle Funktionsumfang wird dabei nicht beleuchtet werden. Insgesamt soll anhand des Leitfadens ein Einblick gegeben werden, wie Webanwendungen angegriffen werden können und wie die Techniken im Detail sich typische Schwächen dieser Anwendungen zu Nutze machen können.

1.3. Was definiert eine Web-Applikation?

Eine Web-Applikation ist ein Programm, welches nach dem Client-Server-Modell arbeitet. Dabei wird die Datenverarbeitung und Datenauswertung nicht auf dem lokalen Client des Anwenders vorgenommen, sondern zentral auf einem Webserver. Dieser kommuniziert mit dem Client über das HTTP-Protokoll. Ein User sendet dabei

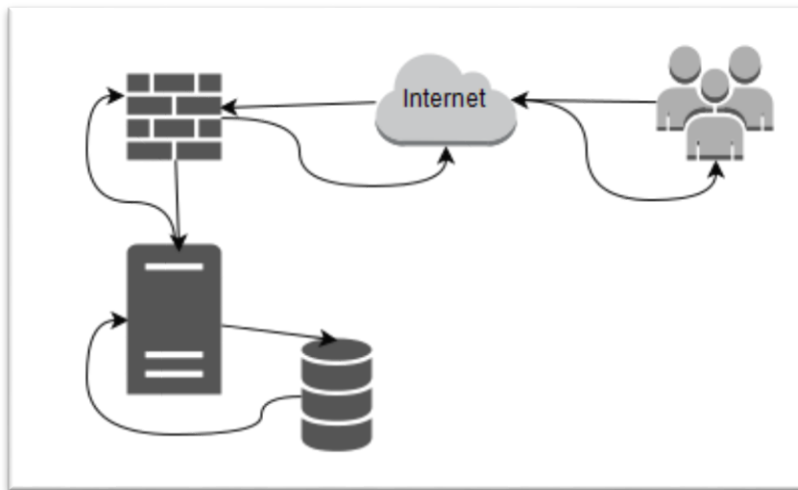


Abbildung 1: Kommunikation User / Server

einen Request an den Server „GET http/1.1
→http://Demo-page.de“ zum Beispiel. Dieser prüft ob der Request korrekt ist und sendet eine Response

„http/1.1 200 OK“

und die entsprechende Antwort auf den Request.

Es gibt viele verschiedene Arten und Anwendungsgebiete von Web-Applikation. Im Grund sind der Aufbau und der Zugriff aber immer gleich gestaltet. Dabei sind in 90% der Fälle immer mindestens die in Abbildung 1 dargestellten Komponenten beteiligt. Ein oder mehrere User, das Internet als Übertragungsmedium, ein Router/Firewall, der Webserver und irgendeine Form von Datenspeicher / Datenbank.

Aufgrund dieser Diversität ist es entsprechend schwer eine Web-Applikation effektiv zu schützen. Spätestens der User ist eine, zumindest teilweise, nicht kontrollierbare Komponente, da eine gewisse Usability erhalten bleiben muss. So ist es zum Beispiel nicht vorstellbar mehr als zwei Formen der Authentifizierung vom Nutzer zu verlangen. Die wenigsten Anwendungen verfügen überhaupt über eine zweite Authentifizierungsmethode und so bleibt beim Betreiben einer Web-Applikation immer das Restrisiko „Nutzer“. Neben dem User, bieten auch die anderen Komponenten genügend Angriffsvektoren, um an sensible Daten zu kommen oder das System komplett zu übernehmen.

2. Burp-Suite

2.1. Kommunikation mit der Web-Applikation

Einer der Angriffsvektoren auf eine Web-Applikation, ist die Kommunikation vom Client zum Server und umgekehrt. Diese, mit den Pfeilen auf Abbildung 1 dargestellt, ist an jedem Ausgangspunkt verletzbar. Für die jetzige Betrachtung gehen wir aber von der Manipulation der Kommunikation direkt an die Web-Applikation aus. Diese Verbindung, die normalerweise im Hintergrund vom Browser abgewickelt wird und von der der User bis auf die angezeigte Seite eigentlich nichts mitbekommt, ist in Form von Requestmanipulation beeinflussbar.

2.2. Burp-Suite Proxy

Die Burp-Suite in der Free Edition ist ein kostenloses Java-Tool und bietet verschiedene Möglichkeiten, um die Kommunikation vom Browser zum Server zu untersuchen. Durch das Vorschalten eines Proxyservers, der von Burp-Suite mitgeliefert wird, werden alle Anfragen über Diese umgeleitet und können so vor dem Senden an den Server um konfiguriert werden. Dafür ist nichts weiter zu tun, als in den Browser-netzwerkeinstellungen den Proxy manuell einzutragen. Dieser läuft entweder auf dem lokalen System 127.0.0.1 auf Port 8080 oder auf einem entsprechenden externen Client / Server. Im externen Fall ist die private IP des Client / Server im Netz oder die öffentliche IP einzutragen. Zusätzlich muss der Haken bei „Für alle Protokolle diesen Proxy-Server verwenden“ gesetzt sein. Nachdem diese erledigt ist, ist Burp-Suite in der Lage Eingaben im Browser zu verarbeiten.



