Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Master's Thesis

Joonas Blomberg

SECURING INTERNET OF THINGS WITH A WEB APPLICATION FIREWALL

Examiners: D.Sc. (Tech.) Ari Happonen
Prof. D.Sc. (Tech.) Jari Porras

Supervisors: D.Sc. (Tech.) Ari Happonen

ABSTRACT

Lappeenranta University of Technology
School of Business Management
Degree Program in Computer Science

Joonas Blomberg

Securing Internet of Things with Web Application Firewall

Master's Thesis

2017

53 pages, 11 figures, 6 tables, 1 appendix

Examiners : D.Sc. (Tech.) Ari Happonen
Prof. D.Sc. (Tech.) Jari Porras

Keywords: computer science, web application firewall, network security

The amount of internet-enabled machines is increasing rapidly due to the internet of things technology and web security is lagging behind. This thesis looks at web application firewalls as the security layer for IoT enabled devices. Web application firewalls' features and security models are explained and XMPP's relevance to IoT and WAFs is researched. A metasearch is also conducted on the WAF field. An XMPP filtering plugin was implemented on the open software WAF ModSecurity and it was evaluated with the help of an evaluation tool.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
LUT Kauppatieteet ja tuotantotalous
Tietotekniikan koulutusohjelma

Joonas Blomberg

Asioiden internetin suojaaminen web-sovelluspalomuurilla

Diplomityö

2017

53 sivua, 11 kuvaa, 6 taulukkoa, 1 liite

Työn tarkastajat: Tekniikan Tohtori Ari Happonen
Professori Jari Porras

Hakusanat: tietojenkäsittelytiede, sovelluspalomuurit, tietoturva
Keywords: computer science, web application firewall, network security

Internetiä käyttävät koneet lisääntyvät huomattavaa vauhtia asioiden internetin johdosta, mutta tietoturva laahaa perässä. Tämä työ tutkailee web-sovelluspalomuurien soveltuvuutta asioiden internetin tietoturvaa lisäävänä kerroksena. Web-sovelluspalomuurien toiminnallisuudet ja tietoturvamallit selvitetään ja XMPP:n roolia asioiden internetin sekä web-applikaatiopalomuurien kannalta tutkitaan. Web-sovelluspalomuurien tutkimuksesta tehdään metakysely trendien selvittämiseksi. Työssä toteutetaan web-sovelluspalomuurin lisäosa XMPP-viestien suodattamista varten. Se toteutetaan vapaan lähdekoodin työkalulla nimeltä ModSecurity ja arvioidaan siihen soveltuvan työkalun avustuksella.

ACKNOWLEDGEMENTS

This thesis was done in Lappeenranta University of Technology. I would like to thank the examiners Ari Happonen and Jari Porras. Special thanks to Ari for directing this work and helping me with coming up with ideas.

I would also like to thank Cygate Oy and my boss Janne Jääskeläinen for the opportunity to do this thesis.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BOSH	Bidirectional-streams Over Synchronous HTTP
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
DoS	Denial of Service
DDoS	Distributed Denial of Service
EDoS	Economic Denial of Sustainability
gzip	GNU zip
H2M	Human-to-Machine
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
M2M	Machine-to-Machine
NAT	Network Address Translation
OSI model	Open Systems Interconnection model
OWASP	The Open Web Application Security Project
PC	Personal Computer
regex	regular expression
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
VPN	Virtual Private Network
WAF	Web Application Firewall
WWW	World Wide Web
XEP	XMPP Extension Protocol
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XSS	Cross-Site Scripting

1 INTRODUCTION

The new trend in home electronics seems to be internet connectivity. While the idea of internet connected refrigerators and smart TVs is not a new one, the price of embedded screens and microprocessors have made these appliances affordable for consumers. While it is very convenient for consumers, there is also another side. Marketers gain more ground on personalized marketing when the appliance vendors undoubtingly sell their collected data. Furthermore, black hat hackers would gladly add your coffeemaker to their botnet.

Indeed, internet of things devices are already contributing to large DDoS (distributed denial of service) attacks. According to Akamai 7 out of 12 mega attacks, which means larger than 100Gpbs, on the last quarter of 2016 can be attributed to Mirai botnet. Mirai is IoT-based and large part of its bots are IP-enabled (Internet Protocol) cameras and digital video recorders. [1]

Security is generally not something you think about before it is compromised. While the higher-end vendors might have better security features in their IoT-devices what happens when other vendors start inevitably mimicking them? Up to this point, the trend has been that internet security lags behind new technologies and it is very likely to continue that way. This thesis tries to enhance home security by extending a web application firewall to block unwanted network traffic of a common IoT protocol.

