Hochschule
für Technik
Stuttgart

Erarbeitung und Evaluation einer Laborumgebung zum Thema Webapplication Firewall

Bachelor Thesis
im Studiengang Informatik

Betreuender Dozent:	Prof. Dr. Jan Seedorf
Betreuer des Unternehmens:	Oliver Paukstadt
Vorgelegt am:	28. Februar 2024
Vorgelegt von:	Lukas Reinke (Mat. NR. 1001213)

1 Abstract

Hier könnte ihr abstract stehen.

Inhaltsverzeichnis

2 Abkürzungsverzeichnis

WAF Web Application Firewall

VM Virtuelle Maschine

NAT Network Acces Translation

IT-SiG IT-Sicherheitsgesetzes

DSGVO Datenschutz-Grundverordnung

HTTP Hypertext Transfer Protokoll

DMZ Demilitarierte Zone

3 Abbildungsverzeichnis

4 Einleitung

4.1 Einleitung

Das Feld der WAF gewinnt in den aktuell immer mehr an Relevanz. Dem Markt wird in den nächsten fünf Jahren ein jährliches Wachstum um 19,9 % auf 14,6 Mrd. vorhergesagt [1]. 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[2, 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 sowohl Design und Implementierung einer WAF vermittelt. Jedoch sollen auch Themen vermittelt werden, die ein Consultant dessen Aufgabe die Betreuung einer WAF ist in seinem Berufsalltag benötigt. Dazu zählen zum Beispiel deployment und Anpassung einer WAF an die zu schützende Web-Anwendung.

4.2 Umfang und Abgrenzung

5 Theoretische Grundlagen

5.1 HTTP

Das Hypertext Transfer Protokoll (HTTP) ist ein Protokoll in der Internet-Kommunikation, dass zur übertagung diverser Daten unterschiedlicher Datentypen genutzt werden kann. Sein Haupt-Einsatzgebiet ist die Datenübertragung zwischen Webseiten und Clients. Seit der Einführung in 1991 wurde es in mehreren RFCs erweitert und ist inzwischen in version drei.

In seiner aktuellen Form kann es Gebrauch von TCP-Sitzungen machen um Verbindungen über längere Zeit aufrecht zu halten und fortgeschrittenere Kommunikation, wie push Nachrichten, zu erlauben. Außerdem kann TCP-Pipelining genutzt werden um die parallele Abarbeitung von Anfragen zu ermöglichen und nicht auf die *Acknowledge*-Nachrichten von TCP warten zu müssen. Um seine Grundfunktion zu erläutern wird in diesem Kapitel die statuslose Kommunikation, definiert in Version 1.0 die mit einem unmittelbaren Anfrage-Antwort Muster arbeitet, beschrieben. Hierin initiiert ein Client mit einer Anfrage Nachricht [1] eine Verbindung, die von einem Server verarbeitet und mit einer Antwort-Nachricht beantwortet wird. Darauf wird die Verbindung geschlossen. Dem Sever ist es nun nicht mehr möglich dem Client weitere Daten zu senden, ohne das der Client eine weitere Anfrage-Nachricht schickt.

HTTP-Nachrichten Die Grundlegende Einheit einer Hypertext Transfer Protokoll (HTTP)-Kommunikation wird als *Nachricht* bezeichnet. Da HTTP ein Klartext-Protokoll ist, werden diese in menschenlesbarer Form als Text übertragen. Eine Nachricht besteht aus einer *Start-Zeile*, die die Nachricht entweder als Anfrage oder Antwort identifiziert. In diesen beiden Fällen hat die Zeile jeweils einen untersiedlichen Aufbau:

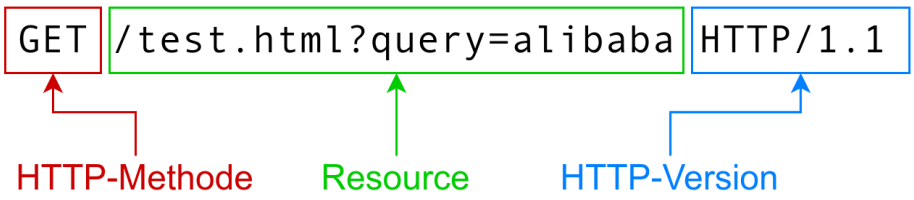

Abbildung 1: Beispiel für eine HTTP Request Zeile

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

Die HTTP-Methode beschreibt die Operation, die der Server ausführen soll und können auch *HTTP-Verb* und *HTTP-Noun* genannt werden. Syntaktisch basieren die

[1] Im folgenden werden HTTP-Anfrage und das Englische HTTP-request synonym verwendet

Methoden auf dem FTP Protokoll, das älter ist und mit Operationen wie GET und PUT arbeitet. Der Satz an Verben und Nomen ist im Verlauf der HTTP Versionen jedoch um weitere Methoden, wie zum Beispiel PATCH oder OPTIONS, erweitert.

