Hochschule
für Technik
Stuttgart

# Entwurf und Umsetzung einer Laborumgebung zum Thema Web Application Firewall

Bachelor Thesis
im Studiengang Informatik


| | |
|---|---|
| Betreuender Dozent: | Prof. Dr. Jan Seedorf |
| Betreuer des Unternehmens: | Oliver Paukstadt |
| Vorgelegt am: | 27. März 2024 |
| Vorgelegt von: | Lukas Reinke (Mat. NR. 1001213) |

# 1 Abstract

Hier könnte ihr abstract stehen.

# Inhaltsverzeichnis

# 2 Abkürzungsverzeichnis

**DMZ** Demilitarierte Zone

**DSGVO** Datenschutz-Grundverordnung

**HTTP** Hypertext Transfer Protokoll

**HTTPS** Hypertext Transfer Protokoll Secure

**IT-SiG** IT-Sicherheitsgesetzes

**MITM** Man in the Mittle

**NAT** Network Adress Translation

**TLS** Transport Layer Security

**URL** Uniefied Resource Locator

**VM** Virtuelle Maschine

**WAF** Web Application Firewall

**XSS** Cross Site Scripting

# 3 Abbildungsverzeichnis

# 4 Einleitung

In einer Zeit in der das Internet und die darauf basierenden Technologien in fast allen Bereichen der Wirtschaft und Gesellschaft zunehmen wichtiger sind, ist die Verteidigung dieser Technologien gegen Angreifer ein wichtiges und präsentes Thema. Im Wettrüsten zwischen Verteidigern und Angreifern muss sich konstant auf neue Angriffstechniken eingestellt werden. Als die ersten Server öffentlich im Internet verfügbar waren, war es üblich diese ohne jegliche Filterung zu tun. Nach und nach mussten die Verteidigungen aufgerüstet werden. Inzwischen ist die bloße Filterung auf Port-Ebene, um den Zugriff auf bestimmte Services zu beschränke, nicht mehr ausreichend. Angreifer nutzen Schwachstellen in den öffentlich zugänglichen Anwendungen aus um ihre Ziele zu Erreichen. Es ist notwendig den Inhalt der Netzkommunikation selber zu analysieren.
Deswegen gewinnt das Feld der Web Application Firewall (WAF) aktuell immer mehr an Relevanz. Dem Markt wird in den nächsten fünf Jahren ein jährliches Wachstum um 19,9 % auf 14,6 Mrd.$ vorhergesagt **WebApplicationFirewall**. Auch Ausarbeitungen zum *Stand der Technik* wie sie zum Beispiel im Deutschen IT-Sicherheitsgesetzes (IT-SiG) und der Datenschutz-Grundverordnung (DSGVO) gefordert werden, beschreiben eine WAF als notwendig zur Absicherung einer Webanwendung[1, 3.1.19 Schutz von Webanwendungen].
Diese Bachelor Thesis beschäftigt sich mit der Erarbeitung von Lerneinheiten und einer Laborumgebung anhand derer das Thema WAF vermittelt werden kann. Es werden Design beziehungsweise Implementierung einer WAF betrachtet. Auch werden Themen vermittelt, die für den Betrieb einer WAF in einem produktiven Umfeld relevant sind. Dazu zählen zum Beispiel Deployment und Anpassung einer WAF an die zu schützende Webanwendung.

# 5 Theoretische Grundlagen

## 5.1 Hypertext Transfer Protokoll

Das HTTP ist ein Protokoll das in der Internet-Kommunikation zum Einsatz kommt. Es dient zur Übertragung von Daten zwischen einem Client und einem Server und wird hauptsächlich zur Auslieferung beziehungsweise Bedienung von Webanwendungen genutzt. Seit der Einführung in 1991 wurde es in mehreren RFCs erweitert und ist inzwischen in Version drei[2].

Das der HTTP-Kommunikation zu Grunde liegende Interaktionsmuster erfolgt nach den *Request-Response-Pattern*. Ein Client beginnt die Kommunikation und fragt Daten oder Daten-Operationen an einem Server an. Der Server beantwortet diese Anfrage[1] und sendet dem Client Informationen wie die Verarbeitung erfolgt ist und die Angeforderten Daten. Diese Kommunikation erfolgt über TCP. Ist ein Request-Response Cycle abgeschlossen, wird die TCP-Verbindung gekappt und der Server kann keine weiteren Nachrichten an den Client schicken bis dieser eine neue Anfrage startet. Übertragen werden können Texte in diversen Formatierungen wie beispielsweise H*TML-Dokument* oder *JSON-Strings*, es ist jedoch auch möglich Binär-Daten wie *Bilddateien* zu übertragen. HTML ist ein *Plaintext*-Protokoll, die Kommunikation erfolgt unverschlüsselt. Außerdem ist HTTP Textbasiert, eine Nachricht wird in Textform übertragen und ist menschenlesbar.
Die oben beschrieben Eigenschaften beziehen sich auf HTTP in Version 1. Seit dieser Version wurden einige Erweiterungen an HTTP vorgenommen, hauptsächlich mit dem Ziel die Kommunikationsgeschwindigkeit des Protokolls zu erhöhen. So kann TCP-Pipelining genutzt werden um mehrere TCP-Nachrichten nacheinander zu schicken ohne Bestätigung des Erhaltens der Nachricht abzuwarten. Außerdem werden Push-Nachrichten unterstützt: Der Server kann durch Aufrechterhaltung einer TCP-Verbindung weitere HTTP-Nachrichten an den Client schicken. Mit Version 3 wurde HTTP Außerdem auf das Stream-basierte QUIC Protokoll umgestellt um die Kommunikation weiter zu Optimieren[3].

Da die grundlegenden Funktion von HTTP 1.0 und 1.1 ausreichend ist um eine WAF zu erklären, wird im Folgenden nur diese betrachtet.

#### HTTP-Nachrichten [4]

Die Grundlegende Einheit einer HTTP-Kommunikation wird als *Nachricht* bezeichnet. Da HTTP ein Klartext-Protokoll ist, werden diese in menschenlesbarer Form als Text

[1] Im folgenden werden HTTP-Anfrage und das Englische HTTP-Request synonym verwendet. Das Gleiche gilt für HTTP-Antwort und HTTP-Response

übertragen. Eine Nachricht besteht aus einer *Start-Zeile*, die die Nachricht entweder als Anfrage oder Antwort identifiziert. In den beiden Fälle ist der Aufbau dieser Zeile unterschiedlich:

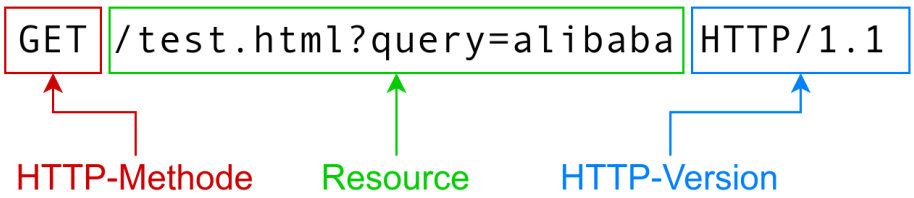



Abbildung 1: Beispiel für eine HTTP Request-Zeile

Quelle: Eigene Darstellung

**Request-Zeile:** Ein HTTP-Request ist durch eine *Request-Zeile* identifiziert. Diese ist in drei Teile aufgeteilt (siehe Abbildung 1).

**Die HTTP-Methode** beschreibt die Operation, die der Server ausführen soll. Die Operationsbezeichner können auch *HTTP-Verben* und *HTTP-Nomen* genannt werden. Syntaktisch basieren die Methoden auf dem FTP Protokoll, das älter ist und mit Operationen wie GET und PUT arbeitet. In höheren Versionen wird der Satz an Verben und Nomen jedoch um weitere Methoden, wie zum Beispiel PATCH oder OPTIONS, erweitert.

**Die Resource** beschreibt das angefragte Objekt, üblicherweise in einer Uniefied Resource Locator (URL). Optional kann die angefragte Resource mit einem Fragezeichen gefolgt von einem sogenannten *Query String* noch spezifiziert beziehungsweise gefiltert werden.

**Die HTTP Version** die angibt in welcher Version die folgende Nachricht verfasst ist.

**Status-Zeile:** Die Antwort auf einen Request beginnt mit einer Status-Zeile in einem eigenen Format, die wie die Request-Zeile aus drei Teilen besteht (siehe Abb. 2).

**Die HTTP Version** gibt, wie an dritter Stelle des HTTP-Requests, die HTTP-Version der Folgenden Nachricht an.

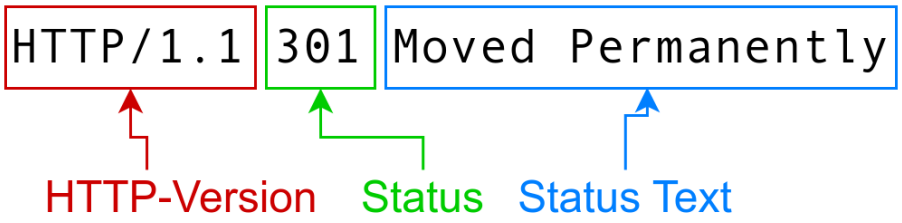



Abbildung 2: Beispiel für eine HTTP Status-Zeile

Quelle: Eigene Darstellung

**Der Status Code** der durch einen dreistelligen Zahlencode angibt wie der Status der Verarbeitung eines Requests ist. Die erste Stelle Signalisiert grob von welcher Art das Ergebnis ist. So steht zum Beispiel 2xx für eine Erfolgreiche Verarbeitung während 4xx auf einen Fehler auf Client-Seite hindeutet. Die zwei folgenden Stellen geben genauer Aufschluss wie der Status ist[5].

**Der Status Text** ist ein Beliebiger Text der den Status genauer beschreibt. Im HTTP Standard ist zwar nicht vorgeschrieben wie der Text auszusehen hat, es gibt jedoch Konventionen[5].

Nach der Start-Zeile folgen sowohl in einem HTTP-Request als auch in der Response zwei weitere Abschnitte. Diese werden genutzt, um der Nachricht weitere Informationen hinzuzufügen.
Die **HTTP-Header** sind Key-Value Pare. Der Key ist ein *case sensitiver* String. Der Value ist beliebig darf jedoch keinen Zeilenumbruch enthalten, da dies den Beginn eines neuen Header Key-Value-Pares anzeigt. Die Header teilen sich in unterkategorien wie *Request-* oder *General-Header* auf, an diesem Punkt ist es im Rahmen dieser Thesis jedoch nicht notwendig weiter in die Tiefe zu gehen. Eine Relevante Header-Gruppe sind die *Representation-Header*, die die Formatierung des HTTP-Bodies genauer spezifizieren. Hier kann der Datentyp und die Endcodierung des Bodies angegeben werden.
Der **HTTP-Body** enthält die Daten die mit der HTTP-Nachricht versendet werden und ist optional Im Body können sich beliebige Daten befinden: Binärdaten wie Bilder, JSON-Formatierte Strings oder einfacher Text[4].

Wie oben beschrieben, bietet HTTP zahlreiche Möglichkeiten an, einem Server Daten zu übergeben oder von einem solchen Daten zu erhalten. Alle Felder werden verarbeitet und bieten somit theoretisch die Möglichkeit zur Übermittlung schadhafter Daten.

Auch die Möglichkeit zur codierten Übertragung, speziell im *HTTP-Body*, kann zu diversen Angriffsvektoren führen. Eine WAF muss in der Lage sein alle dieser Parameter überblicken zu können um effektiven Schutz zu bieten.

## 5.2 Verschlüsselung

Da HTTP Kommunikation unverschlüsselt erfolgt besteht die Möglichkeit, dass ein Angreifer der sich in einer Man in the Mittle (MITM) Position befindet sensible Daten abgreifen kann. Beispielsweise kann ein Angreifer in einem öffentlichen WLAN-Netzwerk wie es beispielsweise in Bahnen oder an einigen öffentlichen Plätzen zur Verfügung gestellt wird Datenverkehr abgreifen. Dadurch können Login-Credentials, Session IDs oder Zahlungsdaten die in einem solchen Netzwerk ausgetauscht werden abgehört werden. Um derartige Angriffe verhindern zu können existiert das verschlüsselte Protokoll **Hypertext Transfer Protokoll Secure (HTTPS)**. Nahezu alle Webseiten werden inzwischen mit Hilfe des Protokolls ausgeliefert.

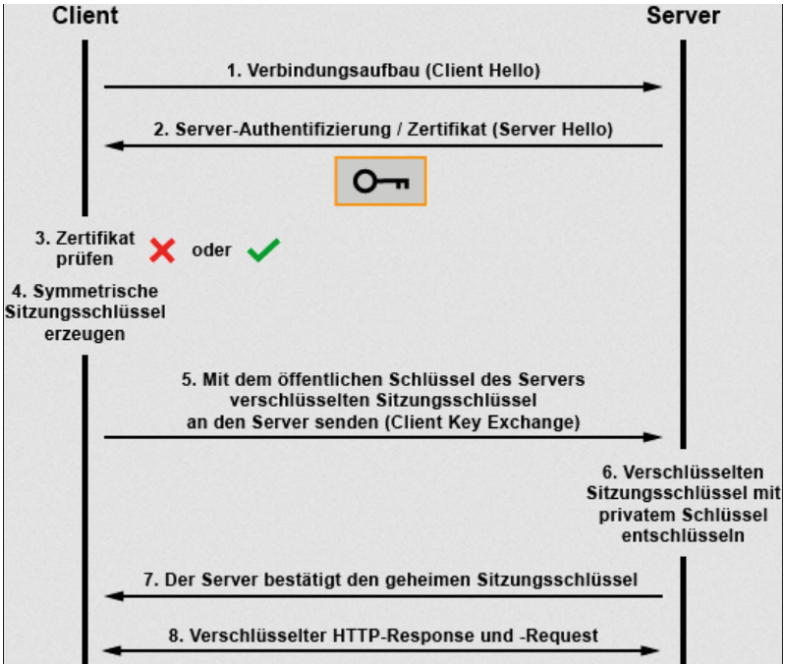



Abbildung 3: Prozess Ablauf eines HTTPS Verbindungsaufbaus

Quelle:[6]

Die Funktion von HTTPS ist die Integrität und Vertraulichkeit der Kommunikation zwischen Webservern und Clients sicherzustellen. Dies wird durch das Protokoll Transport Layer Security (TLS) umgesetzt das in der Lage ist die HTTP Kommunikation zu verschlüsseln. TLS kann nicht ausschließlich im Rahmen von HTTPS eingesetzt werden und kann beispielsweise auch zur Verschlüsselung von Mail-Kommunikation(SMTP) zum Einsatz kommen. Der Ablauf des Verbindungsaufbaus und Zustandekommen der Verschlüsselung die bei HTTPS zum Einsatz kommt ist in Abbildung 3 schematisch dargestellt.

