

Praktikum zur Vorlesung IT-Sicherheit

Thema Web Application Security

In diesem Praktikum testen Sie Angriffe auf Web Applikationen, die Schwachstellen aufweisen, selber aus. Ziel ist es, dass Sie lernen, welche Dinge zu beachten sind, wenn Sie selbst Web Applikationen implementieren.

Wir führen die Angriffe in einer geschlossenen Umgebung durch:

- eine Kali-Linux VM dient als Angreiferrechner,
- eine OWASP Broken Web Applikation VM als „Opfer“-Rechner

Beide VMs sind frei im Internet verfügbar.

Hinweis:

Wenn Sie entsprechende Angriffe oder Skripte im Internet nutzen, machen Sie sich strafbar. Das Strafgesetzbuch (§202, §303) sieht empfindliche Strafen vor.

0 Vorbereitung bei der Durchführung des Versuchs zu Hause

Die folgenden Beschreibungen beziehen sich auf die Nutzung von *Oracle VM VirtualBox*.

Sie können auch VM Ware oder andere Virtualisierungstools nehmen, sind dann aber auf sich selbst gestellt.

Installieren Sie sich *Oracle VM VirtualBox*.

0.1 Appliances aus dem Internet laden und in VirtualBox importieren

Am einfachsten ist es, die VMs als Appliance mit der Endung (.ova) zu importieren.

Hierzu können Sie Kali Linux ova im Internet suchen. Ich habe den Download von

<https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>

verwendet.

Die OWASPbwa ova ist verfügbar unter

https://osdn.net/projects/sfnet_owaspbwa/releases/

https://sourceforge.net/projects/owaspbwa/files/1.2/OWASP_Broken_Web_Apps_VM_1.2.ova/download

Bitte die **64 Bit Version** verwenden, da nur diese die Burp-Suite enthält, die benötigt wird!

Beide als Appliance (.ova-Dateien) nach VirtualBox importieren, dabei "neue MAC Adressen fuer die Interfaces wählen" verwenden:

- In VirtualBox im Datei-Menü "Appliance importieren" auswählen
- Die jeweilige .ova Datei auswählen.
- Unten vor dem Import "Neue MAC-Adressen für alle Netzwerkadapter generieren" auswählen.

0.2 Netzwerke für beide VMs in VirtualBox identisch einstellen:

In VirtualBox links die jeweilige VM auswählen, dann rechts oben auf Ändern klicken.

Links „Netzwerk“ wählen. Adapter 1 auf „*Internes Netzwerk*“ umstellen, Name "*ITS-Netz*" wählen (bei beiden VMs den gleichen Namen für das Netz wählen!)

Links „Allgemein“ wählen. Unter „Erweitert“ die gemeinsame Zwischenablage und Drag'nDrop auf bidirektional stellen.

0.3 OWASPbwa VM starten und vorbereiten:

In VirtualBox links die entsprechende VM auswählen, dann rechts oben auf „Starten“ klicken.

Warten bis zum Login. Dann mit User *root* und Passwort *owaspbwa* einloggen.

IP-Adresse konfigurieren:

```
sudo ifconfig eth0 192.168.0.11
sudo ifconfig eth0 netmask 255.255.255.0
```

Konfiguration anzeigen lassen mit `sudo ifconfig`

Mit `ping 192.168.0.11` prüfen, ob das Interface erreichbar ist.

Falls die Mouse im Fenster der VM "gefangen" ist:

=> mit der rechten STRG-Taste kann sie wieder befreit werden.

0.4 Kali-Linux VM und vorbereiten:

In VirtualBox links die entsprechende VM auswählen, dann rechts oben auf „Starten“ klicken.

Warten bis zum Login. Dann mit User *kali* und Passwort *kali* einloggen. (Bei anderen Kali-VMs ist es häufig User *root* und Passwort *toor*)

Eingabeaufforderung/Terminal öffnen (TerminalEmulator oben links, mittleres Feld).

Kali-Tastatur auf deutsch umstellen: Kommando `setxkbmap de` im Terminal.