Abbildung 2: Proxy Einstellung

2.3. Arbeiten mit Burp-Suite

Im Tool findet man den Reiter „Proxy“ in dem sich die vier Buttons „Forward“ (Anfragen weitergeben), „Drop“ (Anfrage verwerfen), „Intercept is on / off“ (Anfragen abfangen / weiterleiten), „Action“ (Aktionswerkzeuge für abgefangene Anfragen) befinden.

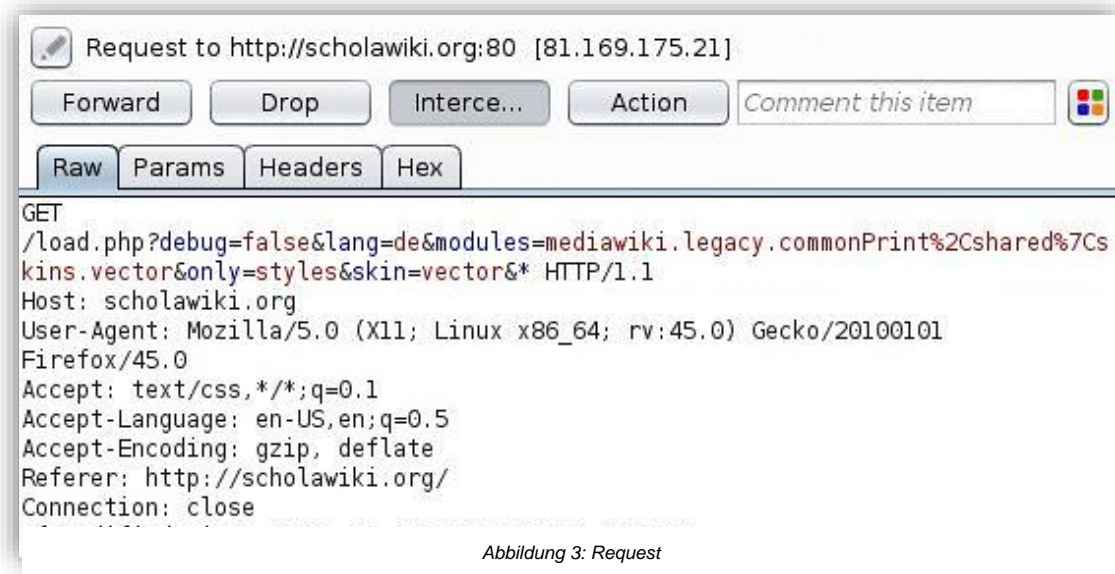


Abbildung 3 zeigt wie ein Request von Burp-Suite angezeigt wird. Dabei werden viele Informationen bereitgestellt, die für einen späteren Angriff genutzt werden können wie IP Adresse des Servers, verwendeter Port für den Request und Aufbau und Schema des GET-Requests. Außerdem kann man neben der „Raw“ Ansicht auch die „Params“ Ansicht wählen, in der man eine deutlich bessere Aufbereitung der mitgesendeten Parameter bekommt.

Vor dem Weiterleiten können am Browser-Request Änderung vorgenommen werden, wie z.B. das Abändern der mitgeschickten Login Informationen. Aber auch andere Parameter lassen sich leicht verändern, sodass man der Web-Application Befehle schicken kann, die so über den Browser nicht möglich wären. Die Möglichkeit diese Befehle mitzuschicken oder Parameter zu verändern, sodass die Anwendung in „Panic“ gerät, ist Inhalt von Kapitel 3. SQL-Injection.