1.1 Background

When HTML (Hyper-Text Markup Language) 1.0 debuted in 1991, the World Wide Web (WWW) was a series of static documents linked together by hyperlinks. When you type a web address on the browser or click a hyperlink, the browser requests that resource from the server and if the request is valid, the document is sent back and then processed by the browser. The request-response process is shown in Figure 1. Due to the static nature of the primitive websites, it was not feasible to make cyber-attacks via the OSI (Open Systems Interconnection) model's 7^{th} layer, also known as the application layer. Figure 2 depicts the layers of OSI and their associated Protocol Data Units. In the early days of the web, the

attacks were made by trying to access vulnerable services of the web servers or in the case of users, by infecting them with malware.

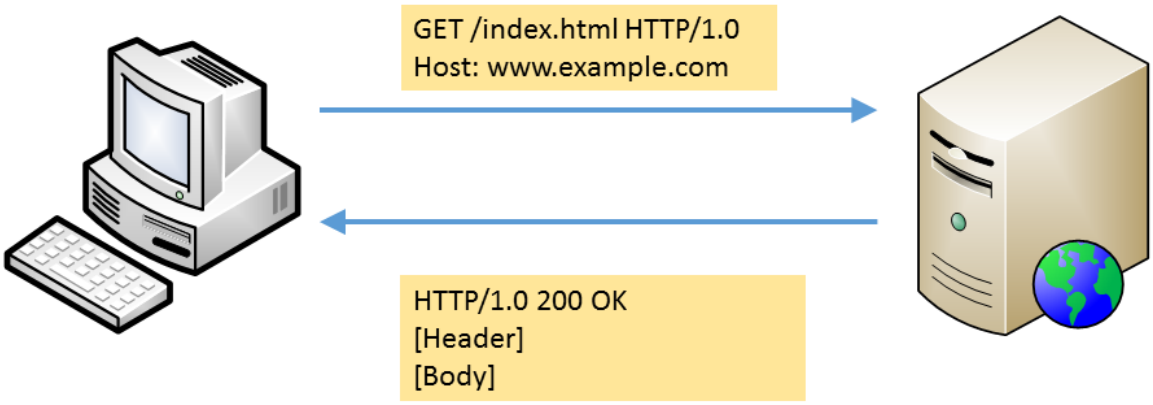

Figure 1. An HTTP (Hyper-Text Transfer Protocol) request-response where a browser requests an HTML file from a webserver.

Layer	Protocol data unit
7. Application	Data
6. Presentation	
5. Session	
4. Transport	TCP / UDP
3. Network	Packet
2. Data link	Frame
1. Physical	Bit

Figure 2. OSI model is a conceptual model of computer communication. Host layers (green) have been separated from Media layers (yellow) by color.

The first generation of firewalls were hardware appliances that isolated corporate networks from the internet by filtering packets. They prevent attacks up to OSI model layer 3. These stateless firewalls inspect the address and port of packets based on a set of filtering rules and either discards them or passes them through as demonstrated in figure 3. The statelessness implies that they have no memory of previous packets so a decision is made

on a straightforward packet-by-packet basis. The second generation of firewalls are connection aware and in addition, operate on OSI model layer 4. By inspecting incoming and outgoing packets over time, they can more intelligently make filtering decisions thus providing more usability and security than the first generation appliances.