Die angefragte Resource spezifiziert welche Resource angefragt wird. Üblicherweise eine URL. Optional kann die angefragte Resource mit einem Fragezeichen Gefolgt von einem sogenannten *query string* noch spezifiziert beziehungsweise gefiltert werden.

Der HTTP Verion die angibt in welcher Version die folgende Nachricht verfasst ist.

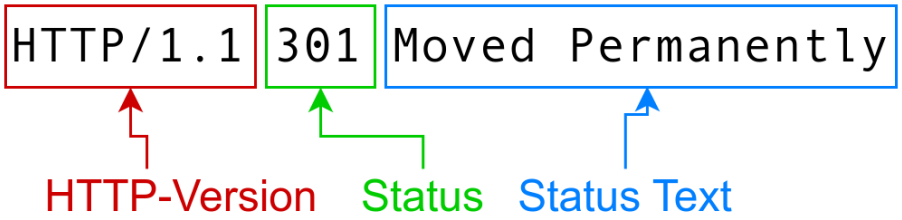

Abbildung 2: Beispiel für eine HTTP Status Zeile

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

Der Status code der durch eine dreistellige Zahl angibt wie der Status der Verarbeitung eines Requests war. Für verschiedene Ergebnisse ist ein Zahlenraum vorgeschrieben. So steht zum Beispiel 2xx für eine Erfolgreiche Verarbeitung während 4xx auf einen Fehler auf Client-Seite hindeutet.

Der Status Text , einem Beliebigen den Text der den Status genauer beschreiben kann. Im HTTP Standard ist zwar nicht vorgeschrieben wie der Text auszusehen hat, es gibt jedoch Konventionen.

Nach des Start Zeile folgen sowohl in einem HTTP-Request als auch in der Response zwei weitere Abschnitte. Diese werden genutzt um weitere Informationen an die Nachricht anzuhängen.
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 haben mehr unterkategorien wie *Request-* oder *General-Header*, hier ist es im Rahmen der Thesis jedoch nicht notwendig weiter in die Tiefe zu gehen. Eine Relevante Header-Gruppe sine 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. Er ist Optional. Im Body können sich beliebige Daten befinden: Binärdaten wie Bilder, JSON-Formatierte String oder einfacher Text.

Wie man oben beschrieben sehen kann bietet HTTP eine Reichweite an 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 genutzt zu werden. Auch die Möglichkeit Daten zu Codieren, speziell im *HTTP-Body*, kann zu diversen schadhaften Datenübertragungen führen. Eine WAF muss in der Lage sein alle dieser Parameter Überblicken zu können um effektiven Schutz zu bieten.

5.2 Verschlüsselung

5.3 NAT und Reverse Proxy

Die Funktion einer WAF baut auf der Fähigkeit aus Internet-Trafic weiterleiten und inspizieren zu können. Die Technische Grundlage hierfür ist ein Reserve Proxy. Im Folgenden wird beschrieben wie diese Technologie funktioniert, wie schon durch den Einsatz eines Proxies erste Sicherheitsaspekte zum Tragen kommen und wie sich ein Reverse Proxy von einfacheren Technologien wie Network Acces Translation (NAT) unterscheiden.

Network Acces Translation (NAT)
NAT ist der Oberbegriff für Funktionen mit denen es möglich ist, die IP Adressen von Netzwerk-Traffic umzuschreiben. Das heißt, Traffic kann über einen Relay-Punkt an unterschiedliche Clients oder Server weitergeleitet werden. Es wird auf den Schichten zwei und drei des TCP/IP Schichtenmodells gearbeitet. Ein Einsatzgebiet für NAT ist zum Beispiel die Adress-Umverteilung an einem Router. Dadurch besteht die Möglichkeit an einer IP-Adresse mehrere Server zu betreiben. Anfragen werden, je nachdem an welchem Port sie eintreffen, an eine Unterschiedliche IP-Adressen *hinter* dem NAT-Gerät weitergeleitet.
Diese Technologie ist in seiner Umsetzung sehr niedrig Komplex, ermöglicht jedoch so gut wie keine Sicherheitsvorteile. Der einzige kleine Vorteil kann die Verschleierung einer Internen Netzwerkstruktur sein. Es ist nicht möglich tiefer gehende Filterung vorzunehmen, da auf einer ISO/OSI Schicht gearbeitet wird auf der die Entschlüsselung von Inhalten nicht möglich ist. Protokolle die Verschlüsselung anbieten arbeiten auf Höheren Schichten.