3. SQL-Injection

3.1. Definition des Problems

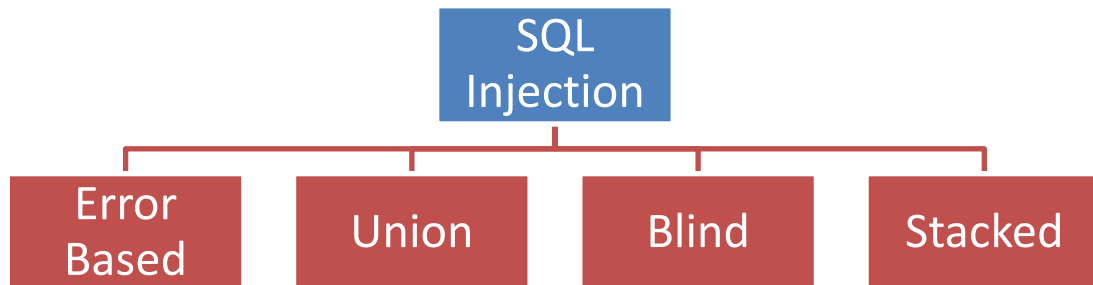
Zur Speicherung von Daten greifen Web-Applications in der Regel auf Datenbanken zu. Die Anwendung nutzt dann über eine definierte Schnittstelle (meistens SQL) die, in der Datenbank bereitgestellten Ressourcen, in vordefinierten Abfragen. Kann ein Angreifer diese Abfrage manipulieren z.B durch die Verwendung von Burp-Suite, so kann dieses Vorgehen die komplette eingesetzte Datenbank kompromittieren.

3.2. Funktionsweise SQL-Injection

Die meisten Webanwendungen verfügen über irgendeine Form von Eingabemaske, z.B. eine Suchfunktion oder Login Möglichkeit. Diese Funktionen benötigen im Hintergrund Durchgriff auf die Datenbank um zu funktionieren. Diese Tatsache kombiniert mit dem Wissen um die Metazeichen von SQL, ermöglicht eine SQL-Abfrage in der Datenbank mit den eingegebenen Zeichen auszulösen. Solche Zeichen sind beispielsweise \, , ' oder ; Meistens findet man diese Möglichkeit der Zeichenmitgabe in CGI-Skripten oder in Anwendungen, die Daten wie Webseiteninhalte sowie Emailadressen in SQL-Datenbanken eintragen. Diese Programme neigen zu einer Inkorrektheit in der Maskierung, was dem Angreifer die Möglichkeit gibt, zusätzliche Abfragen einzuschleusen um Daten zu verändern oder auszugeben. Ein Beispiel:

Szenario ist, dass eine Benutzereingabe nur mangelhaft geschützt wurde. Normalerweise ist das SQL Statement „**SELECT** * **FROM** users **WHERE** name = `` + `userName` + ``; so aus. Dieser Datensatz würde den angegebenen Benutzernamen auslesen. Nun kann man die Variable `userName` abändern, sodass viel mehr Schaden angerichtet wird zum Beispiel durch ersetzen mit 'or ,1='1'. Falls dieser Ersatz zur Authentifizierung verwendet wird, gibt die Datenbank einen Benutzernamen aus, denn 1 = 1 ist immer wahr und lässt den Befehl auch bei falscher Eingabe zu. Diese Art von SQL Injection Technik nennt man Blind-Boolean Based. Neben der Möglichkeit direkt an den Port zu senden, kann auch via URL gearbeitet werden <http://Beispielseite.de/news.php?id=2 and 1=1>. Neben Blind SQL Injection gibt es andere Techniken auf SQL Basis.

3.3. Arten und Beispiele SQL-Injection



Error Based SQL Injection ist die einfachste Form der SQL-Injection, bei der versucht wird, bewusst Fehlermeldungen zu produzieren, um hieraus Hinweise auf eventuelle Schwachstellen gewinnen zu können. Ein Beispiel hierfür sind Anhänge an die URL wie Hochkommata oder Anführungszeichen [www.Beispielseite.de/index.asp?id=77'](http://www.Beispielseite.de/index.asp?id=77) welche bei Durchgriff auf die Datenbank Fehler mit nützlichem Informationsgehalt auswerfen können.

Union SQL Injection setzt voraus, dass der UNION-Operator in Abfragen erlaubt ist. So können neben der normalen Abfrage durch die Webanwendung noch zusätzliche Tabellen abgerufen werden. Als Beispiel: „SELECT Name, Phone, Address FROM Users WHERE Id=\$id“ als der Query der Anwendung, „SELECT Name, Phone, Address FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber,1,1 FROM CreditCardTable“

Stacked SQL Injection sind im Grund wie UNION, nur dass es hier nicht benutzt wird sondern das Semikolon am Ende der normalen SQL- Anfrage. So können theoretisch beliebig viele Statements mitgeschickt werden. Ein möglicher Angriff könnte so aussehen: „SELECT * FROM products WHERE productid=1; DELETE FROM products“ mit der Folge, dass alle Produkte gelöscht wären.

4. XPath Injection

4.1. Definition des Problems

XPath ist eine Sprache, um auf Daten innerhalb von XML-Strukturen zuzugreifen. XPath wird dabei vergleichbar zu SQL eingesetzt, um Daten innerhalb der Struktur zu lesen oder zu schreiben. Erfolgreiche XPath-Angriffe können einem Angreifer unberechtigter Weise Zugriff auf ein Portal gewähren.

