Hochschule
für Technik
Stuttgart

Erarbeitung und Evaluation 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:	18. 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

4.1 Einleitung

Das Feld der Web Application Firewall (WAF) gewinnt 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.

4.2 Umfang und Abgrenzung

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:

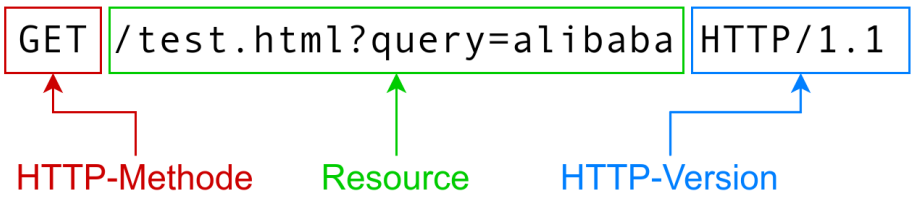

Abbildung 1: Beispiel für eine HTTP Request-Zeile

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.

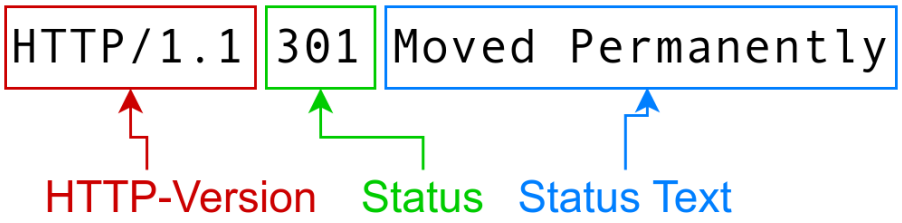

Abbildung 2: Beispiel für eine HTTP Status-Zeile

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.