Reverse Proxy
Ein Reverse Proxy ist in seiner Funktion deutlich Komplexer als NAT. Operiert wird auf Stufe vier des TCP/IP Schichtenmodells. Ein Reverse Proxy fungiert, wie ein NAT fähiges Gerät, als Weiterleitungsstelle für Netzwerk-Verkehr. Der Unterschied liegt jedoch auf der Protokoll-Ebene. Ein Reverse Proxy ist in der Lage höher-Komplexe Protokolle wie HTTP, HTTPS oder FTP zu verstehen und die Weiterleitung aufgrund dem Inhalt der jeweiligen Pakete vorzunehmen.
Hieraus ergeben sich einige Sicherheitsaspekte die durch einen Reverse Proxy entstehen. Da ein Reverse Proxy den Inhalt der HTTP Pakte inspiziert und weiterleitet kann eine Normalisierung der Anfragen erfolgen. Dies kann eine Verteidigung gegen Path-Traversal Attacken sein, bei denen sich Zugang zu Inhalten verschafft werden kann die nicht Veröffentlicht werden sollten.

5.4 OWASP Top-Ten

5.5 Web Application Firewall

Ein WAF ist eine Sicherheits-Applikation, die in der Lage ist den Datenverkehr zu und von einer Web-Anwendung zu Analysieren. Hierbei werden die Übertragenen Inhalte in der Tiefe auf schädliche Inhalte überprüft. Der in Abbildung 3 dargestellte Prozessablauf beschreibt wie eine WAF mit eingehenden Nachrichten umgeht.

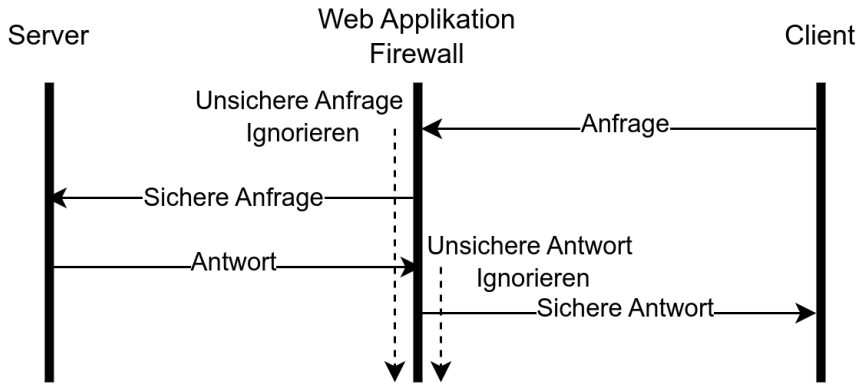

Abbildung 3: Prozess Ablauf der Verarbeitung durch eine WAF

Das grundlegende Muster ist, dass einen Nachricht an der WAF eintrifft und von dieser an den Server weitergegeben wird. Dieser Verarbeitete 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 Nutzer 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.

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 Klassischen Deployment-Methoden, bei denen die WAF als Virtuelle Maschine (VM)

oder Physischer Server im Unternehmensnetzwerk 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 an anderer Stelle betrieben. 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 klassische Firewall nimmt Filterung auf Internet- und Transportschicht (Layer 2 und 3) des TCP/IP Modells vor. Achtet also hauptsächlich auf 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 Umfang 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 klassischen Firewall. Die Erkennung schädlicher Inhalte erfolgen aufgrund von Regeln, die Muster beschreiben die auf schädliche Inhalte 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 regulärer Ausdrücke 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 befindet.
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 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 kann entweder unverschlüsselt oder mit einem separaten 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.

5.5.1 Verarbeitung einer Anfrage

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 der nach dem Eintreffen einer Nachricht an einer WAF. Vor diesem Schritt ist etwaige Verschlüsselung schon geöffnet, die Nachricht liegt in klassischer 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 für einen Angriff genutzt werden kann. Deswegen muss einen Normalisierung der Felder erfolgen, In Abbildung 4 sind die Charakteristiken die Normalisiert werden schematisch dargestellt.