Der Erste wichtige Vorgang ist das sich der Server, nachdem der Client eine Verbindung aufgebaut hat, mit einem TLS-Zertifikat bei den Client meldet. Der Client ist mit diesem Zertifikat in der Lage die Identität des Servers zu bestätigen. Dies erfolgt über eine Kette von Zertifikaten die sich gegenseitig bescheinigen authentisch zu sein. Die Mathematischen Prinzipien, die für dieses Verfahren notwendig sind sind für diese Thesis nicht relevant und werden nicht genauer betrachtet. Nachdem der Client die Identität des Servers bestätigt ist es laut Protokoll auch möglich, dass sich der Client mit dem gleichen Verfahren beim Server Authentifizieren kann, dies wird in der Realität jedoch äußerst selten vollzogen.

Die beiden Kommunikationspartner haben sich nun gegenseitig ihre Identität versichert und können nun beginnen einen Schlüssel auszutauschen den beide nutzen können um verschlüsselt kommunizieren zu können. Um dies zu erreichen ohne das eine etwaige dritte Person diesen Schlüssel Abhören und so die Kommunikation mitlesen kann erstellt der Client einen Schlüssel (shared Secret). Diesen übermittelt er dem Server asymmetrisch, mit dem öffentlich Schlüssel des Servers verschlüsselt. Das *shared Secret* kann nun ausschließlich mit dem privaten Schlüssel des Servers geöffnet werden. Dieser Schlüsselaustausch kann auch mit den *Diffie-Hellman* Verfahren ermittlet werden. Dies ist zum Zeitpunkt eher selten.
Die Kommunikation zwischen Client und Server kann nun verschlüsselt stattfinden und nicht von einer dritten Person auf dem weg abgefangen und verstanden werden. Auch kann sich der Client sicher sein direkt mit dem Server zu kommunizieren und keine MITM-Attacke statt findet.

## 5.3 Weiterleitung von Netzwerkkommunikation

In den meisten Fällen ist es nicht ohne Weiteres möglich eine direkte Verbindung zwischen einem Client und einem Server aufzubauen. Die Gesamtzahl aller IPv4 Adressen ($2^{32}$) reicht inzwischen nicht mehr aus um allen Geräten, die eine Internet Verbindung besitzen, eine eigene, öffentliche IP-Adresse zuzuordnen. Auch ist es zum Teil notwendig Netzwerkverkehr zu einem Server umzuleiten. Beispielsweise wenn auf einem Server mehrere Services betrieben werden, die unter unterschiedlichen Namen erreichbar sein sollen, Auch Sicherheitsbedenken können es notwendig machen eine interne Netzstruktur für einen externen Beobachter ersichtlich zu machen.
Aus den oben genannten Gründen muss in einigen Fällen der Netzwerkverkehr weitergeleitet beziehungsweise umgeschrieben werden. Auch eine WAF operiert auf Basis einer solchen Technologie. Zwei dieser Technologien werden im Folgenden genauer beschrieben.

**Network Adress Translation (NAT)** ist ein Oberbegriff für Technologien mit Hilfe derer es möglich ist auf Ebene der Schicht 2 des TCP/IP Modells Netzwerkverkehr zu verändern. Das heißt, Traffic kann durch umschreiben der IP-Adresse in einem Paket an einem Relay-Punkt an unterschiedliche Netzwerk-Teilnehmer weitergeleitet werden. Ein Einsatzgebiet für NAT ist zum Beispiel wenn mehrere Netzwerk-Teilnehmer mit einer IP-Adresse auf das Internet zugreifen.
Aus Sicht der Netzwerksicherheit bietet NAT Vorteile. Da auf einer niedrigen TCP/IP Schicht operiert wird, kann keine Transportverschlüsselung geöffnet werden und Inhalte betrachtet werden. Protokolle die Verschlüsselung anbieten arbeiten auf höheren Schichten. Der einzige kleine Vorteil kann die Verschleierung einer Internen Netzwerkstruktur sein[7].

Auf der Anwendungsschicht (Layer 4) operieren **Netzwerk Proxys**. Diese können TCP/IP Layer 4 Protokolle auf Basis von Informationen in den Nachrichten weiterleiten. Anstatt eine Verbindung direkt mit dem Ziel aufzubauen kommuniziert ein Client mit dem Proxy, der dann mit dem gewünschten Server kommuniziert. Ein Proxy dient also als Vermittler zwischen Client und Server. Für das Thema WAF speziell relevant sind **Reverse Proxies**. Ein Reverse Proxy befindet sich in einem privaten Netzwerk mit dem Ziel-Server, ist aber als einziger in diesem Netzwerk auch in der Lage mit einem öffentlichen Netz zu kommunizieren(Siehe Abbildung 4). Eingehender Netzwerkverkehr wird Beispielsweise anhand der URL in einem HTTP-Paket an den zugeordneten Server weitergeleitet. Der Proxy hält die Verbindung zu einem Client aktiv bis er eine Antwort vom Server erhält. Antwortet dieser ordnet der Reverse Proxy diese dem passenden Client zu, gibt die Antwort des Servers weiter und schließt die Verbindung zum Client. Transportverschlüsselung wird von dem Reverse Proxy vorgenommen.
Hieraus ergeben sich einige positive Sicherheitsaspekte. Da einem Reverse Proxy Kom-

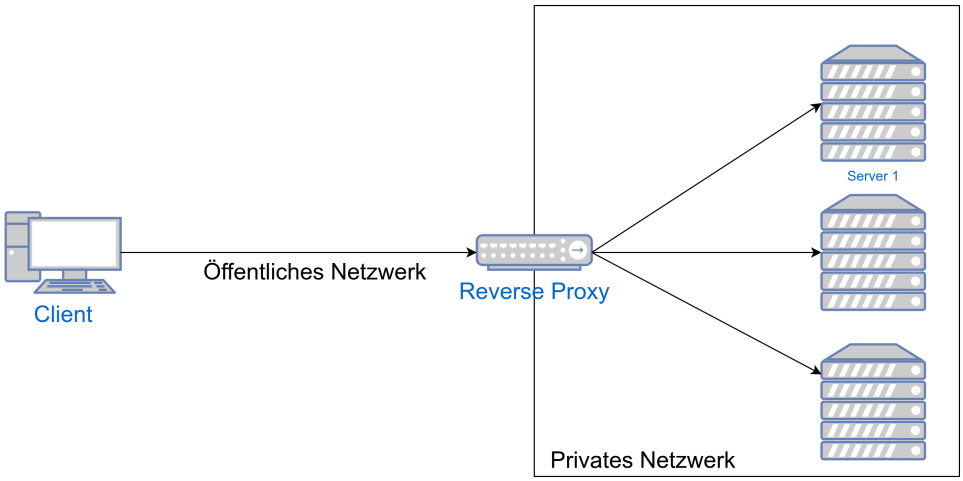



Abbildung 4: Schematische Darstellung eines Reverse Proxies

Quelle: Eigene Darstellung

munikation unverschlüsselt vorliegt kann diese Inspiziert werden. Anwendungen die Netzwerksicherheit auf Layer 4 betreiben operieren als Reverse Proxy um tief in Netzwerkverkehr blicken zu können. Eine WAF kann konzeptionell als Modul auf einem Reverse Proxy betrachtet werden[8].

## 5.4 Web Application Firewall

Ein WAF ist eine Sicherheits-Applikation, die in der Lage ist den Datenverkehr zu und von einer Webanwendung zu Analysieren. Hierbei werden die Übertragenen Inhalte in der Tiefe auf schädliche Inhalte überprüft. Der in Abbildung 5 dargestellte Prozessablauf beschreibt wie eine WAF Nachrichten verarbeitet.

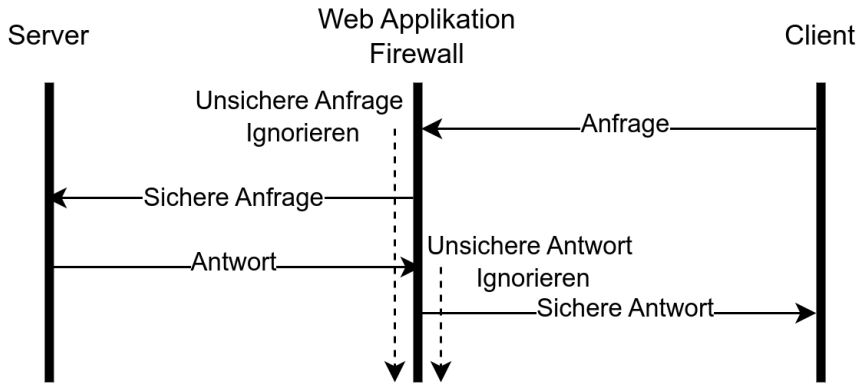



Abbildung 5: Prozess Ablauf der Verarbeitung durch eine Web Application Firewall

Quelle: In Anlehnung an [9, S. 8]

Das grundlegende Muster ist, dass einen Nachricht an der WAF eintrifft und von dieser an den Server weitergegeben wird. Dieser Verarbeitet die Anfrage und sendet seine Antwort an die WAF die die Antwort an den Client weiterleitet. Der Server kennt in diesem Muster den Client nicht. Die WAF ordnet der Anfrage die zugehörige Antwort zu. Als schädlich erkannte Inhalte können sowohl in einer Anfrage als auch Antwort abgelehnt werden. Anstatt einer abgelehnten Nachricht kann eine WAF auch eine eigene Antwort an den Client senden. Das ermöglicht es Nutzern festzustellen, dass ein Fehler in seiner Anfrage oder der Konfiguration der WAF vorliegt. Mittels einer ID, die der ersetzten Antwort angehängt wird lässt sich das Verhalten im Nachhinein nachvollziehen. Neben den beiden Möglichkeiten *weiterleiten* und *ablehnen* kann eine WAF die in einer Anfrage erkannten schädlichen Inhalte auch neutralisieren und die Nachricht weitergeben[10].

Eine WAF wird in produktiven Umgebungen in einem Netzwerk *vor* den zu schützenden Anwendungen positioniert. Das heißt der geschützte Server befindet sich in einer Demilitarierte Zone (DMZ). Eine DMZ ist ein, von anderen Netzwerkaktivitäten abgeschnittenes Netzwerksegment, das nur durch die WAF sowie weitere Sicherheitsan-

wendungen zugänglich ist. Dadurch soll verhindert werden, dass ein Nutzer auf anderem Weg als vorgesehen Zugriff auf den Server erhält.
Neben den *on-premise* Deployment-Methoden, bei denen die WAF als Virtuelle Maschine (VM) oder physischer Server im gleichen privaten Netzwerk wie die zu schützenden Server platziert ist, werden auch einige WAF als sogenannte *Cloud-WAF* zur Verfügung gestellt. Die WAF ist hier nicht im selben Netzwerk wie der Server sondern wird in einem fremden Netzwerk betrieben. Die Daten fließen von der WAF durch das öffentliche Internet zu den zu sichernden Servern. Dadurch ergeben sich einige Änderungen im Aufbau des Deployments: Es muss sichergestellt werden, dass ein Client der eine Verbindung zum Server aufbauen möchte, bei der Namensauflösung (DNS) auf die WAF geleitet wird. Der Server hingegen stellt sicher nur Verbindungen von der WAF zu akzeptieren. Dies kann mit einer Firewall, die nur Verbindungen von der WAF akzeptiert, realisiert. Es existieren auch Zertifikat basierte Verfahren, die die Authentizität der Datenherkunft sicherstellen.

Eine Firewall[2] nimmt Filterung auf Internet- und Transportschicht (Layer 2 und 3) des TCP/IP Modells vor. Operiert also hauptsächlich mit IP-Adresse und Port Nummer. Eine WAF hingegen arbeitet auf der Anwendungsschicht (Layer4). Der Fokus liegt auf Internet-Protokollen wie HTTP und HTTPS. Aber auch Datentransferprotokolle wie FTP sind im Funktionsumfang einer WAF.
Da es sich bei HTTP und auch FTP um plaintext Protokolle handelt, bei denen der Datentransport in einer menschenlesbarer Form erfolgt, unterscheidet sich die Funktion einer WAF grundsätzlich von der einer Firewall. Die Erkennung schädlicher Inhalte erfolgt aufgrund von Regeln, die Muster beschreiben die auf diese hindeuten. In der Implementierung wird dies in der Regel durch die Verwendung Regulärer Ausdrücke realisiert, diese bilden Muster ab die mit den Inhalten des Datenverkehrs abgeglichen werden. Eine WAF muss jedes Datenpaket mit einer Anzahl von Regeln in der Größenordnung mehrerer hunderttausend bis Millionen Regeln abgleichen. Der Rechenaufwand kann zwar durch optimierte regex-Engines oder Tree-Pruning-Algorithmen die irrelevante Überprüfungspfade erkennen, verringert werden. Jedoch ist der Rechenaufwand in einer WAF relativ hoch und die Hardware Anforderungen an eine WAF groß. Auch muss bei einer vorgeschalteten WAF mit erhöhten Antwortzeiten gerechnet werden, die sich im Millisekunden Bereich befinden.
Da es im Aktuellen stand der Technik üblich ist Internet-Datenübertragung zu verschlüsseln, muss eine WAF in der Lage sein die verschlüsselte Kommunikation zu öffnen um so die Inhalte analysieren zu können. Die SSL Verschlüsselung des HTTPS Protokolls wird also an der WAF terminiert. Die Kommunikation zwischen Server *hinter* der WAF und der WAF selber kann entweder unverschlüsselt oder mit einem separa-

[2]Wenn im folgenden von einer *Firewall* gesprochen wird ist eine Firewall auf IP und Port Ebene gemeint. Eine Web Application Firewall wir immer mit WAF bezeichnet.

ten Satz Zertifikate erfolgen. Die WAF verschlüsselt die Kommunikation nach der Verarbeitung wieder um sie an den Server weiterzugeben. Die Abwägung ob in der DMZ verschlüsselt kommuniziert wird muss anhand von Compliance-Gesichtspunkten und der vertrauenswürdigkeit der Umgebung erfolgen[9].