4.2. Funktionsweise XPath Injection

Ähnlich wie die SQL-Injection funktioniert auch die XPath Injection. Die Webanwendung erzeugt hierbei ebenfalls eine Abfrage in Form eines XML-Dokuments oder eines XQueries. In diese Abfrage-Operation können Manipulationen vorgenommen werden um Daten zu bekommen oder Authentifizierungsprozesse zu umgehen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<test_database>
<test_user>
<username>test</username>
<password>testing123</password>
<account>Administrator</account>
</test_user>
<test_user>
<username>Lokesh</username>
<password>test1234</password>
<account>Subscriber</account>
</test_user>
<test_user>
<username>Destructive</username>
<password>mind123</password>
<account>Subscriber</account>
</test_user>
</test_database>
```

Dieser Code zeigt das Grundformat wie in einer XML Datei sensible Informationen gespeichert sind. Um die Informationen abzurufen muss folgender XQuery geschrieben werden:

```
string (// test_user [username / text () = 'test' und Passwort / text () = 'testing123'] /
account / text ())
```

Diese Anfrage wird auch erzeugt wenn zum Beispiel ein Login Prozess durch die Web Anwendung vorgenommen wird. Sofern die Anwendung nicht entsprechend geschützt ist, kann man an dieser Stellen als Angreifer versuchen eigenen Code einzuschleusen. Das könnte so aussehen:

```
string(//test_user[username/text()=" or '1' = '1 and password/text()=" or '1' = '1']/account/text())
```

Ähnlich wie bei der SQL Injection wird auch hier ein „or“ benutzt um den Booleanwert True zu erzeugen und so die Abfrage als gültig in der Datenbank zur Ausführung zu bringen. So kann ohne entsprechende Berechtigung Zugang zu geschützten Inhalten hergestellt werden. Andere Beispiele die anstelle von „or '1' = '1“ genutzt werden können sind:

1. OR user() and '1' = '1
2. OR count(//*) AND '1' = '1
3. OR lower-case('A') AND '1' = '1

5. Command Execution

5.1. Definition des Problems

Wie im Kapitel SQL-Injection und XPath-Injection beschrieben bieten Eingabefelder, die eine Interaktion mit dem Backend erzeugen, die Möglichkeit an Daten zu kommen oder Schutzprozesse zu umgehen. Über diese Eingabemasken kann bei entsprechender Schwachstelle, auch eine Command Execution stattfinden also die Ausführung von Befehlen oder Skripten auf der Commandline des Webserver. Diese Befehle werden einfach entsprechend codiert über die Eingabe an den Server weitergegeben.

5.2. Funktionsweise Command Execution

Wenn keine Filterung der eingegeben Werte durch die Webanwendung vorgenommen wird und das Eingabefeld oder ein Link, direkten Zugriff auf den Server haben, ist es leicht Befehle an den Webserver zu senden. Als Beispiel eine Webanwendung die eine Eingabe zum pingen von IP Adressen bereitstellt.



Ping a device

Enter an IP address:

Abbildung 4: Ping Device

Die Anwendung erwartet eine IP Adresse, die weitergegeben wird an den Webserver. Mit dem Sonderzeichen | (Pipe) ist es möglich, Ein- und Ausgabeumleitungen weiter zu geben. Gibt man nun 1 | und den gewünschten Befehl ein, wird IP Adressabfrage umgebogen auf den gewünschten Befehl. Als Beispiel:



Ping a device

Enter an IP address:

Routenverfolgung zu www.fh-wedel.de [213.39.232.206] über maximal 30 Abschnitte:

Abbildung 5: Ping Device Umleitung (ohne Ausgabe)

Diese harmlose Eingabe wurde vom Webserver entsprechend ausgeführt und das Ergebnisse anstelle des IP Pings ausgegeben.

Abgesehen von dieser direkten Möglichkeit Befehle an den Server zu geben kann man auch modifizierte URLs an die Anwendung senden. Eine URL die z.B Daten abrufen soll bzw. Daten vom Server bereitstellt sieht Beispielweise so aus:

```
http://beispielseite/cgi-bin/userData.pl?doc=geheim.txt
```

Diese Abfrage kann jetzt modifiziert werden, sodass ein ähnlicher Effekt wie bei der Eingabe über die Maske entsteht:

```
http://beispielseite/cgi-bin/userData.pl?doc=/bin/ls|
```

Hier würde der Befehl `/bin/ls` ausgeführt und das Ergebnis im Browser angezeigt werden.

6. Cross-Site Scripting

6.1. Definition des Problems

Generell dient diese Art von Angriff dazu, eigenen Code in eine andere Webseite zu injizieren, sodass die betroffene Seite beim nächsten Ausliefern nicht nur den gewünschten, sondern auch den injizierten Code an den Client zurück liefert. Dieser böartige Code wird dann mit den Rechten der Seite ausgeführt, in die er injiziert wurde. Man unterscheidet bei Cross Scripting (auch als XSS bezeichnet) grundsätzlich zwischen zwei verschiedenen Varianten.

6.2. Reflected Cross Site Scripting

In diesem Fall, liefert der Benutzer den injizierten Code selber an den Server, indem ihm ein böartiger Link zugesendet wird, den das Opfer anklickt. Dabei wird der im Link vorhandene modifizierte Code an den Server übertragen, und ein für XSS anfälliger Server liefert dann diesen modifizierte Code in die Seite integriert zurück an den Client aus, der diesen dann im Rahmen und mit den Rechten der geladenen Seite ausführt. Durch die Nutzung von iFrames, JavaScript (DOM) oder Flash können so mit wenig injiziertem Code auch größere Mengen Code nachgeladen werden, um z.B. Cookies oder den Session Key zu stehlen, oder aber via Redirection oder iFrames Schwachstellen in dem Browser oder seinen Plug-Ins auszunutzen und das Client-System zu kompromittieren. Da der injizierte Code immer mit den Rechten der geladenen Seite ausgeführt wird, kann auf diese Weise schnell ein großes Schadenspotential entstehen.

Wird der injizierte Code durch Codierung verschleiert, und der Link einem Opfer zugesendet, das diesem Link folgt, so wird bei XSS-Verwundbarkeit des Servers dieser Code an den Client zurückgesendet („reflected“) und von diesem ausgeführt, anstatt dass der Username in der Seite angezeigt wird.

6.3. Stored Cross Site Scripting

Stored XSS funktioniert ähnlich wie Reflected XSS, mit dem Unterschied, dass nicht ein Opfer den Code einmalig über ein HTTP GET an den Server überträgt, sondern ein Angreifer seinen Code in eine Webseite einbindet, so dass dieser Code an jeden Client, der die Seite aufruft, zurück geliefert wird. Dadurch ist dieser XSS-Angriff deutlich effektiver und weitreichender als Reflected XSS.

Beim Stored XSS werden User-Eingaben nicht korrekt gefiltert und automatisch in eine Seite eingebunden – wie z.B. in einem Gästebuch oder einem Diskussionsforum: User können eigenen Content hochladen, der beim späteren Aufruf als Teil der Seite vom Server an alle Clients ausgeliefert wird. Ist dieser Inhalt bössartig und filtert der Server User-Content nicht ausreichend, so wird der Code beim nächsten Aufruf im Rahmen und mit den Rechten der Seite auf den Clients ausgeführt.

6.4. Beispiel Codeeingaben

Die Codeeingaben sind in Javascript geschrieben und können so direkt in die Seite eingebunden werden.

Beispiel Reflected XSS:

```
<script language="javascript">alert("Alarm :D")</script>
```



Abbildung 6:Beispiel XSS

Beispiel Stored XSS:

```
<script language="javascript">alert("Alarm :D")</script>
```

Auszug Seitenquelltext: <div id="guestbook_comments">Name: TEst
Message:
<script language="javascript">alert("Alarm :D")</s
</div>

7. Web Application Scanning

7.1. Beschreibung Vorgehen der Tools

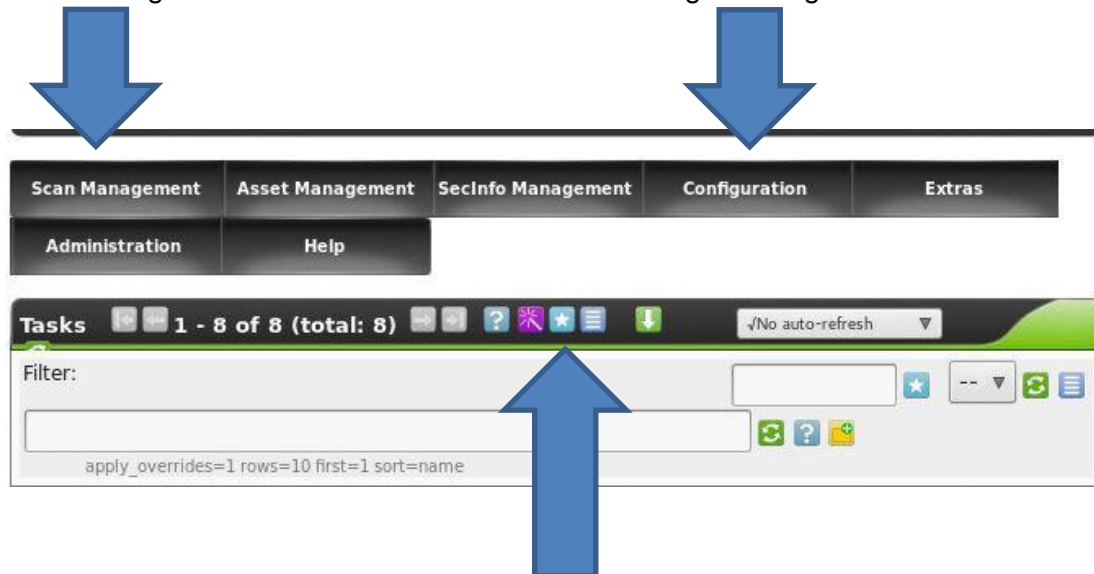
Wie schon beschrieben gibt es eine ganze Reihe an Angriffsvektoren gegen eine Web-Application. Diese Schwachstellen zu kennen ist für Angreifer sowie Betreiber gleichermaßen wichtig. Für diese Erkennung von Konfigurationsfehlern, fehlender Zugriffsbeschränkung z.B. durch eine Firewall und Schwachstellen in der Software selber, gibt es eine ganze Reihe an Tools die die Anwendung gegen einen vorher definierten Rahmen auditieren. Das Vorgehen ist dabei im Grunde immer das Gleiche. Der Benutzer gibt ein Ziel entweder definiert durch die IP oder die URL vor. Das Tool prüft als ersten Schritt, ob das Ziel erreichbar ist über den Standard Port 80 oder 443. Im Anschluss werden je nach Vorgabe auch andere Ports bis hin zu allen 65535 möglichen untersucht. Offene Ports werden gelistet und meistens in einem Report zusammen mit ihrer Funktion ausgegeben. Im zweiten Schritt werden die hinter den offenen Ports befindlichen Services erfasst und anhand einer Datenbank auf bekannte Schwachstellen geprüft. Dafür wird meist der gefundene Versionsstand genommen. Als dritten Schritt versucht das Tool aktiv die Schwachstellen auszunutzen um zu prüfen ob nicht andere Sicherheitsmaßnahmen ergriffen wurden.

7.2. Webanwendung scannen mit Openvas

Openvas ist ein Open Source Schwachstellen-Scanner und –Manager der Firma Greenbone. Zusammen mit der Datenbank des BSI und DFN CERT, bildet Openvas eines der fortschrittlichsten Frameworks für Schwachstellenmanagement ab. Die Anwendung ist dabei unter Kali Linux leicht zu installieren. Mit dem Befehl `apt-get install openvas` bekommt man das komplette Framework und kann nach abgeschlossener Installation sofort loslegen. Dafür mit dem Befehl `openvasmd --user=Wunschuser-name --new-password=Wunschwort` einfach einen User anlegen, danach ist der Login unter <https://127.0.0.1:9392/> möglich.







Tasks anlegen

Targets anlegen



Neue Inhalte erstellen

Unter dem Reiter „Configuration“ befindet sich die Kategorie „Targets“ in dem man die Ziele anlegt. Mit dem Stern „New Target“ kommt man auch die Eingabemaske. Hier sind nur die Felder „Name“ und „Hosts“ auszufüllen. Anschließend findet man unter dem Reiter „Scan Management“ die Kategorie „Tasks“ in der man den Scan mit dem Ziel und den Prüfroutinen verknüpft. Dafür wieder auf den Stern und einen Name sowie das Ziel auswählen. Anschließend muss eine „Scan Config“ ausgewählt werden. Dabei hat man die Auswahl zwischen sieben verschiedenen Konfigurationen. Hier sollte unter http://docs.greenbone.net/GSM-Manual/gos-3.1/de/scan_configuration.html nachgelesen werden, welche geeignet ist. Für den Anfang kann man mit „Full and Fast“ aber nichts verkehrt machen, da diese Konfiguration nur diejenigen Prüfroutinen berücksichtigt, die keinen Schaden auf dem Zielsystem anrichten können. Nachdem die Task fertiggestellt ist, muss man nur noch das „Play“ Zeichen drücken und der Scan beginnt. Ergebnisse können live angezeigt werden oder nach Beendigung in einem Report zur Verfügung gestellt werden. Ist eine Schwachstelle erkannt worden, wird sie ausführlich beschrieben, sodass ein Angreifer sie ausnutzen könnte und der Betreiber weiß wo und wie er sie schließen muss.

| Vulnerability |  | Severity  | QoD | Host | Location | Actions |
|--|---|--|-----|--|----------|---|
| MS15-034
HTTP.sys Remote
Code Execution
Vulnerability
(remote check) |  | 10.0 (High) | 95% |  | 8000/tcp |  |
| Summary
This host is missing an important security update according to Microsoft Bulletin MS15-034. | | | | | | |
| Vulnerability Detection Result
Vulnerability was detected according to the Vulnerability Detection Method. | | | | | | |
| Impact