Interface IP-Konfigurieren:

```
sudo ifconfig eth0 192.168.0.100
sudo ifconfig eth0 netmask 255.255.255.0
```

Konfiguration anzeigen lassen mit `sudo ifconfig`

Mit `ping 192.168.0.11` prüfen, ob die OwaspBWA erreichbar ist.

Webbrowser (Firefox) öffnen und die IP-Adresse 192.168.0.11 als URL eingeben.

(Den Webbrowser können Sie oben ganz links unter dem Kali-Linux-Symbol auswählen.)

Sie gelangen auf die Weboberfläche der OwaspBWA.

1 Vorbereitungen

1.1 Starten der VMs und Ermittlung der IP-Adressen der VMs

Starten Sie Oracle Virtual Box. Starten Sie die OWASPbwa VM und die Kali-Linux VM.

Gearbeitet wird ausschließlich auf der Kali-Linux VM. Die OWASPbwa VM dient lediglich als Server.

Nachdem die Kali-Linux VM hochgefahren ist, loggen Sie sich mit dem Nutzer `root` und dem Passwort `toor` ein. (Oder Nutzer `kali` und Passwort `kali`.)

Starten Sie eine Kommandozeilensitzung (Terminal). Die IP-Adresse der KaliVM ermitteln Sie mit `sudo ifconfig -a`.

IP-Adresse inkl. Netzpräfix der KaliVM: `inet 192.168.0.100 netmask 255.255.255.0`

Die IP-Adresse der OWASPbwaVM ermitteln Sie mittels dem Kommando `nmap`.

`nmap -help` zeigt Ihnen am Ende 3 Examples an. Das Beispiel mit `-sn` könnte ein Ping Scan sein.

`nmap -help | grep sn` bestätigt dies. Führen Sie jetzt per `nmap` einen Ping Scan für das Netz `192.168.0.0/24` aus, um die IP-Adresse der OWASPbwaVM zu ermitteln.

Kommando: _____


Das dauert einige Sekunden. Scrollen Sie anschließend nach oben, um die IP-Adresse der anderen VM zu bestimmen.

IP-Adresse der OWASPbwa VM: _____

Prüfen Sie per Ping noch einmal die Erreichbarkeit der VM.

1.2 Starten des Web Proxies BurpSuite

Die BurpSuite ist ein Web Proxy. Burp erlaubt es, http-PDUs abzufangen, zu ändern und dann weiter zu schicken.

Starten Sie die BurpSuite  in der Schnellstart-Leiste am linken Rand.

(Ansonsten im Kali-Linux Menü unter *03 Web-Application Analysis*.)

Erstellen Sie ein temporäres Projekt mit Standardeinstellungen.

Starten Sie den Firefox-Browser (auch in der Schnellstart-Leiste links).

Der Web-Verkehr des Firefox-Browsers soll jetzt über den Proxy BurpSuite umgeleitet werden. Hierzu gehen Sie in den Firefox-Einstellungen auf *Preferences*, dort ganz unten auf *Network Proxy Settings* und wählen *Manual proxy configuration* für den HTTP Proxy unter `127.0.0.1` und Port `8080`.

Im Burp sehen wir im Reiter *Proxy* unter *HTTP history* zahlreiche Einträge zu *detectportal.firefox.com*. Unter *Intercept* wird ein GET Request zu *detectportal.firefox.com* angezeigt. Damit diese Anfragen nicht mehr abgefangen werden gehen Sie auf *Action* und wählen *Don't intercept requests => To this ip address* und zusätzlich *Don't intercept requests => To this host*

Durch Klick auf **Intercept is on** schalten Sie das Abfangen aus. Nach dem erneuten Starten des „Intercepts“ sollten jetzt keine Requests von *detectportal.firefox.com* mehr abgefangen werden, auch wenn diese unter *HTTP history* weiterhin aufgezeichnet werden.

Schalten Sie jetzt das Abfangen zunächst aus (*Intercept is off*).

1.3 Aufruf der OWASPbwa

Geben Sie im Firefox die IP-Adresse der OWASPbwaVM als URL ein. Sie gelangen auf die Startseite der OWASPbwa. Hier gibt es neben zahlreichen Trainings-Applikationen, realistische beabsichtigte verletzbare Applikationen sowie alte verletzbare Versionen realer Applikationen, wie z.B. Wordpress.