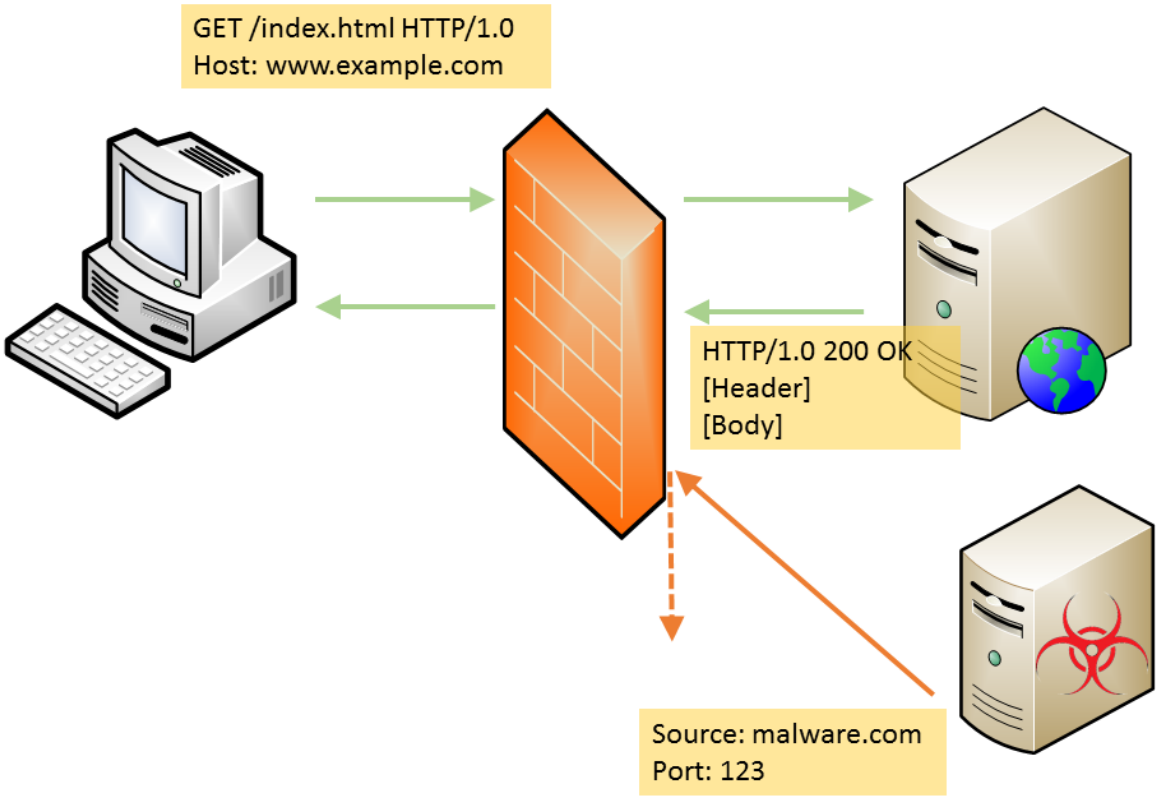

Figure 3. Hardware firewall blocks unknown packets and lets allowed packets through.

As firewalls were intended to secure internal networks from larger networks they were mainly used industrially in the 90's. Back then internet security was not a concern for consumers; the attention was on viruses, worms and Trojans which spread via floppy disks, email and infected or malicious software and their installers. Consequently, in the beginning of the 90's we saw the rise of many still recognizable antivirus software like Panda security, Norton Anti-Virus and F-secure. It is no surprise that at the turn of millennium many antivirus companies began to distribute personal firewalls. They are software-based firewalls installed on a Personal Computer (PC), so essentially a hardware appliance that is not trivial to install and configure is now accessible to consumer and

sometimes this feature is even included in an antivirus software. The Windows Firewall was introduced with Windows XP Service Pack 2 in 2004.

When JavaScript was introduced in 1995, it meant that web browsers were now sharing some of the load with clients by client side scripting. It allows for more dynamic and user friendly websites but also comes with a cost. Since JavaScript runs on the client's browser, it is a security risk. Consider the following situation: an attacker injects a script into a website because user input was not correctly sanitized and another user then visits that site and runs the script. The script instructs the browser to send the current session token to the attacker. The attacker is now logged in as the victim on the website. This is an example of a Cross-Site Scripting (XSS) attack, which occurs on the application layer.

Web application firewall (WAF) as a term started appearing in scientific articles around 2005. They are the first firewalls that prevent attacks up to OSI model layer 7. Numerous high profile data breaches in recent years have driven companies to adopt WAFs and in 2013, the market earned revenues of global WAF market was $431 million [2]. Currently there have been no need for WAFs for consumer use yet but in the future, as more and more home appliances will be connected to the internet, the need for firewalling on the application level increases.

1.2 Motivation and goals

The motivation for the thesis is for the author as a new employee to familiarize himself with various IoT technologies and web application firewalls. This knowledge should be useful in later assignments. Some of the restrictions of this project are the result of it being a one-man project with a limited time and limited resources. The author also wants the work to contribute to the open source community, which explains the usage of open technologies.