Successful exploitation will allow remote attackers to run arbitrary code in the context of the current user and to perform actions in the security context of the current user. | | | | | | |
| Solution
Solution type:  VendorFix

Run Windows Update and update the listed hotfixes or download and update mentioned hotfixes in the advisory from the below link, https://technet.microsoft.com/library/security/MS15-034 | | | | | | |
| Affected Software/OS
Microsoft Windows 8 x32/x64 Microsoft Windows 8.1 x32/x64 Microsoft Windows Server 2012 Microsoft Windows Server 2012 R2 Microsoft Windows Server 2008 x32/x64 Service Pack 2 and prior Microsoft Windows 7 x32/x64 Service Pack 1 and prior | | | | | | |

Abbildung 7: Ergebnis Scan

Insgesamt ist Openvas ein sehr mächtiges Tool das nicht nur einzelne Hosts sondern auch ganze Netze scannen kann. Durch die Community werden ständig Prüfroutinen hinzugefügt, sodass es aktuell bleibt und auch neue Schwachstellen zuverlässig erkennt.

Neben Openvas gibt es noch diverse andere Scanner. Nessus und Acunetix sind zwei vergleichbare Tools, die kommerziell vertrieben werden und über eine bessere Managementaufsicht verfügen ansonsten aber vergleichbare Funktionen bieten.

8. Exkurs Konfiguration Webserver

8.1. Definition des Problems

Die Konfiguration eines Webserver ist eine komplexe sowie sensible Aufgabe und entscheidend für die Sicherheit der Webanwendung. Ohne eine Sicherheitsoptimierung des Servers kann die Webanwendung ohne Probleme umgangen werden. Man bezeichnet dieses Vorgehen auch als „Härtung“ des Systems.

8.2. Apache Härtungsguide als Gegenmaßnahme

Apache ist einer der häufigsten verwendeten Webserver im Internet. Nachfolgend eine Liste mit Punkten die beachtet werden sollten um Anwendung sicher auf diesem System laufen zu lassen:

Tabelle 1: Checkliste Webserversecurity

1. Alle Default- und Testdateien wurden gelöscht
2. httpd.conf: Bei Einsatz von SSI (Server Side Includes) ist nur Option IncludesNoExec anstatt Includes erlaubt
3. httpd.conf: Das Modul MOD_EVASIVE wurde aktiviert (WAF: DOS Verhinderung)
4. httpd.conf: Das Modul MOD_SECURITY wurde aktiviert (WAF: DOS/SQL-Injection/XSS Verhinderung)
5. Das Server Root Verzeichnis wurde von /etc/httpd in ein anderes geschütztes Verzeichnis verschoben
6. Der Webserver wird nicht unter root, administrator oder Local System Account ausgeführt
7. httpd.conf: Die folgenden Module sind nicht aktiv: info, status, autoindex, imape, include, userdir, auth, suexec, cgi/cgid
8. Die globalen Zugriffsrechte werden durch .htaccess File im Web Root Verzeichnis gesteuert
9. Die Key Länge für die SSL Verschlüsselung beträgt mindestens 1024 Bit
10. httpd.conf: Die LISTEN Directive in der http.conf Section ist für Port 80 deaktiviert, um SSL zu erzwingen
11. Die Weblogdateien werden nicht unterhalb des Web Root Verzeichnisses gesichert

12. httpd.conf: Es werden kundeneigene Fehlerseiten verwendet (ErrorDocument 404 errors/404.html usw.)
13. Es werden offiziell signierte SSL Zertifikate eingesetzt
14. Es wird die aktuellste Version des Apache Servers eingesetzt
15. httpd.conf: Absicherung gegen CRIME Attacken durch Variable OPENSSL_NO_DEFAULT_ZLIB=1
16. httpd.conf: Der Zugriff auf Apache lokale Apache Dokumentation wurde deaktiviert
17. httpd.conf: Die Option LogLevel wurde mindestens auf WARN gesetzt
18. httpd.conf: Die Option Timeout wurde auf einen Wert $> 30 < 300$ s eingestellt
19. httpd.conf: Die Variable ServerSignature = Off gesetzt
20. httpd.conf: Die Variable ServerTokens = Prod gesetzt
21. httpd.conf: Die Variable SSLHonorCipherOrder = on gesetzt
22. httpd.conf: Die Variable TraceEnable = Off gesetzt
23. httpd.conf: Die Variable UserDir = disabled gesetzt
24. httpd.conf: Option Directory Listing wurde deaktiviert
25. httpd.conf: Es werden nur sichere SSL Verfahren verwendet (SSLProtocol -ALL +SSLv3 +TLSv1)
26. httpd.conf: Zugriff auf das http root Verzeichnis wurde global deaktiviert
27. httpd.conf: Options -Includes wurden aktiviert