Wir arbeiten heute mit den Training Applications.

2 Cross Site-Scripting (XSS)

Öffnen Sie die *Damn Vulnerable Web Application* (DVWA) und loggen Sie sich mit dem Nutzer *admin* und Passwort *admin* ein. Die DVWA bietet die Möglichkeit, zahlreiche verschiedene Angriffe auszutesten. Dabei sind jeweils unterschiedliche Sicherheitslevel einstellbar: low, medium und high. Unten wird der aktuelle Sicherheitslevel angezeigt.

Sollte der Security Level nicht auf *low* stehen, stellen Sie ihn unter *DVWA Security* auf *low*!

2.1 Reflected XSS

Low: Gehen Sie auf *XSS reflected*. Probieren Sie Ihren Namen einzugeben und drücken Sie *Submit*. Was passiert?

Mein Name wird ausgegeben

Setzen Sie Ihren Namen zwischen die html Tags für Fettdruck: ` Name `. Was passiert?

Mein Name wird fett ausgegeben

Aha, der html-Code wird interpretiert. Klappt das auch mit einem Script? Testen Sie die Eingabe `<script>alert("XSS");</script>`. Was passiert?

Es wird ein alert ausgegeben

Unten unter *View Source* kann man im Quellcode sehen, dass der eingegebene String direkt ausgegeben wird.

Das ist eine schwerwiegende Schwachstelle! Sie haben von Ihnen gewählten Scriptcode in der Webseite ausgeführt.

Um eine Reflected XSS Schwachstelle auszunutzen, müssen Sie ein Opfer dazu verleiten, einen Web-Link zu klicken, in dem ein Script hinterlegt ist. Dann wird der Scriptcode im Browser des Opfers auf dem Opferrechner ausgeführt!

Medium: Stellen Sie den Security Level unter *DVWA Security* jetzt auf medium.

Testen Sie zunächst wieder, wie sich die Seite verhält, wenn Sie Ihren Namen eingeben und wenn Sie html Tags für Fettdruck nutzen.

Testen Sie jetzt die Eingabe `<script>alert("XSS");</script>`. Was passiert?

Es wird Hello alert("XSS"); ausgegeben

Unten rechts auf der Seite finden Sie unter View Source den Grund für das Verhalten: str_replace filtert alle Vorkommen von <script>. Versuchen Sie auf Grundlage dieser Information, das Skript zur Ausführung zu bringen. Hinweis: Die http-Engine im Browser ist nicht case sensitiv. Was führt zum Erfolg?

Ein Angreifer kann zwischen dem script und > ein Leerzeichen setzen oder eine Buchstaben von script großschreiben

Einfache Stringvergleiche reichen also nicht aus, um XSS zu verhindern.

Es gibt Möglichkeiten, in Webseiten Code auszuführen, ohne den script-Tag zu nutzen. Ein möglicher Angriffsvektor ist Event-basiertes XSS, z.B. unter Nutzung des *onload* Attributs. Beispiele sind unter www.w3schools.com/tags/ev_onload.asp angegeben. Beim Laden eines Elements kann mit *onload* direkt ein Skript ausgeführt werden. Probieren Sie

```
<body onload="alert('Mit onload ausgetrickst');"> Test </body>
```

Beim Laden der Seite (body) wird direkt das angegebene Skript ausgeführt.

High: Stellen Sie den Security Level unter *DVWA Security* jetzt auf high.

Testen Sie die Eingabe `<script>alert("XSS");</script>`. Was passiert?

Der gesamte Befehl wird ausgegeben

Wie das geht, zeigt ein Blick in den Seiten Quelltext (*View Source*). Welche PHP-Funktion wird verwendet, um die Eingabe unschädlich zu machen?

Bestimmte für html relevante Zeichen werden in Text umgewandelt, so dass diese als reiner Text ausgegeben werden

Gehen Sie im Internet auf die PHP Dokumentation der Funktion. Wie geht die Funktion in der Standardvariante mit double-quotes („ “) und single-quotes (, ') um?