The goal of this thesis is to develop a web application firewall for an internet of things protocol and more specifically XMPP (Extensible Messaging and Presence Protocol). The protocol was chosen based on its message representation, which is textual instead of

binary. Another significant reason is that XMPP is related to HTML, a language that already has application firewalling implementations. The WAF will be implemented as a plugin for a popular open source web application firewall called ModSecurity, which was chosen for its maturity, popularity and XML (Extensible Markup Language) support.

The research questions for this master thesis are:

- Can ModSecurity be easily extended to handle firewalling of XMPP applications by singular programmers with limited resources?
- In relation to the first question, what are the features that ModSecurity could still be improved upon?

1.3 Structure of the thesis

After this introductory section, the second section will explain the concept of web application firewalls, what they are and how they work. The third section will explain the protocol known as XMPP and its relation to IoT and WAFs. The fourth chapter is a literature review, which expands on what research has been conducted in relation to web application firewalls. There is also a metasearch conducted, which helps in charting out the trends in research and also the amount of research done during the lifetime of WAFs. The fifth section is for describing the practical work done for this thesis. It shows the reader important parts of how the WAF plugin was implemented, what complications there were and it also discusses some issues. The results and evaluation are also in that section. The sixth section concludes this thesis.

2 WEB APPLICATION FIREWALLS

This chapter explains what a web application firewall is and what it does. Their components are explained and two security models are discussed: anomaly and signature based blocking.

2.1 What is a Web Application Firewall?

There does not yet exist a simple way of defining a web application firewall. The Open Web Application Security Project (OWASP) defines it as an "appliance, server plugin, or filter that applies a set of rules to an HTTP conversation" [3]. In general, a WAF is located on the web server it protects or on an intermediate reverse proxy between an external network and one or more web servers as demonstrated in figure 4. It monitors the seventh OSI layer known also as the Application layer and protects the web application from malicious or erroneous behavior. Therefore it can prevent attacks that intrusion detection systems and network firewalls cannot.

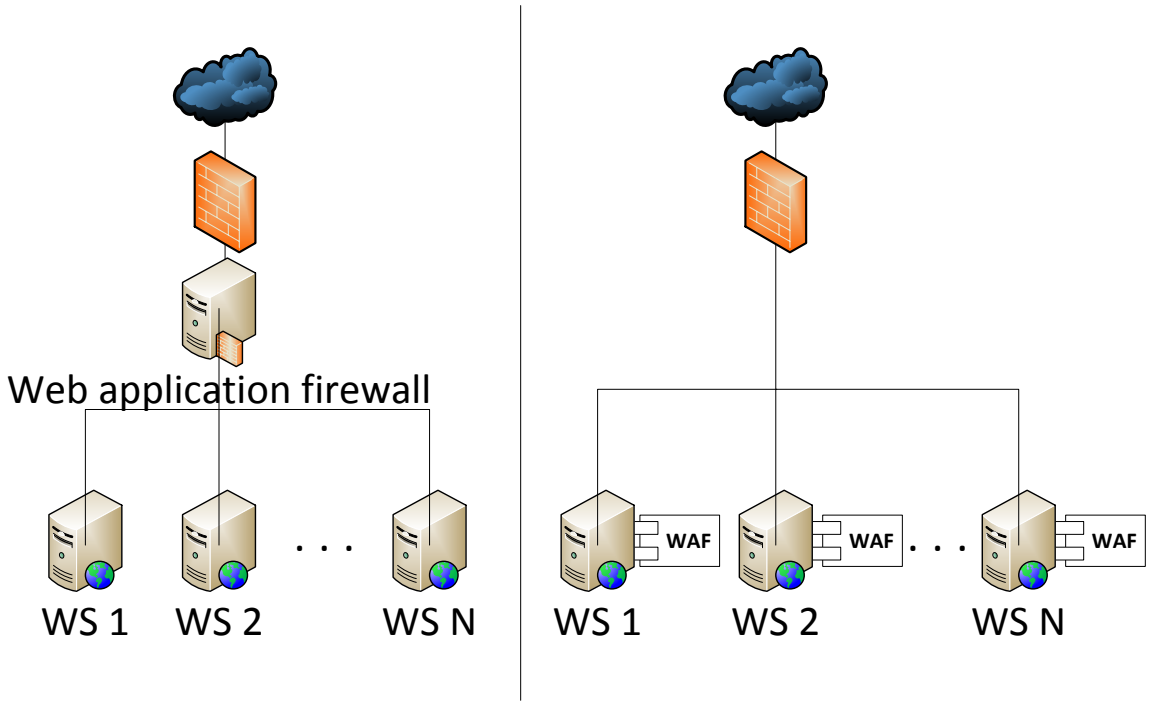

Figure 4. Reverse proxy on the left and web server (WS) embedded WAF on the right.