28. httpd.conf: Options -FollowSymLinks wurde aktiviert
29. httpd.conf: Options -Indexes wurden aktiviert
30. httpd.conf: Options -MultiViews wurden aktiviert
31. SSL Zertifikate werden ausserhalb des Web root Verzeichnis gesichert
32. Der administrative Zugriff auf den Web Server erfolgt über ein gesichertes Administrationsnetzwerk

9. Abschlussbewertung

9.1. Fazit

Webanwendungen sind sehr schwer zu schützen. Jede Organisation muss damit kämpfen, dass ein Angreifer in der Regel nur eine Schwachstelle finden muss, der Betriebsverantwortliche aber alle kennen muss um sein System zu schützen. Dies ist ein Kampf der kaum zu gewinnen ist, denn wenn eins sicher ist, dann das Software immer Schwachstellen haben wird. Durch Scannen der eigenen Systeme kann ein gewisses Risikobewusstsein entstehen und mögliche Schwachstellen können aufgedeckt werden. Da aber die Scanner nur über bekannte Schwachstellenerkennung verfügen, gibt es immer noch ein sehr großes Restrisiko, dass Schwachstellen in der Anwendung aktiv sind und eventuell ausgenutzt werden können. Betreiber können nur nach dem Einbruchschutz Prinzip vorgehen und es dem Angreifer möglichst schwer machen, sodass er sich lieber ein einfacheres Ziel sucht.

9.2. Ausblick

Mit dem Internet der Dinge wird es eine ganze Reihe von neuen Webanwendung geben. Diese werden meist von Privatpersonen verwendet und sind in der Regel, sei es vom Hersteller oder aufgrund mangelnder IT Kenntnisse vom Benutzer, nur sehr schlecht bis gar nicht abgesichert. Trotzdem kontrollieren sie heute schon wichtige Bereiche des Haushalts oder überwachen bestimmte Bereiche. Ein gutes Beispiel ist das smarte Zuhause das sich viele Verbraucher wünschen und an dem mit Hochdruck gearbeitet wird. Spätestens wenn über das Smartphone die Haustür für andere geöffnet wird, muss über die Sicherheit der hinter liegenden Anwendung gesprochen werden.

Eine gute Veranschaulichung bieten die Webseiten <https://www.shodan.io/> für IoT und <http://www.vnckh.com> für den VNC Server Dienst. Shodan listet dabei Geräte und Anwendung die schlecht bis gar nicht geschützt sind. Diese Schwachstellen sind mitverantwortlich für die Existenz des Botnet mirai, dass zum großen Teil aus solchen Geräten besteht. VNCKH veranschaulicht wie fehlende Konfiguration jedes System zum Webserver macht.

Literaturverzeichnis und Quellenverzeichnis

Georgia Weidman "Penetration Testing: A Hands-On Introduction to Hacking"

Andreas Weyert „Hacking mit Kali“ Veröffentlicht: 11.08.2014

Jon Erickson „Hacking Die Kunst des Exploits“ Veröffentlicht: 08.2008

http://www.w3schools.com/sql/sql_injection.asp | Zugriff: 16.09.2016

<https://wiki.ubuntuusers.de/Wiki/Syntax/> | Zugriff : 18.09.2016

<https://github.com/ethicalhack3r/DVWA> | Zugriff : 26.09.2016

https://www.owasp.org/index.php/Main_Page | Zugriff : 09-2016 – 12-2016

<http://resources.infosecinstitute.com/command-execution/> | Zugriff : 19.10.2016

[https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)) | Zugriff: 26.10.2016

<https://www.acunetix.com/websitesecurity/cross-site-scripting/> | Zugriff: 26.10.2016

<https://www.veracode.com/security/xss> | Zugriff: 06.11.2016

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/isi_web_server_checkliste_apache_pdf.pdf?__blob=publicationFile | Zugriff: 12.11.2016

<http://itrig.de/index.php?/archives/1937-Hardening-Apache-Server-In-5min-einen-Webserver-sicherer-machen.html> | Zugriff: 12.11.2016

<https://www.shodan.io/> | Zugriff: 16.11.2016

<http://www.vnckh.com> | Zugriff: 16.11.2016

<https://www.bsi.bund.de/DE/Themen/Cybersicherheit/Tools/OpenVAS/OpenVAS.html> | Zugriff: 02.12.2016

<http://www.admin-magazin.de/Das-Heft/2012/03/Der-Vulnerability-Scanner-Open-VAS> | Zugriff: 02.12.2016

<https://www.tenable.com/products/nessus-vulnerability-scanner> | Zugriff: 02.12.2016

<http://www.acunetix.com/vulnerability-scanner/> | Zugriff: 02.12.2016

https://de.wikipedia.org/wiki/Internet_der_Dinge | Zugriff: 04.12.2016