### 5.4.1 Verarbeitung einer Anfrage

[11] Der Zentrale Punkt, der in der der Thesis anhängenden Laborumgebung vermittelt werden soll, ist die Verarbeitung von Daten durch eine WAF. In dem folgend Anschnitt wird beschreiben wie eine WAF konzeptionell implementiert ist um diese Aufgabe auszuführen. Da die meisten Kommerziell erhältlichen WAFs proprietäre Anwendungen sind, die keine Einsicht auf den Quellcode ermöglichen, basiert diese Beschreibung hauptsächlich auf der, als *Open-Source* Anwendung vertriebenen WAF, *ModSecurity*. Da jedoch ein großer Teil der kommerziellen Anwendungen auf *ModSecurity* aufbauen lässt sich eine gewisse Allgemeingültigkeit vermuten.
Die Verarbeitung in einer WAF erfolgt in vier Schritten. Der Fokus der Beschreibung liegt auf der Verarbeitung eines HTTP Requests.

**Request Parsing** Das Request Parsing ist der erste Schritt nach dem Eintreffen einer Nachricht an einer WAF. Vor diesem Schritt ist etwaige Verschlüsselung schon geöffnet, die Nachricht liegt in HTTP-Form vor. Die Felder der HTTP-Nachricht werden nun ausgelesen und in ein Format überführt, das eine standardisierte Verarbeitung durch die WAF ermöglicht. In dieser Form besteht jedoch immer noch eine gewisse Ambiguität in der Nachricht, die in einem Angriff genutzt werden könnte. Deswegen muss einen Normalisierung der Felder erfolgen, In Abbildung 6 sind die Charakteristiken die Normalisiert werden schematisch dargestellt.
Normalisiert werden muss zum Einen die Zeichendarstellung. Die Nachricht muss in ein einheitliches Encoding überführt werden, außerdem müssen *URL-Encodings* aufgelöst werden. Hier werden Sonderzeichen mit drei bytes dargestellt: Das erste Zeichen ist immer ein % gefolgt von einer Zahl in hexadezimaler Darstellung die das Zeichen repräsentiert. Das @-Zeichen wird von %40 repräsentiert. Soll das Zeichen % selber dargestellt werden muss dies mit %25 codiert werden. Weitere Normalisierungsoperationen sind das Entfernen von *Null Bytes*. Ein *Null Byte* ist kein darstellbares Zeichen, das aber jeh nach Umfeld Nebeneffekte haben kann. Alle oben genannten Techniken können genutzt werden um nicht von regulären Ausdrücken erkannt werden zu können.
Eine weitere Gruppe von Normalisierungsoperationen sind Pfad-Normalisierungen. Sollen Pfade nicht Normalisiert werden kann ein Angreifer Zugriff auf Verzeichnisse erlangen, die nicht zugänglich sein sollten.

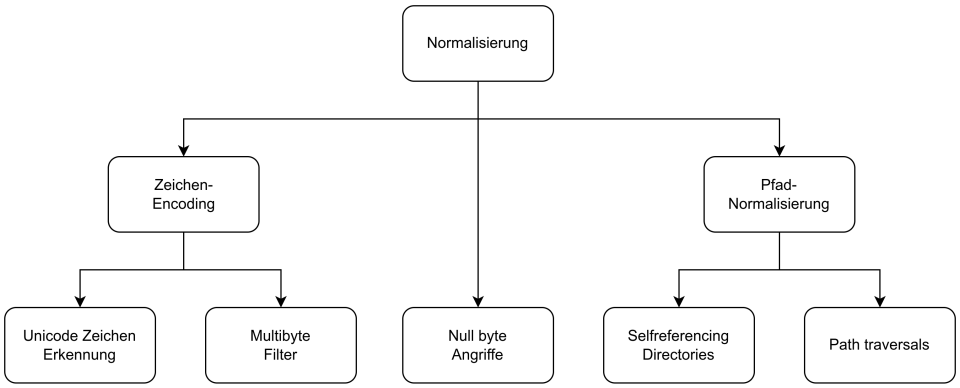



Abbildung 6: Normalisierungsoperationen

Quelle: In Anlehnung an [9, S. 6]

Ist der Request Parsing Schritt abgeschlossen, liegt einen Nachricht in einer Form vor, die eine Einheitliche Analyse ermöglicht.

**Muster-Abgleich gegen Regeln** Die normalisierte Nachricht kann von der WAF nun auf schädliche Inhalte untersucht werden. In der Regel erfolgt das durch den Abgleich gegen eine Regel. Eine Regel besteht aus einem regulären Ausdruck und einem Set an Instruktionen wie mit der Nachricht, im Fall einer Übereinstimmung mit dem regulären Ausdruck, verfahren werden soll. Es ist möglich Nachrichten zu transformieren, beispielsweise können HTTP-header hinzugefügt oder modifiziert werden. Außerdem wird das weitere Verfahren definiert.

**drop** lehnt die Nachricht ab, es werden keine weiteren Überprüfungen durchgeführt und die Nachricht nicht weitergeleitet.

**pass** gibt die Nachricht weiter ohne weitere Überprüfungen durchzurühren. Ist eine Konfiguration nicht sorgfältig überprüft kann dies zu Einschränkungen in der Sicherheit einer WAF führen.

In den Operationen können auch weitere Matching-Operationen definiert werden an die die Nachricht weitergeleitet werden soll. So kann die Menge an Regeln reduziert werden gegen die eine Nachricht überprüft werden muss. Es wird ein Baum an Regeln aufgebaut und nur die für eine Nachricht relevanten Checks ausgeführt. So muss beispielsweise nur dann wenn der HTTP-Header der signalisiert, dass der HTTP-Body JSON formatiert ist, eine Überprüfung durchgeführt werden ob es sich um gültiges JSON handelt.

Das Regelwerk einer WAF kann entweder durch *White-* oder *Blacklisting* erfolgen. Whitelisting ermöglicht es genau zu kontrollieren welche Anfragen an einen Server gestellt werden könne. Der daraus resultierende Konfigurationsaufwand für eine*n WAF-Consultant*in ist jedoch im Vergleich zu Blacklisting höher. Blacklisting hingegen ermöglicht bei unsicherer Konfiguration, dass das Umgehen der WAF deutlich einfacher möglich ist.

Neben Regelbasiertem erkennen schädlicher Inhalte bieten einige WAFs Signatur basierte Verfahren an wie sie beispielsweise auc in einem Virenscanner genutzt werden. Diese werden im Rahmen dieser Thesis jedoch nicht genauer betrachtet.

**Logging** Ein Weiterer Schritt in der Verarbeitung einer Nachricht ist das Logging. Dies ist zur Nachvollziehbarkeit der Operation einer WAF wichtig und ermöglicht es die Qualität der Konfiguration der WAF zu beurteilen. Aber auch nach einem erfolgreichen Angriff nachvollziehen wie dieser erfolgt ist und eventuell Schlüsse auf den Verursacher zuzulassen.

**Weiteres Vorgehen** Ist die Bearbeitung durch die WAF erfolgt muss die Nachricht wieder in ein verständliches HTTP überführt werden. Etwaige Änderungen an der Nachricht die durch die WAF durchgeführt werden müssen übernommen werden. Des Weiteren muss etwaige Transportsicherheit auf dem Weg zum Ziel wieder angewendet werden.

Wird eine Nachricht abgelehnt, kann in deren Statt eine Antwort der WAF an den Client gesendet werden die ihn über den Vorgang informiert. Beispielsweise kann eine HTTP-Nachricht mit einem 400er Status und der Information, dass der Client nicht die Berechtigung hat eine Aktion durchzuführen.

# 6 Design und Umsetzung der Lernumgebung

Dieses Kapitel beinhaltet die Beschreibung der Laborumgebung deren Erstellung eines der Hauptziele dieser Thesis ist. Es werden zuerst Abwägungen zu den Inhalten, die übermittelt werden sollen, angestellt. Des Weiteren werden Produkte evaluiert die für die Realisierung der Laborumgebung genutzt werden können und deren Nutzung in der Laborumgebung beschrieben. Es wird sowohl eine WAF als auch eine Anwendung benötigt, die zu schützende Schwachstellen aufweist. Final werden die erstellten lerneinheiten beschrieben und Abwägungen angestellt wie die Lerneinheiten die erarbeiteten Leerinhalte übermitteln können.

## 6.1 Zielsetzungen und grundlegende Überlegungen

Der Fokus der Lerneinheiten soll auf der Funktion einer WAF liegen. Die Lerneinheiten sind nicht geeignet um das Aufgabenfeld von WAF-Consultants zu vermitteln. Der Fokus liegt auf dem Erkennen, Verstehen und abwehren von Cyber-Angriffen. Im Betrieb einer WAF wird das hierfür notwendige Regelwerk mit den Produkten vorkonfiguriert ausgeliefert. Die Aufgaben in diesem Fall sind die Reaktion und Adaption der WAF auf entstehende Fehler um die Nutzung der zu schütztenden Webseite ohne Funktions-Einschränkungen zu ermöglichen.
In der Laborumgebung sollen diese vorkonfigurierten Regeln von den Lernenden erarbeitet werden. Die Entstehende Konfiguration ist höchst spezialisiert und kann in einem realen Szenario nicht ansatzweise Sicherheitsvorteile erbringen. Die bei einer Produktiven-WAF mitgelieferten Regelwerke werden von Mathematikern oder Theoretischen Informatikern erstellt und sind deutlich Allgemeingültiger als diejenigen die in dieser Laborumgebung erarbeitet werden.

Die Lerneinheiten sind angelegt begleitend zu theoretischem Unterricht durchgeführt zu werden. Parallel zu der Durchführung sollen die Lernenden das Wissen erhalten welches in den Grundlagen-Kapiteln (Kapitel 5) beschrieben ist. Die Laborumgebung ist nicht dafür ausgelegt dies theoretischen Grundlagen zu vermitteln und stützt sich zu Teilen auf dieses Wissen. Die Lerneinheit soll in einer Vorlesung mit 4 SWS innerhalb dreier Wochen durchführbar sein. Das Heißt es werden pro Lerneinheit 3 bis 5 Stunden Zeitaufwand angesetzt.
Die Laborumgebung besteht aus drei, aufeinander aufbauenden Lerneinheiten. In einem erste Schritt soll, nachdem sich mit dem Aufbau der Laborumgebung vertraut gemacht wird, die generelle Position einer WAF im Netzwerkverkehr beschrieben werden. Es sollen unkompliziert zugängliche Inhalte des HTTP-Protokolls analysiert werden um den generellen Aufbau einer einer Regel und die schritte der Verarbeitung in einer WAF zu Verstehen.

Die Zweite Lerneinheit beschäftigt sich mit grundlegenden Angriffen die von einer WAF abgefangen werden können. Hierfür werden einige grundlegende Schwachstellen herangezogen die sowohl in der Realität auftreten als auch ohne Vorkenntnisse mit Sicherheitslücken in Webanwendung verständlich sind. Die benötigten Vorkenntnisse sollen nur im Bereich der Software-Entwicklung und der Erstellung von Webseiten sowie einem Grundverständnis des HTTP-Protokolls liegen. Anhand dieser Kenntnisse und der Beschreibung von Schwachstellen in der zu schützenden Webanwendung soll ich ein Verständnis der Angriffsvektoren erarbeitet werden und ein Regelwerk erstellt werden, das diese abdeckt und die Ausnutzung vereitelt.
Um neben dem Regelwerk auch einen Einblick in den betrieb einer WAF zu vermittel fokussiert sich die dritte Lerneinheit darauf mit einer WAF im alltäglichen Betrieb zu arbeiten. Hier sollen die Lernenden mit einer Fehlerhaften WAF-Regel Konfrontiert werden, die das rechtmäßige Funktionieren der Webanwendung hinter der WAF beschränkt. Sie sollen diese Regel analysieren, mit den Folgen für die Netzwerkkommunikation beschäftigen und die Regeln adaptieren, sodas die Funktion der Webseite wiederhergestellt werden kann. Des Weiteren soll die dritte Lerneinheit sich mit Techniken der Filter-evasion beschäftigen. Hier sollen sich die Lernenden mit Techniken auseinandersetzen, die genutzt werden können um zu verhinder, dass ein Regulärer Ausdruck einen schädlichen Inhalt als solchen erkennt. Solche Techniken werden in realen Angriffsszenarien an einem Filter vorbei Angriffe zu ermöglicht. Eine WAF bietet auch für derartige Angriffe Verteidigungen.

## 6.2 Technische Umsetzung

### 6.2.1 Evaluation verfügbarer Produkte

#### Auswahl der WAF-Anwendung

Das Herzstück der Laborumgebung ist die WAF Anwendung. Mit dieser interagieren die Lernenden in der Laborumgebung am meisten. Sie wird nach einigen Kriterien Bewertet:

**Bedienbarkeit:** Um den Fokus auf den WAF-Regeln legen zu können sollten die Hürden zum Einstieg in die WAF möglichst gering sein

**Flexibilität:** Da sich die Laborumgebung von dem gewöhnlichen Einsatzgebiet einer WAF unterscheidet, muss die gewählte Anwendung leicht anpassbar sein. So ist es beispielsweise notwendig das vorkonfigurierte Regelwerk, mit dem die meisten WAFs ausgeliefert werden zu entfernen und eine eigenständige Konfiguration von Grund auf zu ermöglichen.

**Kosten:** Da die Laborumgebung in de Lehre eingesetzt werden können muss dürfen nahezu keine Kosten entstehen.

**Debugging:** Da die Lernenden die WAF-Regeln voraussichtlich nicht direkt fehlerfrei erstellen werden, muss ein einfacher Zugriff auf Debugging Logs möglich sein um die Funktion der erstellten Regeln analysieren können.

Eine unkomplizierte Möglichkeit eine Website mit einer WAF abzusichern sind die Angebote von Cloud-Providern wie *CloudFlare* oder *Microsoft Azure*. Beide ermöglichen es Nutzern mittel weniger Klicks eine einfache WAF vor ihrer öffentlichen Webseite zu Platzieren. Für Studierenden sind diese Angebote Außerdem kostenfrei nutzbar. Jedoch bieten diese einfachen *Cloud-WAFs* nahezu keine Möglichkeit der Konfiguration oder Anpassbarkeit. Deshalb eigenen sich derartige Angebote nicht für die Laborumgebung.