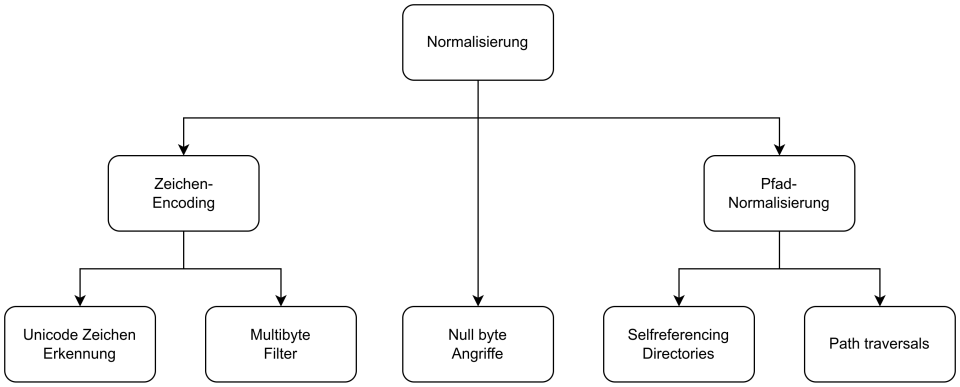

Abbildung 4: Normalisierungsoperationen

Normalisiert werden muss zum Einen die Zeichendarstellung. Die Nachricht muss in ein einheilichs 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 der 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 ein Zeichen das nicht dargestellt wird. 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.
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, verfahren werden soll.

Logging

Weiteres Vorgehen

5.5.2 Erweiterte Funktionen

Lernen von Regeln aus vorhergegangenem Datenverkehr

KI-Features

5.5.3 Deployment einer WAF

Postitionierung einer WAF

Betrieb einer WAF

5.5.4 Schwächen und Nachteile einer WAF

6 Design der Lernumgebung

6.1 Zu übermittelnde Inhalte

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-Applikation 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.

Modifizierbarketit der Anwendungen: Um in den Lerneinheiten grundlegende Techniken zu übermitteln, ist es notwendig Basis-Funktion der ModSecurity WAF entfernen zu können. Dies muss automatisierten und einheitlichen Weg erfolgen können.

Bekannte Basis-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 zuj 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, wurde die in Abbildung 5 schematisch dargestellte Umgebung erstellt.
Als Basis wird die Containervirtualisierungsumgebung Docker 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 lässt sich festlegen wie die einzelnen Anwendungen (Container) untereinander kommunizieren und virtuelle Netzwerke anlegen um die Kommunikation zu isolieren. Außerdem können Dateien oder Ordner aus dem Host-Dateisystem