Es werden double-quotes oder single-quotes umgewandelt wenn ENT_NOQUOTES nicht gesetzt wurde

Dies kann an speziellen Stellen wieder ausgenutzt werden, um Skripte zu injizieren. Bei Einsatz der Funktion also bitte darauf achten, dass ENT_QUOTES gesetzt ist.

2.2 Stored XSS

Low: Stellen Sie den Security Level unter *DVWA Security* wieder auf low.

Auf XSS stored finden Sie ein Gästebuch, bei dem Sie einen Namen und eine Nachricht eingeben können. Geben Sie Ihren Namen und als Message *Hallo* ein und klicken Sie *Sign Guestbook*.

Probieren Sie, den Namen und die Message fett gedruckt ins Gästebuch zu schreiben.

Als nächstes probieren Sie `<script>alert("Stored XSS");</script>` im Message-Feld. Was passiert?

Es wird ein alert ausgegeben

Den Unterschied zu Reflected XSS bemerken Sie, wenn Sie die Seite wechseln (z.B. links auf About gehen) und anschließend wieder XSS *stored* wählen. Was passiert? Weshalb ist das so?

Der injizierte Befehl wird erneut aufgerufen

Genau deshalb spricht man von Stored XSS. Hier ein paar Beispielangriffe:

Samy Kamkar hat 2005 mit StoredXSS auf MySpace innerhalb von 24 Stunden >1 Mio. Freunde bekommen: [en.wikipedia.org/wiki/Samy_\(computer_worm\)](https://en.wikipedia.org/wiki/Samy_(computer_worm))

Noch 2017 wurden eBay-Nutzerkennungen per StoredXXS gestohlen:

news.netcraft.com/archives/2017/02/17/hackers-still-exploiting-ebays-stored-xss-vulnerabilities-in-2017.html

Medium: Stellen Sie den Security Level unter *DVWA Security* auf medium.

Zum Löschen des Guestbooks gehen Sie auf *Setup* und klicken auf *Create / Reset Database*.

Zurück auf *XSS stored* probieren Sie gleich das Skript im Message-Feld. Was passiert?

Die Script Tags wurden entfernt und der Rest wird als Text ausgegeben

Unter *View Source* sehen sie, weshalb das so ist. Allerdings werden der Name und die Message unterschiedlich gesäubert (*sanitized*), der Name nur mit *str_replace*. Das hatten wir doch schon oben! Also probieren Sie es wie oben bei 2.1 medium. Welches Problem tritt auf?

Der Befehl kann nicht eingegeben werden, da die Anzahl der Zeichen begrenzt ist

Dann ändern Sie das! Gehen Sie im Firefox unter Einstellungen auf WebDeveloper => Inspector. Öffnen Sie den html body nach und nach so weit, bis Sie das Eingabefeld für den Namen gefunden haben. Ändern Sie dort `maxlength="10"` auf `"100"`.

Zurück auf der Seite verifizieren Sie, dass jetzt das Einfügen des Skripts im Namensfeld klappt.

Check ☒

Falls das Skript nicht ausgeführt wird, suchen Sie den Fehler.

High: Stellen Sie den Security Level unter *DVWA Security* auf high.

Wird Ihr Gästebuch-Eintrag, der gerade noch ausgeführt wurde, jetzt auch ausgeführt? Was steht jetzt im Gästebuch-Eintrag unter Name?

Der Befehl wird nicht mehr ausgeführt, sondern steht vollständig im Namen

Ermitteln Sie unter *View Source*, weshalb das so ist:

htmlspecialchars wird für Name und Message verwendet

3 SQL Injection

Per SQL Injection lassen sich Datenbanken abfragen und Authentifizierungen brechen, wenn Schwachstellen vorhanden sind. Wir testen mit dem Framework OWASP Bricks, Logins zu brechen.

Gehen Sie auf die OWASPbwa Startseite, wählen Sie *OWASP Bricks* und unter Bricks die *Login pages*.

3.1 Login #1

Versuchen Sie sich mit User *admin* und Passwort *test* einzuloggen. Nach dem Versuch wird unten das zugehörige SQL Query angezeigt, mit dem die Prüfung erfolgte. Es wird geprüft, ob zu dem angegebenen User und Passwort ein Eintrag in der Datenbank vorhanden ist. Das Login ist erfolgreich, wenn die Rückgabe aus der Datenbank nicht leer ist.