Während grundlegende Cloud-WAFs wie sie oben beschreiben sind hauptsächlich für Privatnutzer bestimmt sind, werden in Industriellen und Produktiven Umgebungen eigenständige WAFs genutzt. Im Ramen stehen zur evaluation die Produkte *Airlock* der Schweizer Firma *Ergon* sowie die *Barracuda WAF* der US Amerikanischen Firma *Barracuda Networks* zur Verfügung. Die Unterschiede der beiden Produkte liegen hauptsächlich in Performance, erweiterten Funktionen und Kosmetik. Alles Kriterien die für die Laborumgebung nicht relevant sind.
Vorteile finden sich jedoch in der Nutzbarkeit. Die Grafische Oberfläche ermöglicht eine flache Lernkurve und schnellen Einstieg in die kerninhalte der Laborumgebung. Außerdem werden beide Produkte mit Log-Analyse Programmen ausgeliefert die es Nutzern mittels grafischer Interfaces ermöglicht die Logs der WAF zu analysieren.
Jedoch können diese kommerziellen Produkte in der Laborumgebung nicht sinnvoll eingesetzt werden: Die Kosten belaufen sich auf mehrere Tausend Euro pro Instanz, die auch nur über Vertriebspartner erhalten werden könne. Des Weiteren werden Sicherheitskritische Anwendungen wie eine WAF üblicherweise mit einem gehärteten Betriebssystem ausgeliefert. Dadurch ist ein Zugriff auf die tieferen Ebenen der Anwendung nicht möglich. Auch lässt sich die mitgelieferte Konfiguration nicht aus der Anwendung entfernen. Zwar lassen sich einzelne Regeln deaktivieren, eine völlig Konfigurationslose Auslieferung ist jedoch nicht möglich. Auch sind die Anwendungen schon mit ihren minimalen Hardwareanforderungen relativ Leistungshungrig und würden die Hardware gängiger privatrechner fast vollständig ausnutzen. Aus diesen Gründen ist es nicht möglich diese Produkte in der Laborumgebung einzusetzen.

Aus diesem Grund fällt der Fokus auf *Open Source* WAFs. Auch in diesem Feld gibt es einige Anbieter. Jedoch bauen die meisten auf den beiden Projekten *Mod_defender* oder *ModSecurity*. Auch große Software Hersteller wie zum Beispiel Microsoft basie-

ren ihre WAFs auf diesen Anwendungen. Beide werden als Module des Apache Webservers betrieben und existieren in diversen Distributionen: Als VM, Docker Container oder alleinstehende Anwendung. Des weiteren kann bei beiden WAFs problemlos sämtliche Konfiguration entfernt werden.
Ein großer Nachteil der *Open Source* WAFs ist die Konfiguration auf basis von Textdateien und das Fehlen von Grafischen Oberflächen zur Konfiguration. Dies kann im Einstieg in die Lerneinheiten für mehr Lernaufwand sorgen. Jedoch muss sich dadurch auf einer tieferen Ebenen mit der Funktion einer WAF beschäftigt werden. Dies kann als positives für die Lernerfahrung gewertet werden.

Für die Laborumgebung wird die *ModSecurity* waf in einer Distributionen der OWASP Foundation mit dem Namen *ModSecurity - Core Rule Set*. Die unterschiedlichen Distributionen unterscheiden sich zwar kaum, jedoch ist die Dokumentation der gewählten Anwendung sowie die Aktivität der Community am ansprechendsten, was den Lernenden den Einstieg möglichst einfach gestaltet.

#### Verwundbare Anwendungen

Neben einer WAF muss in der Laborumgebung eine Webanwendung zu Verfügung steht. In dieser Webanwendung sollen Schwachstellen bestehen, die mittels der WAF abgesichert werden können. Es ist von Vorteil, wenn die verwundbare Anwendung möglichst einfach zu nutzen ist und die Schwachstellen Verständlich dokumentiert sind.
Eine Möglichkeit ist, im Rahmen der Thesis eine eine Webanwendung zu erstellen, die genau auf die Aufgaben abgestimmt ist. Dies würde das Stellen der Aufgaben vereinfachen und den Größten freiraum im Erstellen der Lerneinheiten bieten. Die Schwachstelle und zugehörige Lösung sind direkt aufeinander abgestimmt. Außerdem kann den Lernenden direkt von der verwundbaren Anwendung rückmeldung gegeben werden ob die erstellte Konfiguration der WAF erfolgreich war. Es steht im Rahmen der Thesis jedoch nicht genug Zeit zur Verfügung um eine solche verwundbare Webanwendung zu Erstellen.
Aus diesem Grund fällt die Wahl auf eine der Öffentlich verfügbaren verwundbaren Anwendungen. Hier gibt es einige Angebote die sich für den Zweck der Thesis alle eignen. Die meisten Angebote werden ohne Lizenzkosten als *Open Source* Anwendung zir Verfügung gestellt. Dies vereinfacht die Nutzung in der Lehre ohne Einschränkungen. Ein weiterer Vorteil der sich daraus ergibt ist die einfache Modifizierbarkeit: Um die Laborumgebung Kontrollierbar zu machen können beispielsweise zusätzliche Daten in der Anwendung hinterlegt werden um *edge-cases* in der Konfiguration der Lernenden automatisiert überprüfen zu können.
Die *Open Source* Natur der Webanwendung hat Außerdem ein einfaches Deployment

zur Folge. Ob von den Projekten selber oder von Dritten wird eine weite Auswahl an Deployment-Methoden bereitgestellt. Für die meisten der verwundbaren Anwendungen lassen sich Distributionen als Docker-Container, VM oder alleinstehende Anwendung finden. Dies ermöglicht einen große Auswahl an Möglichkeiten wie die Laborumgebung gestaltet werden kann.
Final fällt die Entscheidung auf der *Juice Shop* der OWASP-Foundation. Die Anwendung bietet die Gleichen Vorteile wie die meisten anderen verwundbaren Anwendungen. Jedoch besteht ein Vorteil: Die Funktion des *Juice Shops* und einige seiner Schwachstellen sind den Lernenden aus einer vorherigen Vorlesung schon bekannt. Somit ist die Einstiegshürde für Lernende geringer und der Fokus kann einfacher auf die WAF und das Erstellen eines Regelwerks gelegt werden.

### 6.2.2 Labor-Umgebung

Die in Kapitel 6.1 beschriebenen Inhalte sollen in einem Praxisnahen Umfeld vermittelt werden. Dazu kommt nach den Abwägungen aus Kapitel 6.2.1, die WAF ModSecurity zum Einsatz. Die zu diesem Zweck vorgesehene Laborumgebung muss einige Kriterien erfüllen:

**Einheitliches Deployment:** Der Ausgangspunkt der Lerneinheiten muss reproduzierbar und wiederholbar sein. Bei wiederholten Durchführungen der Übungen soll es einfach sein den Lernenden ohne zusätzlichem manuellen Konfigurationsaufwand eine Laborumgebung zu übergeben. Diese Laborumgebung muss Platformunabhängig aufgebaut sein und auf Windows, MacOS unf Linux genutzt werden können.

**Modifizierbarkeit der Anwendungen:** Um in den Lerneinheiten grundlegende Techniken zu übermitteln, ist es notwendig vorkonfigurierte Funktion der ModSecurity WAF entfernen zu können und den JuiceShop mit Daten zu versehen um eine automatisierte Überprüfung zuu ermöglichen. Dies muss automatisiert und auf einem einheitlichen Weg erfolgen können.

**Bekannte Technologien:** Der Fokus der Lerneinheiten liegt auf dem Erlernen der Technik und Funktion einer WAF. Um einen Einstieg möglichst direkt zu gestalten sollen hierfür Technologien zum Einsatz kommen, die den Lernenden schon bekannt sind und keinen zusätzlichen Lernaufwand erzeugen.

**Komplexe Netzwerkumgebungen:** Da die verschiedenen Anwendungen in der Laborumgebung über Netzwerkkommunikation miteinander kommunizieren, muss es möglich sein automatisiert virtuelle Netzwerke zu erzeugen.

Um den oben genannten Anforderungen möglichst genau zu entsprechen, wird als Teil der Thesis die in Abbildung 7 schematisch dargestellte Laborumgebung erstellt.

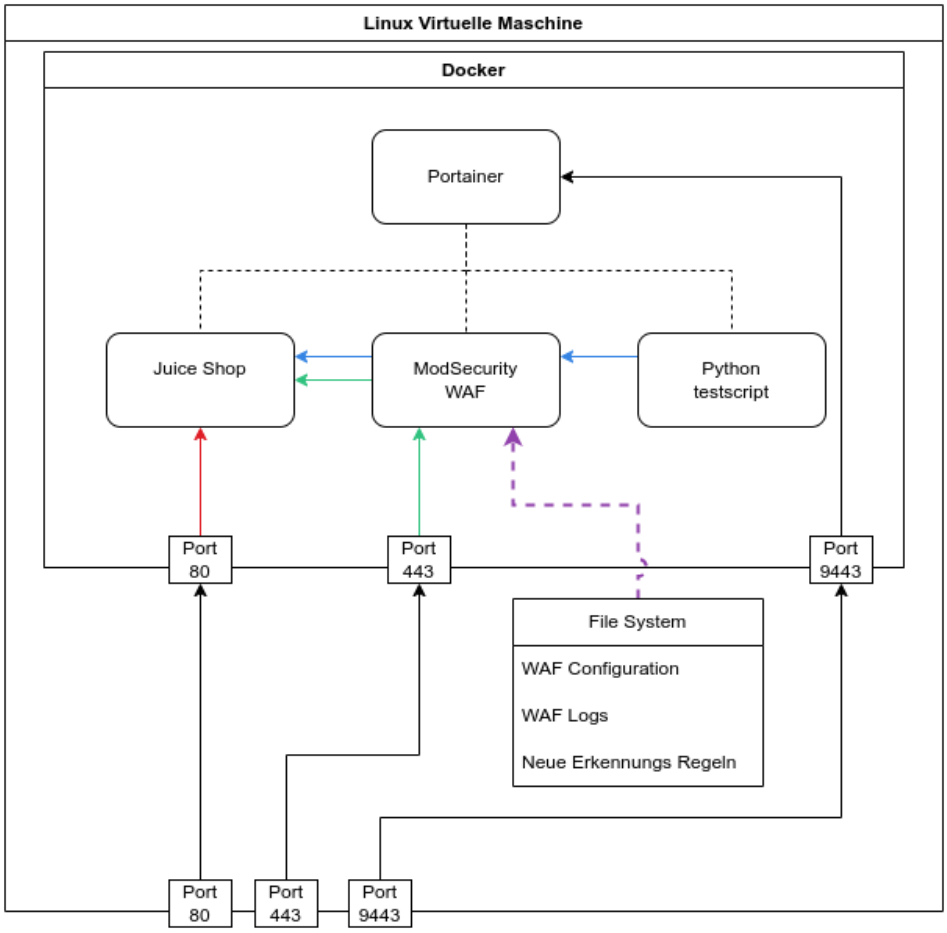



Abbildung 7: Aufbau der Laborumgebung

Quelle: Eigene Darstellung

Die Containervirtualisierungsumgebung Docker wird als Deployment-Umgebung verwendet. Diese ermöglicht es isolierte Ausführungsumgebungen für die Anwendungen zu erzeugen. Der Aufbau dieser Umgebungen lässt sich mittels einer Konfigurationsdatei genau beschreiben und wiederholbar ausrollen. Innerhalb der Umgebungen lassen sich virtuelle Netzwerke anlegen um Netzwerkkommunikation vom Host zu isolieren. Dadurch lässt sich festlegen, wie und welche der einzelnen Anwendungen (Container) untereinander kommunizieren können. Außerdem können Dateien oder Ordner aus dem Host-Dateisystem in den Container übergeben werden. Dies ermöglicht es Konfigurationsdateien zur Verfügung zu stellen um die Container zu modifizieren, die sonst *Stateless* sind und nicht vorkonfiguriert ausgeliefert werden. Im

Gegensatz zu virtuellen Maschinen greift die Containervirtualisierungsumgebung direkt auf die Mittel des Host-Betriebssystem zu und benötigt dadurch deutlich weniger Rechenaufwand.

Wie in Abbildung 7 dargestellt, werden in der Laborumgebung vier Container betrieben:

**ModSecurity WAF:** Der Hersteller, der in der Laborumgebung verwendeten WAF: *ModSecurity*, stellt sein Produkt auch in Form eines Docker-Containers zur Verfügung. Diese wird jedoch in modifizierter Form genutzt. Um die Lerninhalte zu vermittel muss die große Anzahl vorkonfigurierter Regeln, mit denen die WAF ausgeliefert wird um den Betrieb zu ermöglichen, entfernt werden. An dessen Stelle wird ein Verzeichnis aus dem Host-Dateisystem durchgereicht, in dem die Lernenden eigene Konfigurationen Platzieren können. Daneben werden, um das Debugging zu ermöglichen, die Log-Dateien aus der WAF im Host-Betriebssystem zur Verfügung gestellt.

**Juice Shop:** Dieser wird in Version 16.0.0 verwendet, da der Hersteller (OWASP) die Anwendung regelmäßig verändert. Durch die Verwendung der neuesten Version könnten Challenges verloren gehen, die für die Durchführung der Aufgaben notwendig wäre. Auch diese Anwendung wird modifiziert. Es werden Daten hinterlegt und Nutzeraccounts angelegt. Diese ermöglichen es mittels eines automatischen Kontroll-Scripts die Konfiguration der WAF zu überprüfen. Im Rahmen der Thesis war es aus Zeitlichen Gründen nicht möglich das Kontroll-Script so fertigzustellen, dass es zuverlässig Ergebnisse liefert.

**Python Test-Script:** Dieser Container enthält ein Python Skript, das es den Lernenden mittels des Unittest-Frameworks *Pytest* ermöglicht, die erarbeiteten Lösungen zu überprüfen. Das Skript schickt HTTP-Requests durch die WAF und evaluiert die Antworten, um den Lernenden Rückmeldung über den Erfolg ihrer Konfiguration zu geben. Im Rahmen der Thesis war es aus Zeitlichen Gründen nicht möglich das Kontroll-Script so fertigzustellen, dass es zuverlässig Ergebnisse liefert.