By looking at the most common features that WAFs have we can gain a good understanding of what WAFs generally look like. The following table 1 was collected by the author and it shows the features that are essential, and some extended features that more advanced WAFs have. There is also some web application security features that can be mistaken to belong to WAFs but do not.

Table 1. WAF features.

Feature	Core WAF feature	Extended WAF feature	Not a WAF feature
Understands application layer protocols: HTTP	✓		
├HTTPS	✓		
├XML		✓	
IP blocking by IP range	✓		
├ by geolocation		✓	
├ by reputation		✓	
Request blocking	✓		
Response blocking		✓	
Negative security model	✓		
Positive security model		✓	
Virtual Patching	✓		
Character decoding	✓		
Brute force attack mitigation	✓		
Cookie protection		✓	
Logging	✓		
Reporting		✓	
Vulnerability scanning			✓
Software debugging			✓

A web application firewall is a powerful tool because it changes the behavior of a web application without requiring any changes to the web application itself. Web applications are rarely totally safe from attackers and fixing known security holes can take a significant amount of time; a factor that is not always dependent of the service provider. In the light of this and zero-day vulnerabilities, which are vulnerabilities not yet known to security vendors, it is crucial to implement a WAF to systems that have severe consequences when they are exposed to cyber-attacks. In cases where setting up a WAF is not necessarily

practical or efficient it is possible to evaluate the effort to set up a WAF versus implementing the same security measures in the protected web application. In general, the further down in development the application is the costlier it is to make changes. This also applies to security features. The following figure 5 demonstrates the potential benefits of a WAF depending on the access to the application. We can conclude that when there is no access to an application, a WAF is highly beneficial for security.