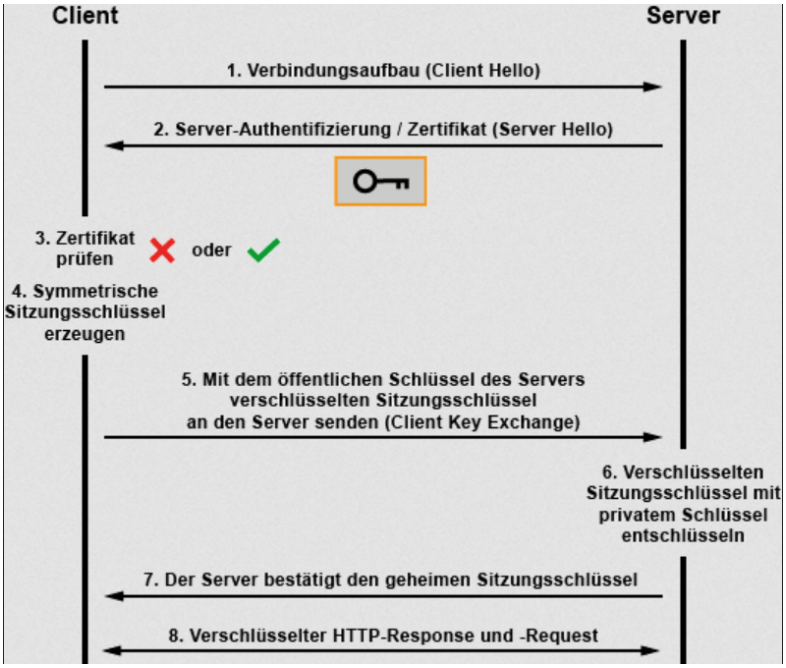

Abbildung 3: Prozess Ablauf eines HTTPS Verbindungsaufbaus

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-

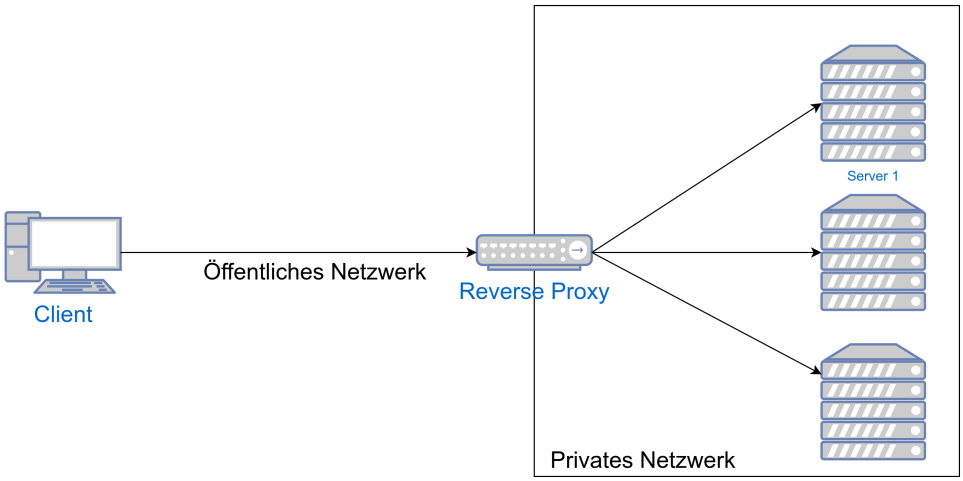

Abbildung 4: Schematische Darstellung eines Reverse Proxies

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.

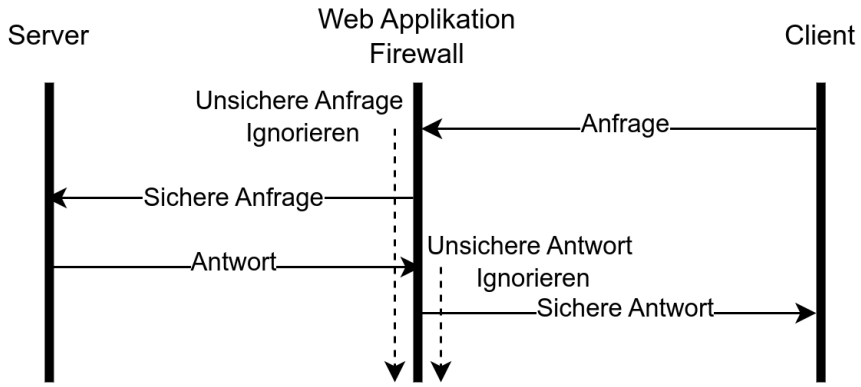

Abbildung 5: Prozess Ablauf der Verarbeitung durch eine Web Application Firewall

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 Sicherheitsanwendungen

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 separat-

[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.

en 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.

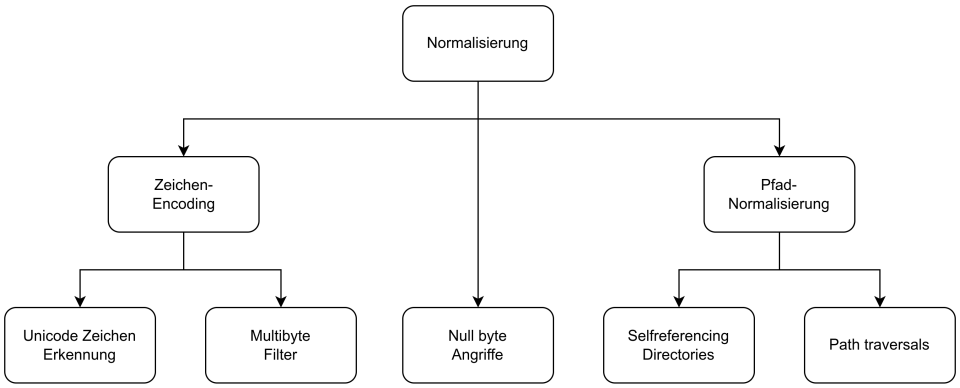

Abbildung 6: Normalisierungsoperationen

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 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 Zu übermittelnde Inhalte

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 dieser 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 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.

6.2 Technische Umsetzung

6.2.1 Evaluation verfügbarer Produkte

Auswahl der WAF-Anwendung

Verwundbare Anwendungen

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 Platform-unabhängig 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.