**Portainer:** Die Anwendung *Portainer* ermöglicht es eine Docker Umgebung mittels einer Grafischen Oberfläche zu verwalten. In der Laborumgebung kann sie genutzt werden um die Laborumgebung zu bedienen ohne sich tiefer mit der Funktion von Docker auseinander setzen zu müssen. Zwar können die Lernenden dies auch über das Docker Kommandozeilen-Interface tun, jedoch wird dies als eine vermeidbare Hürde betrachtet, die den Einstieg erschweren könnte. Die grafische Oberfläche soll unter anderem genutzt werden um den WAF-Container

nach einer Konfigurationsänderung neu zu starten und mit dem Test Skript zu interagieren.

Aus den oben genannten Containern ergeben sich einige Netze, die im Hintergrund existieren müssen. So ist es notwendig, dass eine Verbindung von dem Python-Test-Container zur WAF und von dieser zum Juice Shop aufgebaut werden kann. Hierfür werden separate Docker-Netzwerke erstellt die an den Containern angeschlossen sind. Um den Nutzern eine Interaktion mit den Containern zu ermöglichen werden einige Ports aus der Docker-Umgebung freigegeben:

- Die ungesicherte Weboberfläche des Juice Shops (Port 80 [HTTP])
- Die, durch die WAF gesicherte, Weboberfläche des Juice Shops (Port 443 [HTTPS])
- Das Management Interface der Portainer-Anwendung (Port 9443 [HTTPS])

Durch die oben beschriebene Docker Umgebung sind Anforderungen an die Laborumgebung wie dem *einheitlichen Deployment* und der *Modifizierbarkeit der Anwendungen* bereits erfüllt. Es ergeben sich jedoch auch einige Herausforderungen:

- Docker ist zwar als Cross-Platform Anwendung konzipiert. Es stehen Versionen für die drei gängigen Betriebssysteme Windows, MacOS und Linux zur Verfügung. Jedoch bauen die verwendeten Container hauptsächlich auf Linux auf. In der Theorie sollte dies zu keinen Problemen führen, da Docker in der Lage ist nicht Platform-Native Container auf sich unterscheidenden Betriebssystemen auszuführen, jedoch kann ein solcher Aufbau durchaus zu unvorhergesehenen Problemen führen.
- Ein weiteres Problem ist, dass durch das Durchreichen von Dateien zwischen Container und dem Host-Dateisystem zusätzlicher Konfigurationsaufwand für die Nutzer entsteht.

Um diese Probleme zu mittigieren wird die Laborumgebung als Linux Virtuelle Maschine ausgeliefert. In dieser ist eine Docker-Umgebung vorinstalliert und die Container und Netzwerke bereits präsent und werde automatisiert gestartet. Dies ermöglicht die Auslieferung mittels einer VM-Datei, in der die Konfigurationen schon an einer einheitlichen Stelle enthalten sind. Lernende müssen zur Nutzung also nur eine Virtuelle Maschinen auf ihren Rechnern importieren und mittels eines Virtuellen Netzwerk Interface auf die Weboberflächen zugreifen. Die Konfiguration der WAF findet in Textdateien statt, die sich in der Virtuellen Maschine befinden. Der Zugriff auf diese ist mit dem Text-Editor Visual Studio Code der Firma Microsoft vorgesehen, da dieser eine SSH Erweiterung hat die es mit geringen Konfigurationsaufwand ermöglicht Dateien auf entfernten Servern oder in Virtuellen Maschinen zu bearbeiten.

Die Nutzung der Laborumgebung werden durch diese Maßnahmen als einfach genug beurteilt um einen schnellen Einstieg zu ermöglichen. Die Konfigurationen, die vorgenommen werden müssen, werden in der ersten Lerneinheit (Kapitel A) beschrieben. Es steht den Lernenden frei weitere oder andere als die beschriebenen Technologien zu verwenden, um mit der Laborumgebung zu interagieren. Diese können im Rahmen dieser Thesis und den Aufgabenstellungen jedoch nicht berücksichtigt werden.

## 6.3 Lerneinheiten

Die in diesem Kapitel beschriebenen Lerneinheiten machen einen zentralen Bestandteil dieser Thesis aus.Sie ähneln sich in Aufbau und Durchführung, beschäftigen sich jedoch mit unterschiedlichen Aspekten einer WAF: Eine Aufgabe in einer Lerneinheit beginnt immer mit einigen Worten zu dem Thema mit dem sich die Lernenden beschäftigen sollen, gefolgt von einer Referenz auf eine Challenge aus dem Juice Shop die mit der WAF abgesichert werden soll.
In diesem Kapitel wird beschrieben welche Überlegungen in die Konzeption der Lerneinheit geflossen sind. Mit welchem Überthema sich die jeweiligen Lerneinheiten beschäftigen, nach welchen Kriterien die Challenges des Juice Shops ausgewählt wurden und wie die Challenges gelöst werden können.

### 6.3.1 Teil 1: Erster Kontakt zu einer WAF

In der Ersten Lerneinheit sollen sich die Lernenden mit der Laborumgebung vertraut machen. Erst sollen die Lernenden die Laborumgebung auf ihren Geräten einrichten und sich versichern, dass sie Zugriff auf alle Services der Umgebung haben. Dann gilt es ein Verständnis aufzubauen wie der Fluss einer HTTP-Anfrage durch die Netzwerke der Umgebung erfolgt. Den Lebenden sollte vor Beginn der Lerneinheit bekannt sein wie eine WAF in der Theorie vor eine Webanwendung platziert werden kann.

Die Aufgabenstellungen zu der hier beschreiben Lerneinheit sind in Anhang A zu finden.

#### Einrichtung der Laborumgebung

In der ersten Aufgabe (Anhang A.1) der Lerneinheit sollen die Lernenden die Laborumgebung installieren. Hierfür muss die Laborumgebungs VM bereit gestellt werden. Die weiteren Anwendungen, die im Zuge der Durchführung benötigt werden, werden in der Aufgabenstellung genannt und die Konfiguration erläutert. Die Aufgabe besteht aus einer detailliert bebilderten Anleitung zum Aufsetzern der Laborumgebung. Die Bestandteile sind:

1. Installation der VM
2. Konfiguration eines virtuellen *Host-Only* Netzwerkadapters um vom Host Betriebssystem auf die WAF-Laborumgebung zugreifen zu können.
3. Abrufen der Verfügbaren Webanwendungen

4. Verbinden mit der SSH-Schnittstelle um die WAF Konfiguration bearbeiten zu können.

Die Anleitung ist mit dem Fokus auf einige vorgeschrieben Technologien verfasst. Es wird jedoch auch auf einer konzeptionellen beschreiben was zu tun ist. Lernende, die andere als die beschrieben Technologien nutzen möchten können dies tun, dies ist in der Konzeption der Laborumgebung berücksichtigt, wird in der Anleitung aber nicht genauer beschrieben. Lernende die sich dafür entscheiden müssen auftretende Probleme mit den von ihnen gewählten Technologien eigenständig und ohne Hilfe der Anleitung lösen.
Nach Abschluss der Aufgabe sollen die Lernenden in der Lage sein die Laborumgebung zu nutzen um weitere Aufgaben lösen zu können. Die Lernenden müssen sich nur in geringem Maße neues Wissen erarbeiten, da die Anleitung sehr eng geführt ist.

**Erste Arbeit mit der WAF**

In dieser Aufgabe in der sich die Lernenden das erste Mal mit WAF-Regeln beschäftigen. Für den Einstieg soll sich zuerst mit der Syntax der *ModSecurity* Regeln anhand einfacher Beispiele beschäftigt werden. Hierfür werden Zwei Challenges des *Juice Shops* herangezogen deren Lösungen ohne größere Probleme nachvollziehbar sind. In der Ersten dieser Aufgaben soll ein Pfad unzugänglich gemacht werden, auf dem sensitive Informationen preisgegeben werden. Die hierfür notwendige Regeln ist verhältnismäßig unkompliziert und kann beispielsweise wie Folgt aussehen:

```
SecRule REQUEST_URI "@streq /ftp/acquisitions.md" \
    "id:1, \
    msg:'Stop access to critical /ftp/aquisitions.md', \
    deny, \
    status:403"
```

Der zweite Teil der Aufgabe erfordert es den Inhalt eines HTTP-Bodies zu Parsen und zu analysieren. Die zugehörige Challenge im *Juice Shop* weist eine Schwachstelle auf in der Parameter im Backend nicht auf Gültigkeit geprüft werden. Dadurch können ungültig Daten an den Webserver übergeben werden, die dort gespeichert und von dort an Nutzer übergeben werden. Der Parameter *rating*, der in einem JSON-Objekt im HTTP-Request Body transportiert wird soll nur die Werte 1 bis 5 annehmen können.
Der schädliche Parameter kann auf mehreren wegen erkannt werden. Zwei Beispiele wären:

Mithilfe eines regulären Ausdrucks ($[\wedge 1-5]$):

```
SecRule REQUEST_HEADERS:Content-Type "application/json" \
    "phase:2, id:3, t:none, t:lowercase, \
    log, ctl:requestBodyProcessor=JSON" \
    chain, \
    msg:'JSON parameter rating found in request body'"

    SecRule REQUEST_BODY:rating "@rx [^1-5]" \
        "id:4, phase:2, \
        deny, log, status:403"
```

Mithilfe von Vergleichsoperationen ($@gt5$ und $@lt1$):

```
SecRule REQUEST_HEADERS:Content-Type "application/json" \
    "phase:2, id:3, t:none, t:lowercase, \
    log, ctl:requestBodyProcessor=JSON" \
    chain, \
    msg:'JSON parameter rating found in request body'"

    SecRule ARGS:rating "@gt 5" \
        "id:4, phase:2, \
        deny, log, status:403"

    SecRule ARGS:rating "@lt 1" \
        "id:5, phase:2, \
        deny, log, status:403"
```

Diese Challenges sollen den Lernenden grundlegende Konzepte zum Aufbau einer *ModSecurity* Regel beibringen:

- Den grundlegenden Aufbau nach der Struktur

  ```
  SecRule VARIABLE [OPERATOR] [TRANSFORMATIONS,ACTIONS]
  ```

  in der erst das Ziel der Operation (VARIABLE), dann die Matching-Operation (OPERATOR) und dann das Verfahren nach dem *Match* (TRANSFORMATIONS,ACTIONS) beschrieben werden.

- Das Prinzip des *Multi Stage Processing*: Nicht die Ganze HTTP-Nachricht steht zur gleichen Zeit zur Verfügung. Der Zugriff auf ein JSON-Body element kann, wie in dem Beispiel oben, also erst nach vorherigem Parsing betrachtet werden.

Die beiden oben beschrieben Aufgaben sind gedacht um die WAF kennen zu lernen. Sie bilden jedoch kein Verhalten ab das in einer WAF-Regel einer produktiven WAF verwendet werden würde: Die Regeln sind viel zu detailliert und auf einen spezifischen Fall ausgerichtet.

### 6.3.2 Teil 2: Grundlegende Angriffe

Nachdem sich in der ersten Lerneinheit mit der Laborumgebung, der WAF und ihrer Funktion vertraut gemacht wurde, kann der Fokus in dieser zweiten Lerneinheit auf Angriffsszenarien gelegt werden. Die Lernenden beschäftigen sich exemplarisch mit zwei Angriffsszenarien, die in der aktuellen IT-Sicherheit und Verteidigung von Webanwendungen relevant sind.

Die Aufgabenstellungen zu der hier beschreiben Lerneinheit sind in Anhang B zu finden.

**SQL Injections**
Die SQL-Injection ist eine Angriffstechnik bei der Nutzereingaben direkt und ungefiltert an eine SQL-Datenbank im Backend weitergegeben werden. Dies kann zu unvorhergesehenen nebeneffekten wie Befehlsausführung auf dem Server und Daten-Exhilaration führen. Nach OWASP-top-ten gehören SQL Injections zu den zehn schwerwiegendsten Schwachstellen, die in Webanwendungen zu finden sind.
Die Funktion und Verteidigung gegen SQL Injection Angriffe sollen Lernenden in dieser Aufgabe erarbeiten. Zuerst soll sich mit einer spezifischen Sicherheitslücke beschäftigt werden und diese mit einer eigenständigen WAF-Regel mittigiert werden. In einem zweiten Schritt wird einen SQL-Injection generell betrachtet. Hierfür sollen die Lernenden die Gemeinsamkeiten von zwei *Juice Shop* Challenges herausarbeiten und beide Schwachstellen mit einer Regel mittigieren.
Die erste Teilaufgabe beschäftigt sich mit einem SQL Login-Bypass. Zum Abgleich ob ein Nutzer existiert und das Passwort gültig ist wird im Backend die SQL-Datenbank genutzt. Ein Angreifer kann in dieser *Juice Shop* Challenge an den Nutzernamen einen Vergleich in SQL Syntax anhängen und somit einen gültigen Login erzwingen ohne das Passwort eines Nutzers zu kennen.
Um diesen Angriff erkennen zu können sollen sich die Lernenden auf escape syntax fokussieren, mit deren Hilfe ein SQL-Befehl beendet werden kann um danach eigene Befehle auszuführen.
zwei Beispiel2 könnten wie Folgt aussehen:

```
SecRule ARGS:email "@rx ^['].*--" \
    "id:6, phase:2, \
    deny,log,status:401"
```

```
SecRule ARGS:email "@rx ==\w*--" \
    "id:7, phase:2, \
    deny,log,status:401"
```

In beiden Fällen wird nach dem Beginn eines SQL Kommentars gesucht. Die beiden dargestellten Möglichkeiten suchen vor dem Kommentar jedoch nach unterschiedlichen Angriffsmustern. Im ersten Beispiel werden Hochkommata gemacht, die in einer Mailadresse nicht vertreten sein sollten. Im zweiten Fall wird nach einem Vergleich (==) gesucht, der zwar theoretisch in einem nicht bösartigen Szenario vorkommen könnte, jedoch eher unwahrscheinlich ist und keine Einschränkung für Nutzer darstellen würde.
Im Zweiten Teil der Aufgabe soll sich nun auf das generelle filtern von SQL-Syntax fokussiert werden. Die beiden Challenges die die Lernenden analysieren sollen weisen SQL-Injection Schwachstellen im HTTP-Body auf. Damit die Lernenden keine Regeln schreiben können, die sich nur auf einen spezifischen Fall beschränkt sollen die Lernenden eine Regel für beide Challenges schreiben.
Ein Lösungsvorschlag für das generelle Erkennen einer SQL-INjection Attacke kann sein die sql keywords zu erkennen wie in dem folgenden regulären Ausdruck nachvollziehbar ist.

```
\s*(select|union|update|delete|insert|drop|--|or|and|alter|exec
|create|script|table|from|where|join|having|cast)\s*
```

Beim schreiben der Regel müssen die Lernenden darauf achten, dass die Regel keine unvorhergesehenen Nebeneffekte hat. Wird beispielsweise nur nach dem substring `and` gefiltert, könnte ein Nutzer mit dem Namen *Andy* Probleme bekommen die Webanwendung zu nutzen.