`SELECT * FROM users WHERE 1=1` liefert z.B. den gesamten Inhalt der Tabelle *users*.

Idee: als Passwort wird `1' or '1' = '1` gewählt. Wie endet dann das SQL-Query?

`WHERE name = 'admin' and password = '1' or '1' = '1'`

Damit die Passwort-Eingaben angezeigt werden, unter Einstellungen über WebDeveloper => Inspector im Seiten Quelltext *method="Post"* suchen und den *type* des Felds von von *"password"* auf *"text"* ändern.

Damit sollte der Login klappen. Falls nicht, suchen Sie den Fehler.

3.2 Login #2

Versuchen Sie sich mit User *admin* und Passwort *test* einzuloggen. Das SQL-Query ist identisch. Versuchen Sie es wie zuvor mit dem Passwort `1' or '1' = '1`. Was passiert?

Es wird ein alert ausgegeben, weil Sonderzeichen nicht erlaubt sind

Es stellt sich die Frage, ob die Prüfung im Client oder im Server erfolgt. Das ermitteln wir mit Burp. Aktivieren Sie auf Burp das Abfangen (Intercept).

Versuchen Sie sich mit User *admin* und Passwort *test* einzuloggen. Der Request wird von Burp abgefangen. Schicken Sie den Request per Klick auf *Forward* weiter an den Server.

Versuchen Sie es jetzt mit User *admin* und Passwort `1' or '1' = '1`. Was zeichnet Burp jetzt auf? Die gesamte HTML Seite inklusive script

Wo also erfolgt die Prüfung? Im Client oder im Server? Im Client

Idee: Wir ändern den Request NACH der Prüfung.

Versuchen Sie sich mit User *admin* und Passwort *test* einzuloggen. Jetzt wird der Request vom Client abgeschickt und von Burp abgefangen. In Burp können Sie im Request das Passwort durch `1' or '1' = '1` ersetzen. Anschließend den Request per *Forward* an den Server weiterleiten. Was passiert?

Der log in war erfolgreich

3.3 Login #3

Schalten Sie das Intercept in Burp wieder ab.

Versuchen Sie sich mit User *admin* und Passwort *test* einzuloggen und betrachten Sie das SQL-Query. In diesem werden jetzt Klammern verwendet.

Wie müssen Sie (ähnlich wie oben) das Passwort wählen, um sich einzuloggen?

)1' or '1' =1(

3.4 Login #5

Wir überspringen Level#4 und gehen direkt zu Level#5.

Versuchen Sie sich mit User *admin* und Passwort *test* einzuloggen und betrachten Sie das SQL-Query. Was ist anders?

Das Passwort wird gehasht

Fangen Sie den Request mit Burp ab. Was stellen Sie fest:

Das Passwort wird als Klartext übermittelt

Versuchen wir diesmal, den Namen direkt abzukürzen: 1' or '1' =1

Schicken Sie den Request weiter. Ermitteln Sie auf Basis des angezeigten SQL-Queries, warum der Versuch nicht erfolgreich war:

Das Passwort wird auf dem Server gehasht

Wenn man nur das Query vor der *and*-Verknüpfung beenden könnte Ja, das geht! Zur Erinnerung: -- dient in SQL zur Kommentierung. Alles hinter einem -- ist Kommentar. Bitte Beachten: Der Kommentar MUSS mit einem LEERZEICHEN beginnen.

Also schließen Sie den Namen und machen Sie alles hinter einer *or 1=1* zum Kommentar! Was müssen Sie als Namen eingeben, um sich erfolgreich einzuloggen?

' or 1=1 -- kommentar

Das Passwort ist damit völlig ausgeblendet.

4 Cross Side Request Forgery CSRF

Beim CSRF wird über einen untergeschobenen Link versucht, eine bereits geöffnete Sitzung eines anderen Users zu übernehmen.

Beispielszenario: Ein Nutzer hat sich ins Online-Banking eingeloggt und die Webseite hat eine CSRF Schwachstelle. Wenn der Nutzer dann auf einen vom Angreifer präparierten Link (z.B. in einer E-Mail) klickt, kann der Angreifer die Session übernehmen.