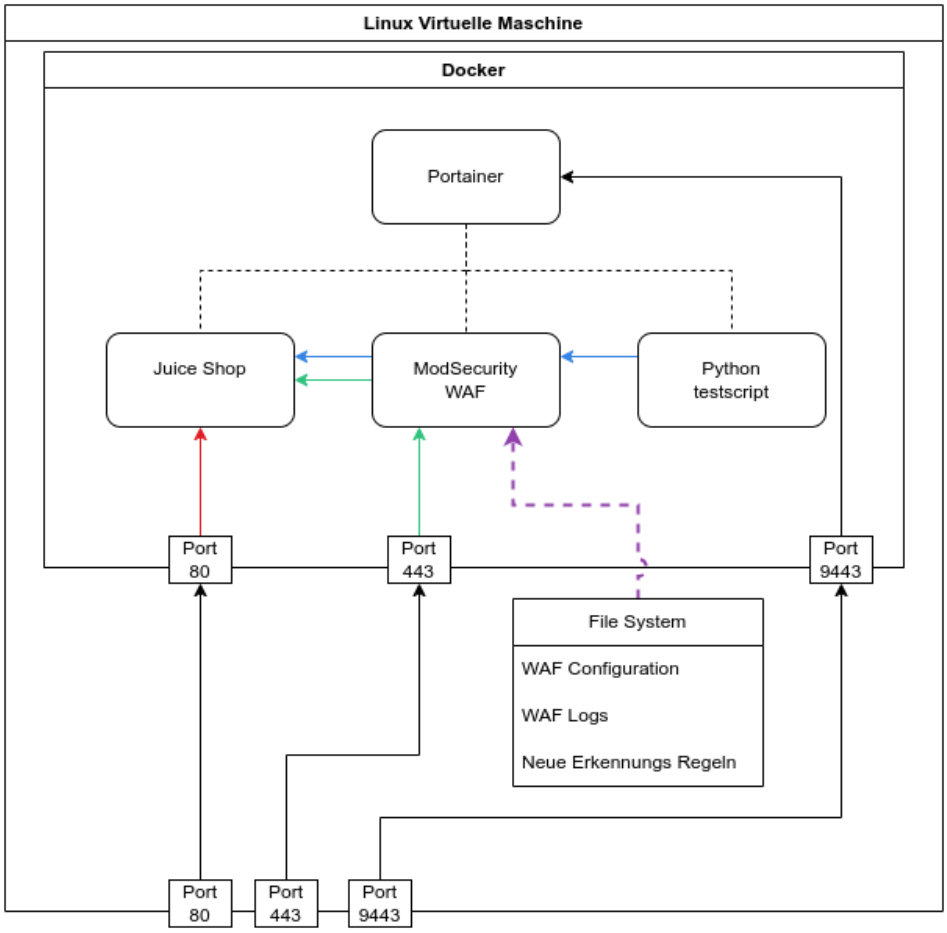

Abbildung 5: Aufbau der Laborumgebung

in den Container übergeben werden. Dies ermöglicht es Konfigurationsdateien zu Verfügung zu stellen und somit, die sonst Status-losen Container, anzupassen. 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 5 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. Da die WAF mit einer großen Anzahl vorkonfigurierten Regeln ausgeliefert wird, die für die meisten Aufgaben schon eine Lösung enthalten würden, müssen diese entfernt werden. An dessen Stelle wird ein Verzeichnis aus dem Host-Dateisystem durchgereicht in dem die Lernenden eigene Konfiguratio-

nen Platzieren können. Daneben werden, um das Debugging zu ermöglichen, die Log-Dateien aus der WAF im Host betriebssystem zur Verfügung gestellt.ds

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 in modifizierter Form Ausgeliefert. Es werden Daten hinterlegt und Nutzeracounts angelegt. Diese ermöglichen es Mittels eines automatischen Kontroll-Scriptes 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 Kofiguyration 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 Funtion von Docker auseinander setzen zu müssen. Zwar können die Lernenden dies auch über das Docker Komandozeilen-Interface tun, jedoch beurteile ich dies als eine vermeidbare Hürde, die den Einstieg erschweren könnte. Die grafische Oberfläche soll unter anderem genutzt werden um des WAF kontainer nach einer Konfigurationsänderung neu zu starten und mit dem Test Skript zu interagieren.

Aus den oben genannten Containern ergeben sich einige Netz die im Hintergrund existieren müssen. So ist es notwendig das von dem Python Test-Container zur WAF und von dieser eine Verbindung zum Jucice Shop aufgebaut werden muss. 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 Web-Oberfläche des Juice Shops (Port 80 [HTTP])
- Die, durch die WAF gesicherte, Web-Oberfläche des Juice Shops (Port 443 [HTTPS])
- Das Management Interface der Portainer-Anwendung (Port 8443][HTTPS])

Durch die oben beschriebene Docker Umgebung sind Anforderungen an die Laborumgebung wie dem *einheitlichen Deployment* und der *Modifizierbarketit 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 auszufühen, jedoch kann ein solcher Aufbau durchaus zu unvorhergesehenen Problemen führen. Ein weiteres Problem ist, dass durch das Durchreichten von Dateien zwischen Container und dem Host-Dateisystem zusätzlicher Konfigurationsaufwand für die Nutzer entsehen könnte.
Um diesen Problemen Vorzubeugen wird die Laborumgebung als Linux Virtuelle Maschine ausgeliefert. In dieser ist eine Docker Umgebung vorinstalliert und die Containerumgebung bereits präsent und wird automatisch 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 Web-Oberflächen 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 halte ich durch diese Maßnahmen für einfach genug um einen schnellen einstieg zu Ermöglichen. Die Konfigurationen, die vorgenommen werden müssen, werden in der Ersten Lerneinhiet (Kapitel 6.3.1) Beschrieben. Es steht den Lernenden frei weitere oder andere als die beschriebenen Technologien zu verweden um mit der Laborumgebung zu interagieren. Diese können im Rahmen dieser Thesis und den Aufgabenstellungen jehdoch 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] *Web Application Firewall Marktgröße & Anteilsanalyse - Branchenforschungsbericht - Wachstumstrends*, https://www.mordorintelligence.com/de/industry-reports/web-application-firewall-market. (besucht am 07. 01. 2024).

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