Ist die Aufgabe abgeschlossen sollen die Lernenden ein Verständnis dafür haben wie eine SQL-Injection Funktioniert. Außerdem sollen sie durch das lösen von Problemen und betrachten der Logs herausgefunden haben, dass ein regulärer Ausdruck dessen Folgen nicht bedacht wurde Probleme bei der regulären Nutzung der Webanwendung hervorrufen kann.

### Cross Site Scripting (XSS)

Neben der *SQL-Injection* nutzen Angreifer häufig sogenannte Cross Site Scripting (XSS) Schwachstellen in Webanwendungen aus. Bei einem XSS ist es einem Angreifer möglich HTML und Java Scrip Code an die Webanwendungen zu übergeben sodass dieser

von der Anwendung nicht als Text interpretiert wird sonder als Code Ausgeführt bzw. dargestellt wird.
In diesem Teil der Lerneinheit sollen sich die Lernenden mit der Funktion einer XSS Schwachstelle vertraut machen und daraus ableiten, wie ein solcher Angriff erkannt und verhindert werden kann. Des weiteren besitzen Browser Funktionen die das Ausnutzen eines XSS erschweren. Diese werden können in einem HTTP request durch das Setzen eines bestimmten Headers oder dem erstellen einer *Content Security Policy* die festlegt von welchen Quellen Daten nachgeladen werden dürfen. Dies kann auch durch die WAF erfolgen. An diesem Beispiel ist es auch möglich das Umschreiben eines HTTP-Request durch die WAF zu übermitteln.
Im ersten Teil der Aufgabe sollen die Lernenden eine WAF-Regel schreiben die in eingehendem Traffic ein HTML-Tag und javascript erkennt und Blockiert. Es existieren mehrere Möglichkeiten einen solchen Angriff zu erkennen und abzuwehren. Anhand der öffnenden und schließenden HTML-Tags oder dem *javascript* Tag. Im folgenden beispiel ist eine Möglichkeit dies anhand der HTML *script Tags* zu tun dargestellt:

```
SecRule REQUEST_HEADERS:Content-Type "application/json \
    [...]
    SecRule ARGS: "@rx <script[^>]*>.*</script>|<[^>]+>" \
    [...]
```

Für eine allgemeine Lösungen nicht hinreichende, jedoch in dem spezifischen Fall der Übung gültige Lösung wäre das erkennen des javascript codes. Das Hierfür zum Testen genutzte *javascript:alert* ist austauschbar und würde keinen zuverlässigen Schutz sorgen.
Der Zweite Teil der Aufgabe beschäftigt sich mit dem mit dem setzen des *X-XSS-Protection* Headers. Hierfür muss in einer Regel ausgehender Netzwerkverkehr erkannt werden. Anstatt diesen zu blockieren kann der Header gesetzt werde, wie im Folgenden Beispiel zu sehen ist:

```
SecRule RESPONSE_HEADERS:@streq 200 \
   "id:10, phase:3, nolog, pass,\
   hdrOut: Header set X-XSS-Protection '1; mode=block'"
```

Die Lernenden sollen nach Abschluss dieser Lerneinheit ein Verständnis für die Funktion eines XSS haben. Außerdem soll ein Bewusstsein für die Funktion der XSS-Protection Header bestehen, die einen wichtigen ersten Verteidigungsschritt gegen diese Angriffe darstellen. Neben dem Verständnis für den Angriff wird auch die Fähigkeit einer WAF den Netzwerkverkehr nicht nur zu bloc sonder auch umzuschreiben vermittelt.

### 6.3.3 Teil3: Nutzung einer WAF in produktivem Umfeld

Die dritte Lerneinheit soll einen Fokus auf der Arbeit mit einer WAF in produktivem Umfeld Legen. Die Lernenden sollen mit den Logdateien arbeiten um den Fehler in einer WAF-Regel zu finden. Außerdem ist vorgesehen, dass sich die Lernende mit Techniken der *Filter evasion* beschäftigen, die ein Angreifer Nutzen könnte um nicht von den regulären Ausdrücken einer WAF erkannt zu werden.

Im Rahmen der Thesis war es aus Zeitlichen gründen diese dritte Lerneinheit zu realisieren.

# 7 Evaluation

Um einschätzen zu können wie die Lerneinheiten in der Lehre eingesetzt werden könne wird eine Evaluation durchgeführt. Wie viel Zeit wird für die Durchführung der Übungen benötigt? Welche Vorkenntnisse sind notwendig um die Übungen erfolgreich durchführen zu können? Welches wissen erwerben die Lernenden in der Laborumgebung?
Bevor die Laborumgebung begonnen wird, soll eine Selbsteinschätzung durchgeführt werden in der die Kenntnisse zu bestimmten Technologien abgefragt werden, die in der Laborumgebung relevant sind. Die Teilnehmer sollen auf einer Skala von 1 bis 6 ihre Fähigkeiten in den folgenden Technologien bewerten:

- Linux
- Virtuelle Maschinen
- Containervirtualisierungsumgebung Docker
- Softwareentwicklung
- Reguläre Ausdrücke
- HTTP auf Protokollebene
- SQL-Syntax
- Sicherheitslücken in Webanwendung
- Web Application Firewall

Nach der Selbsteinschätzung werden die Lerneinheiten durchgeführt. Die Probanden sollen die Lerneinheiten eigenständig bearbeiten damit das Umfeld möglich nah an dem geplanten Einsatzgebiet der Laborumgebung ist. Nach der Durchführung werden die Probanden in einem direkten Gespräch befragt und die erzielten Lösungen Analysiert.
Aus den Gesammelten Informationen lässt dich abschätzen welches Vorwissen für die Laborumgebung notwendig ist ist ob die geplante Zeitvorgabe realistisch ist.

## 7.1 Evaluation mit Probanden

Die Evaluation erfolgt mit vier Probanden. Alle Teilnehmer sind im Bereich der der IT-Sicherheit tätig oder in der Ausbildung zum Fachinformatiker. Dadurch haben alle Probanden ein Verständnis für die Funktion von IT-Systemen, jedoch in in unterschiedlicher Tiefe.

Die Gruppe Lässt sich in zwei Teile aufteilen. Diejenigen, die in der vorhergegangen Befragung angegeben haben keine oder geringe Kenntnisse in den Themengebieten *Reguläre Ausdrücke* und *HTTP* waren nicht in der Lage die Aufgaben erfolgreich abzuschließen. Vorwissen in diesen beiden Themengebieten lässt sich also als Voraussetzung für die Durchführung betrachten. Es lässt sich Vermuten, dass das Gleiche für Probanden ohne Kenntnisse mit SQL Probleme bei der Durchführung haben werde. Bei alle Probanden war dieses wissen jedoch Verfügbar und diese These lässt sich nicht betätigen. Es scheint also empfehlenswert, dass Probanden mit wissen über Reguläre Ausdrücke, der Funktion von HTTP und Kenntnisse der SQL-Syntax in die Lerneinheit einsteigen.
Die andere beiden Probanden waren in der Lage die Laborumgebung ohne Einschränkung durchzuführen. Einer der Probanden, der im Bereich WAF tätig ist, war in der Lage die Laborumgebung innerhalb von drei Stunden durchzuführen. Bei dem der letzte Proband, dessen Wissen dem eines Studenten am besten entspricht, hat die Laborumgebung in einem vollen Arbeitstag abgeschlossen. Aus diesen Begrenzten Informationen lässt sich Vermuten, dass einen Durchführung der Lerneinheit im Rahmen einer Vorlesung mit eigenarbeitszeit möglich ist.

## 7.2 Überlegungen zur Bewertung der Lernenden

Als Teil der Thesis wird sich mit der Frage beschäftigt, wie eine Bewertung der Ergebnisse, die die Lernenden in der Laborumgebung erarbeiten, erfolgen kann. Kann erkannt werden ob lernende Lösungen untereinander austauschen? Es wird auch die Möglichkeit untersucht ob es möglich ist, die Laborumgebung so dynamisch zu generieren, dass die Laborumgebungen für jeden Lernenden einzigartig gestaltet ist. Dies ließe sich beispielsweise durch dynamisch generierte Netzwerk-Topologien erreichen. Einen weitere Möglichkeit wäre, jedem Lernenden unterschiedliche zu schützenden Challenges des *Juice Shops* zuzuweisen. Beide Ansätze stellen sich in der Durchführung der Thesis als nicht oder nicht innerhalb der Zeitvorgabe umsetzbar heraus.

# 8 Fazit

Der Bereich der Cybersicherheit oft als eine Einheit wahrgenommen wird besteht er heutzutage aus einer weiten Auswahl an Technologien und Spezialgebieten, die alle einen wichtigen Beitrag zum sicheren Betrieb von IT-Systemen leisten. Die Verschiedenen Felder benötigen alle qualifiziertes Fachpersonal um diese Sicherheit garantieren zu können. Aus diesem Grund soll in dieser Thesis Lerneinheiten Entwickelt werden die in der Lehre eingesetzt werden kann um die Funktion einer WAF zu vermitteln.

## 8.1 Zusammenfassung

Das Ziel dieser Arbeit ist es Lerneinheiten zu erstellen mit deren Hilfe es Lernenden möglich ist einen Einblick in die Funktion einer WAF zu vermitteln. Die Lernenden sollen, nachdem sie die Lerneinheiten bearbeitet haben, ein Verständnis dafür haben wie eine WAF im Netzwerkverkehr zwischen Server und Client platziert ist um ihre Sicherheitsfunktion zu erfüllen. Es soll ein Tiefer Einblick in die Implementierung einer WAF vermitteln werden, besonders wie es mithilfe von Regeln möglich ist Netzwerkverkehr zu analysieren und schädliche Inhalte zu erkennen. Dadurch sollen die Lernenden ein Verständnis dafür erlagen welche gängige Taktiken Angreifer nutzen um Schwachstellen in Webanwendungen auszunutzen. Daraus soll sich das Wissen erarbeitet werden wie die Verteidigung gegen diese möglich ist. Des weiteren sollen die Lernenden einen Einblick erhalten wie der Einsatz einer WAF in produktivem Umfeld erfolgen. Die Lernenden sollen erfahren wie eine WAF in einem Netzwerk platziert wird um eine Webanwendung abzusichern und welche Tätigkeiten im betrieb einer WAF notwendig sind.
Im Ramen dieser Thesis wurde eine Laborumgebung Implementiert in der Lernende Aufgaben erfüllen können. Anhand der Aufgaben wird den Lernenden aufeinander aufbauend die im vorherigen Abschnitt beschriebenen Inhalte vermittelt. Um eine Abschätzung wie Lernende mit der Laborumgebung interagieren und wie erfolgreich die vorgesehenen Inhalte übermittel werden, wurde eine Evaluation der Laborumgebung mit Probanden durchgeführt.

Die Laborumgebung wird in Form einer Virtuellen Maschine ausgeliefert um die Durchführung möglichst Betriebssystem-Agnostisch zu ermöglichen. In der Laborumgebung werden einige Services in Form von Docker-Containern zur Verfügung Gestellt. Zum einen wird die verwundbare Webanwendung *OWASP Juice Shop* betrieben. Dies ist eine Open Source Webanwendung die mit Absicht Sicherheitslücken enthält und zu Ausbildung von Sicherheitsexperten oder Sensibilisierung für Schwachstellen genutzt werden kann. Der zweite Service stellt mit der WAF *ModSecurity* den Kern der Labor-

umgebung dar. Es handelt sich um eine Open Source Anwendung, aus der das gesamte Regelwerk entfernt ist.

Die Lerneinheiten sind in drei Abschnitte unterteilt. Der erste Abschnitt führt Lernende in die Laborumgebung und die Arbeit mit dem *ModSecurity* Regelwerk ein. Die Lernenden setzen angeleitet die Laborumgebung auf ihren Rechnern auf und interagieren mit den Services um ein Verständnis für die Umgebung bekommen. In dieser Lerneinheit soll ein erstes Verständnis für das Schreiben und die Syntax der WAF-Regeln aufgebaut werden. Nachdem sich die Lernenden mit der Laborumgebung vertraut gemacht haben, kann in der zweiten Lerneinheit begonnen werden angriffe zu Verstehen und abzuwehren. Dies wird exemplarisch an den beiden Schwachstellen-Kategorien *SQL-Injection* und *Cross Site Scripting* durchgeführt. Es wird erst exemplarisch auf einzelne Schwachstellen im *Juice Shop* verwiesen, anhand derer in die Funktion der Angriffe eingeführt wird und Regeln zum expliziten mittigieren dieses einen Angriffs verfasst. Danach sollen die Schwachstellen allgemein betrachtet werden und anhand von einer Auswahl an Schwachstellen ein Schutz mithilfe einer Einzelnen Regel erstellt werden. Nach dem Verständnis für angriffe wird in der dritten Lerneinheit der Blick auf erweiterte Techniken und die Arbeit mit einer WAF gewendet. Die Lernenden sollen sich in dieser Lerneinheit mit dem Härten von Filterregel beziehungsweise dem Umgehen von Filterregeln beschäftigen. Des Weiteren soll mit den Log-Dateien gearbeitet werden um Fehler in der Konfiguration einer WAF aufzuspüren und diese zu Reparieren. Im Ramen der Thesis war es aus zeitlichen Gründen nicht möglich diese Dritte Lerneinheit zu realisieren.

Die Laborumgebung wurde nach erstellen mit der Hilfe von Probanden Evaluiert um des Einsatz mit Lernenden Einschätzen zu können. Ergebnisse sind zum einen, dass für die Durchführung ein Vorwissen in den Bereichen Reguläre Ausdrücke, der Funktion des HTTP-Protokolls und der Datenbanksprache SQL nötig sind. Die Durchführung der Lerneinheiten kann von einem Lernenden in Ungefähr acht Arbeitsstunden erfolgen. Von der Dauer her ist also ein Einsatz im Ramen einer Vorlesung möglich.