Um das zu testen, gehen Sie wieder auf die *Damn Vulnerable Web Application* (DVWA), loggen sich mit *admin/admin* ein und stellen Sie den Security Level auf **low**.

Gehen Sie auf CSRF. Die Seite bietet die Funktion, dass Admin-Passwort zu wechseln. Ändern Sie das Passwort auf *test* und achten Sie auf die URL. Was passiert in der URL?

Das neue Passwort wird als Parameter in der URL angegeben

Dann versuchen Sie in einem Terminal, einen entsprechenden URL-Seitenaufruf zur Passwortänderung. Dazu nutzen Sie folgendes `curl` Kommando:

```
curl -s "http://192.168.0.11/dvwa/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change"
```

Wurde das Passwort geändert? Bitte begründen.

Das Passwort wurde nicht geändert, da der Aufruf für curl wahrscheinlich blockiert wurde, da sich ein Nutzer erst anmelden muss um das Passwort zu ändern

Öffnen Sie im Editor die Datei *dasIstSuper.html*. Die Datei enthält einen versteckten `iFrame`, der die URL für den Passwortwechsel aufruft.

Loggen Sie sich aus der DVWA aus. Dann öffnen Sie *dasIstSuper.html* im Firefox (z.B. neuer Tab, drag&drop). Anschließend loggen Sie sich wieder ein. Wurde das Passwort geändert? Begründung?

Das Passwort wurde nicht geändert, da der Benutzer ausgeloggt war

Loggen Sie sich mit dem korrekten Passwort in die DVWA ein. Jetzt öffnen Sie *dasIstSuper.html* im Firefox. Wurde das Passwort geändert? Begründung?

Das Passwort wurde geändert, da beim Aufruf der html über einen iFrame im Hintergrund die URL zum Passwort wechsel aufgerufen wurde

Sie haben es geschafft einen Passwortwechsel durchzuführen, ohne dass der Nutzer davon etwas bemerkt hat!

Stellen Sie das Passwort bitte wieder auf admin zurück!

5 File Uploads

File Uploads zu erlauben ist für Webseiten generell gefährlich.

Wir testen das mit OWASP Bricks aus. Gehen Sie in Bricks auf File Upload pages.

5.1 Dateien in die Kali-VM kopieren

Um die Uploads zu testen, müssen Sie zunächst die Dateien ins Kali-Linux kopieren.

In den neueren Kali-Versionen klappt das per Drag'n Drop direkt auf den Kali-Desktop.

Falls das nicht klappen sollte, legen Sie auf dem Desktop einen neuen „Folder“ an (Auswahl über rechte Mause Taste), öffnen den Folder und ziehen dann die Dateien per Drag'n Drop in den Folder.

Voraussetzung ist, dass in Virtual Box für die Kali-VM unter „Allgemein“=>„Erweitert“ das Drag'n Drop auf bidirektional gestellt ist.

5.2 File Uploads mit Prüfung austricksen

In Bricks wählen Sie unter File Upload pages direkt den „Upload #3“:

Versuchen Sie folgende Dateien hochzuladen (Wählen Sie jeweils die Datei über *Browse*, klicken Sie *Upload*. Anschließend können Sie unter *here* einen Blick auf die Datei werfen.)

Datei	Ergebnis
HSLogo.png	Erfolgreich
Textdatei.txt	Fehlgeschlagen
echoversion.php	Fehlgeschlagen

Sie haben jetzt ermittelt, welche Dateierweiterung akzeptiert wird: .png

Testen Sie folgendes: Ändern Sie den Namen der Datei *Textdatei.txt* auf *Textdatei.png* und laden Sie die Datei hoch. Was passiert?

Das Hochladen war erfolgreich

Aktivieren Sie das Intercept in Burp und Wiederholen Sie den Upload von *Textdatei.png*. In Burp ändern Sie im abgefangenen POST Request die Dateierweiterung wieder zu *.txt* und schicken den POST dann weiter. Was passiert?