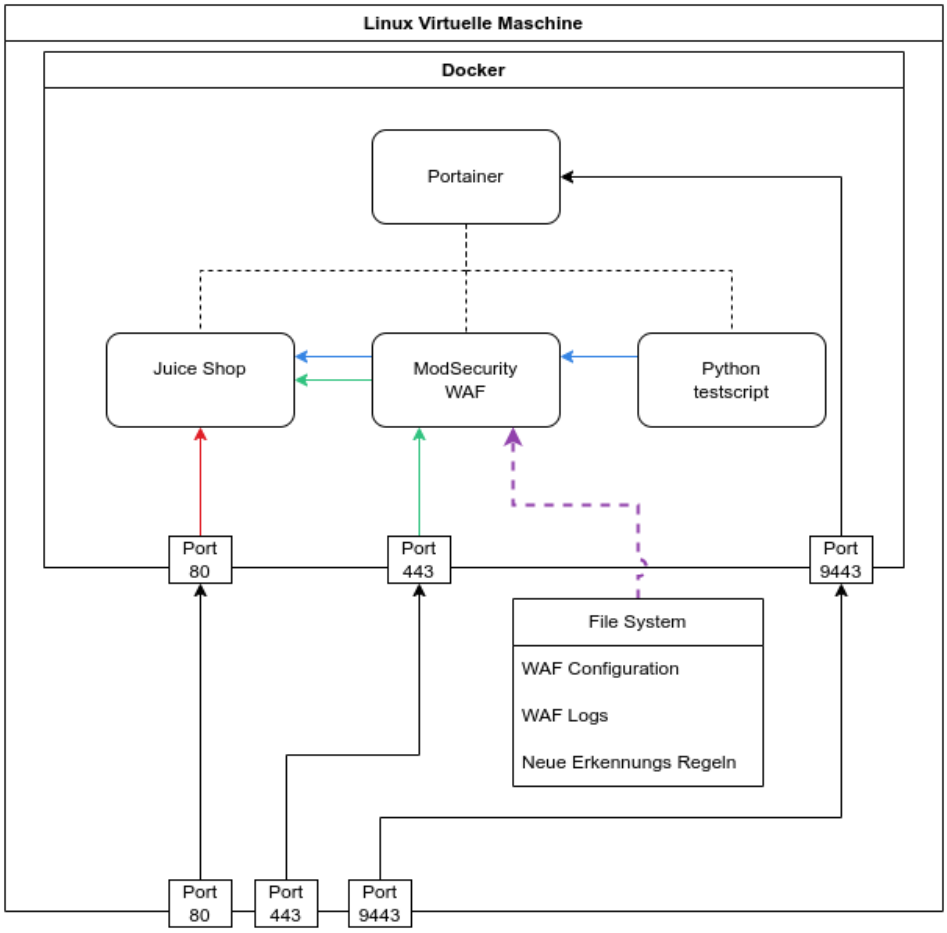

Abbildung 7: Aufbau der Laborumgebung

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. Die genauen Änderungen werden in den Sub-Kapiteln von Kapitel 6.3 im Detail beschrieben.

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. Die jeweilige Funktion wird in den Sub-Kapiteln von Kapitel 6.3 im Detail beschrieben.

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 6.3.1) 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

6.3.1 Teil 1: Erster Kontakt zu einer WAF

6.3.2 Teil 2: Grundlegende Angriffe

SQL Injections

Cross Site Scripting (XSS)

6.3.3 Teil3: Details und Gefahren in der Nutzung einer WAF

Log Analyse

7 Evaluation

7.1 Evaluation mit Probanden

7.2 Überlegungen zur Bewertung

8 Fazit

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 vorm=nehmen.

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

Es sind Konfigurationen notwendig um

A.1.2 Websites aufrufen

A.1.3 Die WAF Konfigurationsdatei bearbeiten

A.2 Erste Konfiguration

A.2.1

A.2.2

B Aufgabenstellung Teil II

C Aufgabenstellung Teil II