## 8.2 Ausblick

Diese Arbeit bietet einige Möglichkeiten weitere Überlegungen anzustellen und Erweiterungen vorzunehmen.
Zum Einen ist es für die tatsächliche Nutzung in der Lehre notwendig die Dritte und fehlende Lerneinheit zu Implementieren um die vorgesehene Lernerfahrung bieten zu können. Es ließe sich beispielsweise den Lernenden ein Regelsatz für eine bestimmte Schwachstelle des *Juice Shops* zur Verfügung stellen. Diese Regelsatz müsste so gestaltet sein, dass die Funktion die eigentlich an dieser Stelle steht nicht verfügbar ist

und nur durch das Umschreiben der Regeln ermöglicht werden kann.
Neben den fehlenden Inhalten existieren einige Punkte in denen noch weitere Arbeit betrieben werden kann um den Lernerfolg oder die Arbeit mit der Laborumgebung zu optimieren. Als Teil der Laborumgebung ist es möglich einen Service zu Implementieren der den Lernenden mittels HTTP-Anfragen und Analyse der Antworten rückmeldung über die Qualität der erstellten Regeln geben kann. Die Umsetzung eines solchen Services war als Teil der Laborumgebung geplant, in der vorgegebenen Zeit lies sich dies jedoch nicht zuverlässig Implementieren. Die rückmeldung an die Lernenden war nicht von einer nutzbaren Qualität.
Weiter Überlegungen können außerdem zur Bewertung der Ergebnisse angestellt werden. Kann die Laborumgebung prozedural so generiert werden, dass es möglich ist zu erkennen ob die Lernenden ihre Lösungen untereinander getauscht haben. Dies wäre beispielsweise durch individuelle Aufgabenzuteilung möglich, sodass jeder Lernenden individuelle aufgaben hat.

# 9 Bibliografie


[1] *Stand Der Technik*, https://www.teletrust.de/publikationen/broschueren/stand-der-technik/. (besucht am 07. 01. 2024).

[2] h. online, *IETF erhebt HTTP/3 zum RFC*, https://www.heise.de/news/IETF-erhebt-HTTP-3-zum-RFC-7135411.html, Juni 2022. (besucht am 01. 03. 2024).

[3] *An overview of HTTP - HTTP | MDN*, https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview Dez. 2023. (besucht am 01. 03. 2024).

[4] *HTTP Messages - HTTP | MDN*, https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages, Feb. 2024. (besucht am 01. 03. 2024).

[5] *HTTP response status codes - HTTP | MDN*, https://developer.mozilla.org/en-US/docs/Web/HTTP/St Nov. 2023. (besucht am 01. 03. 2024).

[6] *HTTPS / HTTP Secure (HTTP + SSL/TLS)*, https://www.elektronik-kompendium.de/sites/net/181128 (besucht am 06. 03. 2024).

[7] *NAT - Network Address Translation*, https://www.elektronik-kompendium.de/sites/net/0812111.ht (besucht am 01. 03. 2024).

[8] *Was ist ein Reverse Proxy, was ist ein Proxy Server?* https://www.cloudflare.com/de-de/learning/cdn/glossary/reverse-proxy/. (besucht am 01. 03. 2024).

[9] N. Gupta und A. Saikia, *WEB APPLICATION FIREWALL*, Apr. 2007. (besucht am 01. 03. 2024).

[10] P. Schmitz und G. Güttich, *Grundlagen der Web Application Firewalls*, https://www.security-insider.de/grundlagen-der-web-application-firewalls-a-694666/, März 2018. (besucht am 09. 01. 2024).

[11] H. Yuan, L. Zheng, L. Dong, X. Peng, Y. Zhuang und G. Deng, „Research and Implementation of WEB Application Firewall Based on Feature Matching," in *Application of Intelligent Systems in Multi-modal Information Analytics*, V. Sugumaran, Z. Xu, S. P. und H. Zhou, Hrsg., Ser. Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2019, S. 1223–1231, ISBN: 978-3-030-15740-1. DOI: 10.1007/978-3-030-15740-1_154.

# A Aufgabenstellung Teil I

Mit dieser Laborumgebung soll die Funktion einer Web Application Firewall (WAF) in drei Lerneinheiten vermittelt werden. In dieser Lerneinheit sollen Sie sich mit der Lernumgebung vertraut machen und eine erste Konfiguration der WAF vornehmen.

## A.1 Vorbereitungen

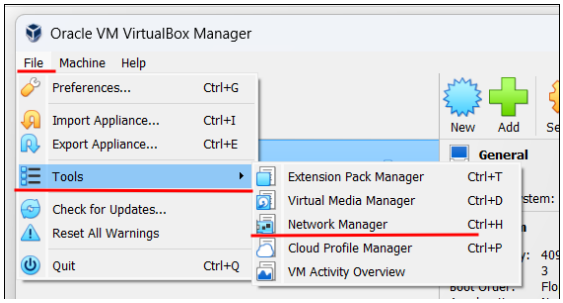
Um die Laborumgebung zu nutzen werden die Folgenden Anwendungen benötigt:

- VirtualBox
- Visual Studio Code
  Die nicht quelloffene Version von Microsoft ist notwendig, da ein benötigtes Plugin in *Open-Source* Distributionen wie *vscodium* nicht verfügbar sind.
- Einen Web-Browser

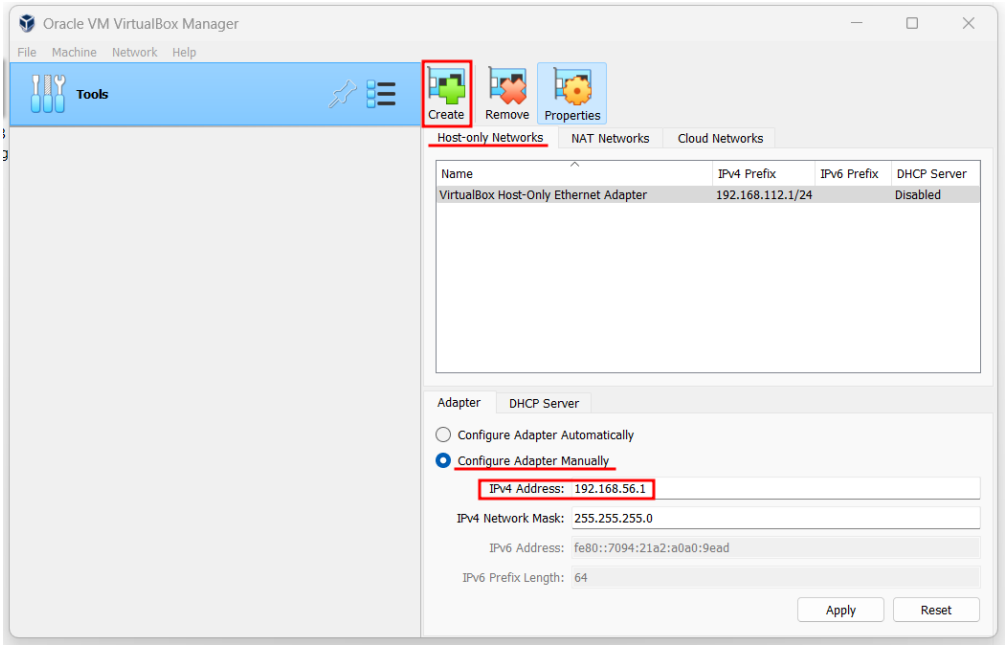
### A.1.1 Virtuelle Maschine starten

Importieren Sie nun die Virtuelle Maschine (VM), die die Laborumgebung enthält in die VirtualBox Umgebung. Die Datei (*WAF-Laborumgebung.ova*) finden Sie im Moodle. Bevor die VM gestartet werden kann, muss ein Host-Only Netzwerkadapter erstellt werden, um von ihrem Host-System auf die Services in der VM zugreifen zu können. Die VM soll auf der IP-Adresse **192.168.56.2** aufrufbar sein.
Die folgende detaillierte Anleitung ist exemplarisch für Windows Systeme. Wenn Sie ein anderes Betriebssystem oder eine andere Virtualisierungsumgebung nutzen, können Sie die Anleitung als Orientierung nutzen und müssen sich den Rest eigenständig erarbeiten.

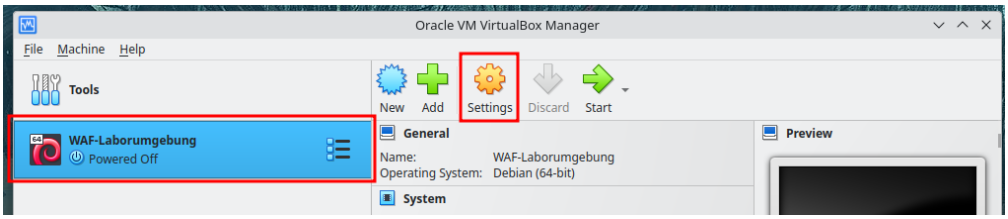
1. Den Netzwerk-Manager von VirtualBox aufrufen (*File -> Tools -> Network Manager*).



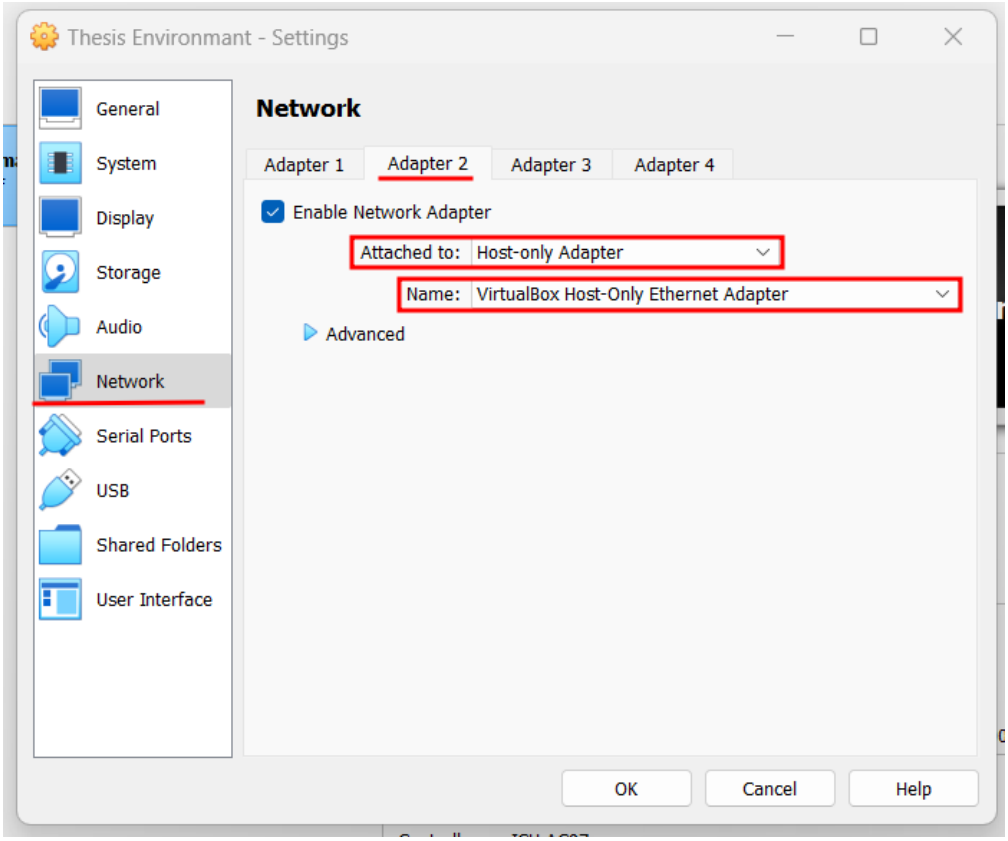
2. Den Reiter *Host-only Networks* auswählen.
3. Mit *Create* einen neuen Adapter erstellen.
4. Die IPv4-Adresse muss auf 192.168.56.**1** konfiguriert sein damit die Virtuelle Maschine aufgefunden werden kann.




5. Die Konfiguration der Laborumgebungs-VM aufrufen.




6. Den zweiten Netzwerk-Adapter unter (*Network -> Adapter 2*) aktivieren.
7. Den in Schritt 3 erstellten Adapter der VM zuordnen




Jetzt kann die Virtuelle Maschine gestartet werden.

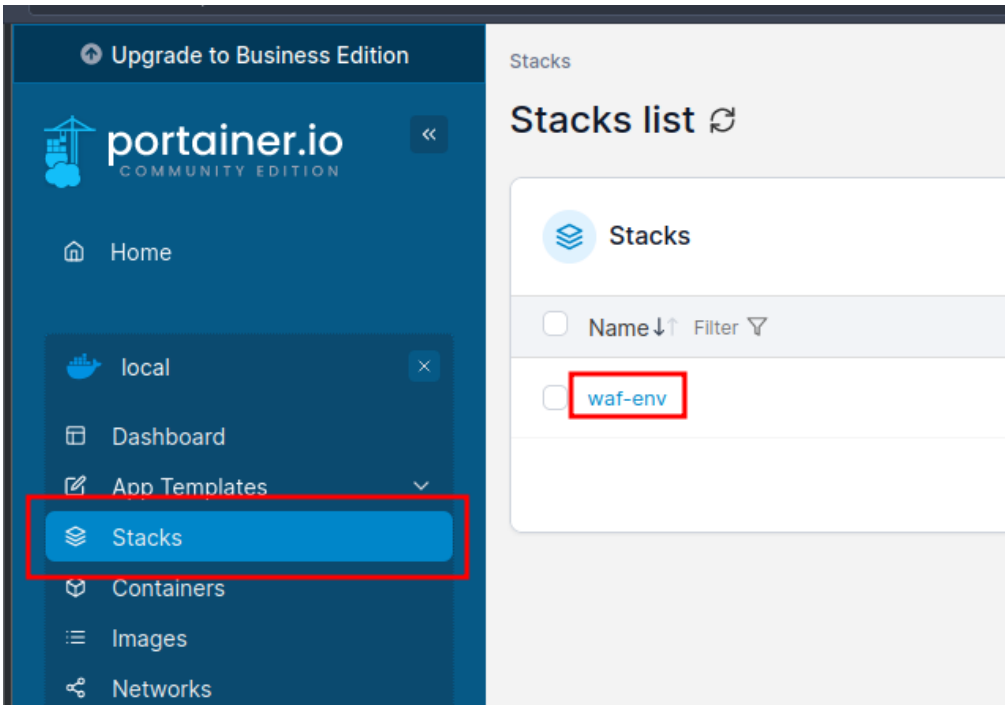
### A.1.2 Websites aufrufen

Die Lernumgebung bietet mehrerer Services mit denen interagiert werden kann, die alle mit Hilfe der Containervirtualisierungsumgebung Docker verwaltet werden. Unter https://192.168.56.2:9443 steht eine Docker-Weboberfläche zur Verfügung, um die Bedienung zu erleichtern. Die Login Daten sind:

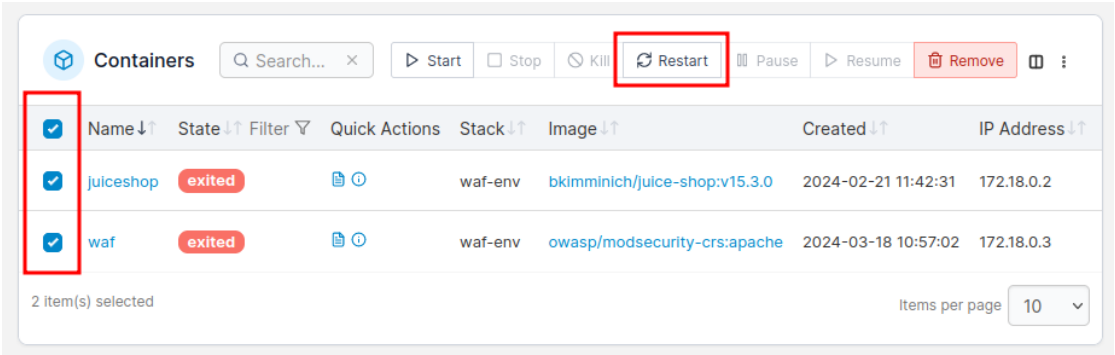
- **Username:** admin
- **Passwort:** waf-env12345

Nach einem Neustart der VM muss der WAF Docker-Container neu gestartet werden. Auch wenn die WAF-Konfiguration geändert wurde, muss die WAF neugestartet werden damit die Änderungen übernomen werden.
Nach dem Login muss als erstes das lokale Environment und der Docker-Stack ausgewählt werden. Dann kann die Laborumgebung (neu) gestartet werden:

1. *Stacks -> waf-enf* auswählen




2. Hier können die Container ausgewählt und neugestartet werden




Nun sind alle Services der Laborumgebung aktiv und sind unter den folgenden URLs abrufbar:

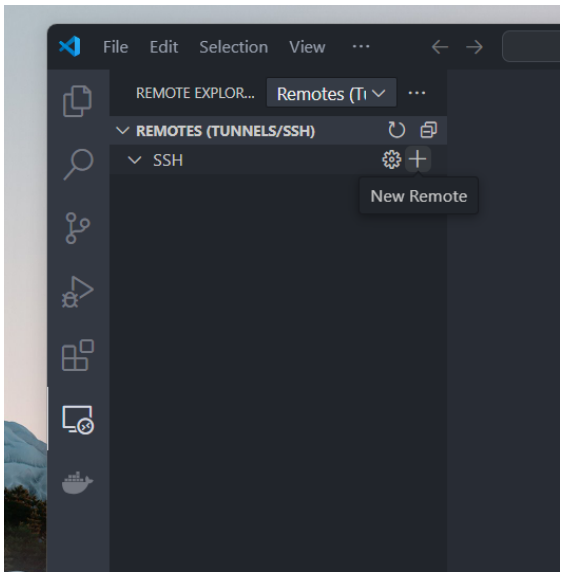
1. Verwundbare Anwendung mit WAF https://192.168.56.2:443
2. Verwundbare Anwendung ohne WAF http://192.168.56.2:3000
3. Portainer https://192.168.56.2:9443

### A.1.3 Die WAF Konfigurationsdatei bearbeiten

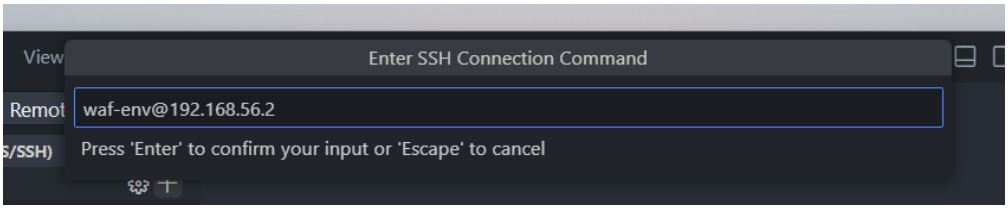
Der letzte Schritt zur Vorbereitung der Umgebung ist der Zugriff auf die WAF-Konfiguration. Dies erfolgt über SSH auf die Virtuelle Maschine. In dieser Anleitung wird die Verbindung mit der *Visual Studio Code* Extension Remote - SSH erläutert. Der Zugriff ist jedoch auch mit jedem anderen SSH-Client möglich.

- **URL:** 192.168.56.2:22
- **Username:** waf-env
- **Passwort:** waf-env

Ist die Extension in vscode installiert, kann unter *Remote Explorer -> SSH* mit dem +-Icon eine neue Verbindung angelegt werden.

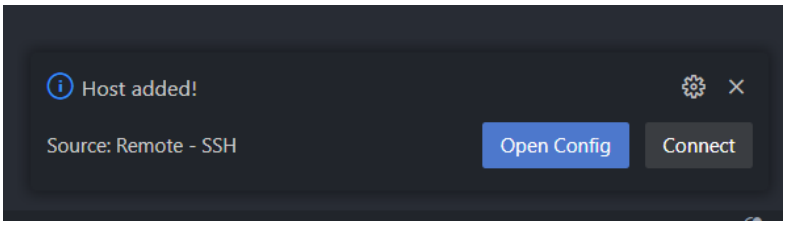



In dem sich öffnenden PopUp muss die SSH Verbindung eingetragen werden: **waf-env@192.168.56.2**.

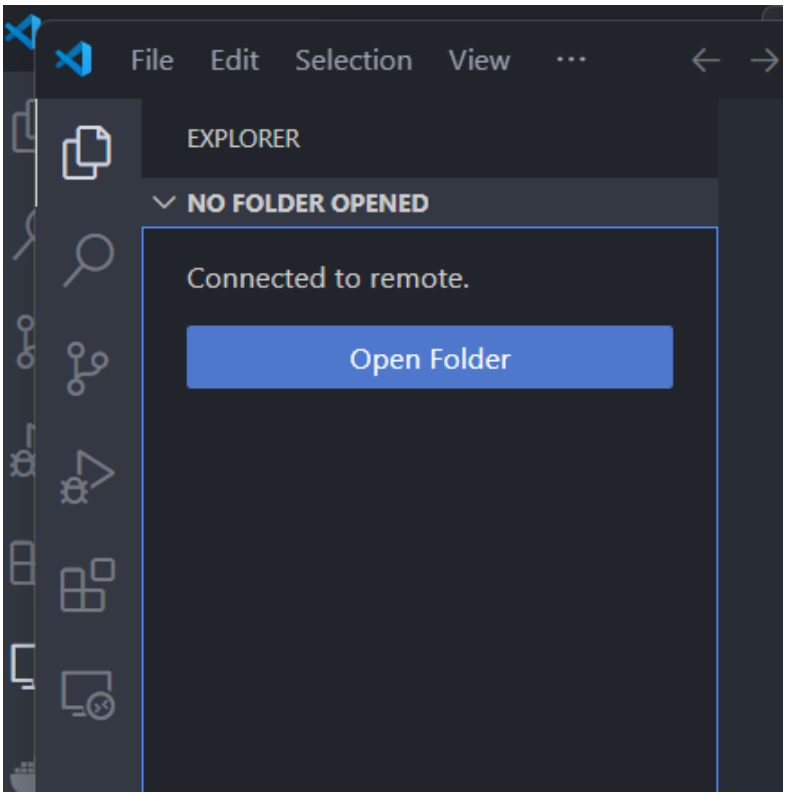



Es folgen PopUps in denen nach dem Gast Betriebssystem (Linux) und der Konfigurationsdatei für ssh gefragt werden. Nutzen Sie die vorgeschlagenen Optionen.

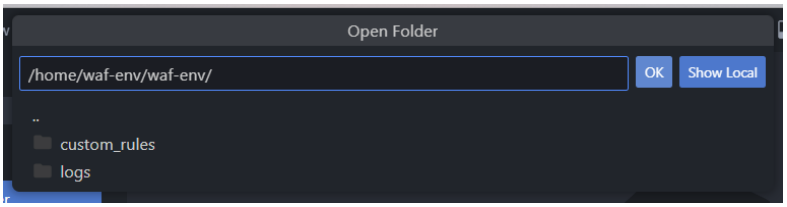
Während dem Prozess wird wiederholt nach dem Passwort (*waf-env*) gefragt.
Ist die Konfiguration abgeschlossen kann die Verbindung zu der VM mittels des Popups in der rechten unteren Ecke aufgebaut werden.




Nachdem die Verbindung aufgebaut wurde muss zu dem Ordner navigiert werden, der die Konfigurationsdateien und Logs der WAF enthalten.
Im Reiter *Explorer* kann der Ordner geöffnet werden.




In dem PopUp den Pfad */home/waf-env/waf-env* angeben und bestätigen.
Visual Studio Code merkt sich die Konfiguration die bei wiederholtem Aufruf im *Remote Explorer* aufgerufen werden kann.




## A.2 Erste Konfiguration

Die Ressourcen die Sie in der folgenden Liste finden, können Ihnen bei der Bearbeitung der Aufgabenstellungen behilflich sein.

- Das ModSecurity Refernece Manual besonders die Anleitung zum Schreiben von Regeln
- Auflistung der OWASP Juice Shop Challenges
- Beispiellösungen für den Juice Shop
- Der Progress tracker des Juice Shops der unter der URL */#/score-board* zu finden ist

### A.2.1 Sperren eines Pfades

Um sich mit dem Aufbau einer ModSecurity Regel vertraut zu machen, soll der Zugang zu einem Pfad in der Website gesperrt werden.
In der Challenge „access a confidential document“ ist der Zugriff auf ein Dokument möglich.
**Schreiben Sie eine neue Regel, die den Zugriff auf dieses Dokument verhindert und mit dem HTTP-Fehlercode 401 antwortet.**

### A.2.2 Parsen von HTTP Parameter

In der Challenge „give a devastating zero-star feedback to the store“ kann durch Bearbeiten eines HTTP-Requests ein Produkt mit 0 Sternen bewertet werden.
Vollziehen Sie das beschriebene Problem nach:

- Was ist der Fehler, der die Schwachstelle ermöglicht?
- Welcher Parameter enthält die schadhaften Daten?
- Was wäre das korrekte Verhalten?

**Wenn Sie die Schwachstelle verstanden haben, schreiben Sie eine weitere Regel, die verhindert, dass ein Angreifer keine Daten außer den gültigen Werten an den Endpoint senden kann.**

# B Aufgabenstellung Teil II

Sie haben sich im ersten Kapitel mit der Funktion der Laborumgebung vertraut gemacht und erste Regeln verfasst. Dieses Kapitel befasst sich nun mit den ersten Angriffen die in ähnlicher Form auch an öffentlich erreichbaren Servern beobachtet werden können. Im Folgenden werden Sie sich *SQL-Injections* und *Cross Site Scriptings (XSS)* genauer anschauen und WAF-Regeln für Angriffsszenarien im *Juice Shop* schreiben, die diese in den Dort aufgeführten Fällen mittigieren.

## B.1 SQL Injections

Die SQL-Injection ist eine sehr häufig auftretende Schwachstelle, die aus der unsauberen Weitergabe von Nutzerdaten an eine SQL Datenbank entstehen kann.

### B.1.1 SQL-Login-Bypass

In der Challenge „Log in with the administrator's user account“ kann die Passwort-Validierung bei einem Login umgangen werden.
Vollziehen Sie das beschriebene Problem nach:

- Was muss in der Login Maske angegeben werden, um als administrator eingeloggt zu werden?
- Wie funktioniert der Login Prozess im Backend?
- Überlegen Sie sich Funktionierende Abwandlungen des Befehls zum Login als Administrator.
- Was haben diese Gemeinsam?

**Schreiben Sie eine WAF-Regel, die es ermöglicht den Angriff zu verhindern, indem Anfragen die den Login Bypass enthalten, abgelehnt werden. Bedenken Sie, dass nicht nur ein Weg zu einer Ausnutzung führen kann.**

### B.1.2 SQL-Injections allgemein an drei Bespielen

Der Juice Shop bietet weitere Möglichkeiten für SQL injections. Zwei Beispiele sind die beiden Challenges „database schema extraction“ und „log in with Bender's user account“.

- Welche SQL-Keywords werden in den Angriffen genutzt?

- Welche Mutationen dieser Keywords können verwendet werden und der Angriff funktioniert immer noch?
- Welche Probleme können Auftreten wenn die SQL-Keywords zu streng blockiert werden?

**Schreiben sie eine Regel die beide Angriffe aus den Challenges anhand der SQL Keywords verhindern kann.**

## B.2 Cross Site Scripting (XSS)

Cross Site Scripting (XSS) ist eine weiter Klasse an Angriffen die in der aktuellen Cybersicherheit eine große Rolle Spielen. Eine XSS-Schwachstelle zeichnet sich dadurch aus, dass Nutzereingaben die HTML, CSS oder JavaScript enthalten ausgeführt oder als Teil der Webseite dargestellt werden.

### B.2.1 Reflected XSS

Die Challenge „Perform a persisted XSS attack bypassing a server-side security mechanism“ enthält eine persistente XSS Schwachstelle. Das heißt ein in HTML eingebetteter JavaScript Code-Abschnitt kann auf dem Server gespeichert werden und wird bei Aufruf auf einem Beliebigen client ausgeführt.

- Was macht eine XSS-Schwachstelle aus?
- Können andere Elemente als *iframes* und dem *javascript:alert* genutzt werden?

**Schreiben Sie eine Regel die das Ausnutzen der Schwachstelle verhindert. Erweitern sie die Regel um eine Rewrite direktive, die den schadhaften Abschnitt aus der Anfrage entfernt sie aber an den Server weitergibt.**

### B.2.2 XSS Protection HTTP Headers

Neben dem erkennen von XSS durch einen WAF existieren den HTTP-Header *X-XSS-Protection* der in Browsern eie XSS-Schutzfunktion aktiviert.
**Schreiben Sie eine Regel die in allen Antworten des JuiceShops diesen Header setzt.**