Das Hochladen war erfolgreich

Anstelle die Datei auf *.png* umzubenennen und dann beim Hochladen wieder auf *.txt* zu ändern, kann auch die Originaldatei hochgeladen werden. Damit der Server diese akzeptiert, versuchen Sie den Content-Type im POST Request auf *image/png* zu ändern.

Testen Sie das mit dem PHP-Skript *echoversion.php*. Was passiert?

Das hochladen war erfolgreich es wurde jedoch kein Skript ausgeführt

Jetzt ist es Ihnen als Angreifer gelungen über den Upload ein PHP-Skript auf dem Server auszuführen! Das ist absolut kritisch. Ein Web-Administrator sollte es nicht soweit kommen lassen!

Sie sehen, eine Prüfung der Dateierweiterung oder des Content-Typs reichen nicht für einen sicheren Upload aus.

6 Biete SSL/TLS Schutz?

Überlegen Sie, ob ein Schutz der Verbindung mit SSL bzw. TLS die bisher beschriebenen Angriffe verhindert. Wenn Sie es ausprobieren möchten, versuchen Sie einen der beschriebenen Angriffe über https. Die OWASPbwaVM erlaubt einen https-Zugriff.

Wählen Sie einfach https://192.168.0.11. Anschließend müssen Sie ggf. noch das Zertifikat akzeptieren, falls es nicht im Browser vorhanden ist. (Advanced => AddException => Confirm Security Exception)

Bietet SSL/TLS Schutz gegenüber entsprechenden Angriffen? Bitte begründen.

SSL bietet keinen Schutz, da der Angreifer keine Man in the middle Angriffe anwendet

7 Aufräumen

Falls Sie noch Zeit haben, können Sie die Option (s.u.) noch bearbeiten.

Ansonsten schließen Sie die virtuellen Maschinen folgendermaßen:

Gehen Sie in VirtualBox auf die entsprechende Maschine => rechte Moustaste => Schließen wählen => Ausschalten wählen => Ausschalten bestätigen.

8 Option: Remote Command Execution

Die Schwachstelle beim File Upload kann in einfacher Weise verwendet werden, um auch anderen Code auszuführen.

Damit nicht ständig die Requests in Burp abgefangen und geändert werden müssen, wechseln Sie in Bricks auf Upload#1, bei dem keine Prüfungen durchgeführt werden. Alles Folgende würde ebenso mit den geschützten Uploads#2 und #3 gehen mit dem jeweiligen Abfangen und Ändern der Requests in Burp.

Schauen Sie sich die Datei *command.php* im Editor an. Das PHP-Script liest den Parameter *cmd* ein und führt ihn in einer Shell aus.

Laden Sie *command.php* hoch und öffnen Sie den Link zur Datei. Noch passiert nichts. Ergänzen Sie jetzt die URL am Ende um `?cmd="ls"`. Was passiert?

Es geht noch schlimmer. Wir können auch eine Shell auf dem Server starten.

Normaler Weise erfolgt dies per telnet oder SSH. Das ist jedoch in der Regel nicht möglich, da die Firewall Zugriffe von außen blockiert.

Also müssen wir die Shell so aufbauen, dass die Shell vom Server Kontakt nach außen, zum Angreiferrechner aufbaut. Man spricht auch von einer Reverse-Shell.

Das Skript *php-reverse-shell.php* startet auf dem Server eine Shell und versucht dann eine Verbindung zu 192.168.0.100 auf Port 1235 aufzubauen. (Das Originalscript gibt es frei auf GitHub.)

Auf dem KaliLinux müssen wir jetzt auf Port 1235 lauschen und die Verbindung vom Server annehmen. Hierzu starten Sie ein Terminal und verwenden Sie den Befehl

```
nc -nlv 192.168.0.11 -p 1235
```

Anschließend das Skript *php-reverse-shell.php* hochladen und auf den Dateilink gehen.

Im Terminalfenster ist jetzt die Shell aktiv! Probieren Sie folgende Kommandos

```
ls-al          ps -l          whoami
```

Brechen Sie die Verbindung mit STRG-C ab.

Versuchen Sie die Datei *command.php* in ein Verzeichnis des Windows10-Hosts zu kopieren und dort mit einem Editor zu öffnen. Was stellen Sie fest? Wieso ist das so.