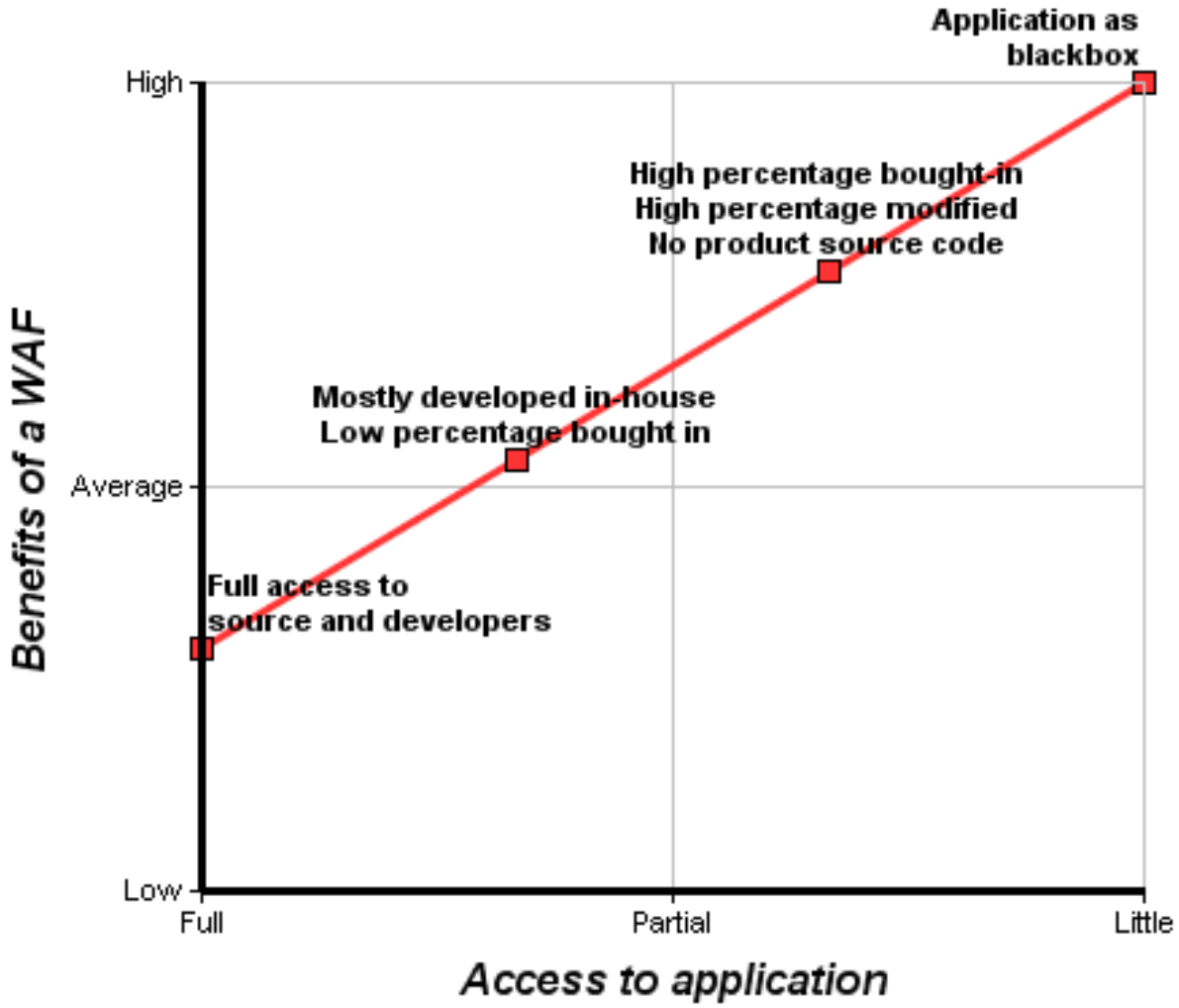

Figure 5. Benefits of a WAF for different levels of access to an application [4].

As was mentioned earlier, there are two deployment options for WAFs. The reverse proxy implementation has the benefit of having complete control over the web traffic since all traffic is passed through it. It can also provide Secure Socket Layer (SSL) offloading which saves resources from the web servers, cloaking server data and error messages, better DDoS protection, and masking of sensitive data leakage. The disadvantages of reverse proxy WAF is that it introduces an additional point of failure into the network and it is also less scalable. Although the embedded WAF puts additional load on the server and has less features it might still be viable in some cases because it doesn't need any network reconfiguration. In those cases the absent features must be implemented in some other way to insure the security of web applications. [5]

2.2 Preventive security measures of WAFs

WAFs are often simplified as signature or anomaly based request blockers but in reality, they provide a variety of security measures to mitigate cyber-attacks. Preventive features of WAF are input validation, cloaking, data loss protection, denial of service (DoS) protection and virtual patching.

2.2.1 Input Validation

Unexpected user input can cause errors, unexpected behavior and crashes in web applications. Web attacks that exploit inputs include the following: buffer overflow, automated input, code injection, hidden field manipulation, and cross-site scripting. Client side input validation can easily be circumvented by an attacker.

Web application firewalls can decrypt and normalize data before running checks on the input that might otherwise bypass validation. They can also validate input key names, and value type and length. Some WAFs are able to recognize if read-only or hidden fields have been manipulated.

2.2.2 Cloaking

Hackers are often trying to probe information about the underlying web server, database server and operating system to assess which types of attacks they might be vulnerable to. By suppressing error messages, server banners, HTTP headers, default return codes and debug information the risk of security breach is reduced. Reverse proxy WAFs provide cloaking for web servers.

2.2.3 Data loss protection

Web application firewalls can recognize and filter out sensitive data on web pages like credit card information and social security numbers. These two options are usually implemented out-of-the-box but custom patterns can also be created by the WAF administrator. The Payment Card Industry Data Security Standard version 1.1 compliance requires companies to use either WAF for data loss protection or manual code reviews [6].

2.2.4 DoS protection

In Denial of Service attacks, the attacker is trying to slow down or make the service unavailable for other users by overloading the server with requests. Service in this case usually means the application that is running on the server. In DDoS the requests come from several unique IP-addresses which are usually malware-infected Personal Computers. WAFs can detect when a website is under DoS attack based on statistics such as transactions per second and latency. The attack is mitigated by blocking the IPs that have higher activity or by serving them with a slower rate. Because the attacks are performed by bots the WAF can send them a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) challenge to distinguish human users from bots. It is also possible to block IPs based on their geolocation and reputation score. Big firewall appliance vendors usually have their own block lists, which are updated periodically.

2.2.5 Virtual patching

In [7] virtual patching is defined as “a security policy enforcement layer that prevents the exploitation of a known vulnerability”. Sometimes other WAF features such as, for example, input validation can prevent software vulnerabilities but virtual patching is usually its own feature. The purpose of virtual patching is to prevent a known malicious transaction from reaching the application until the vulnerability is fixed. In essence, it buys time for the service provider to implement a fix into the web application, which can take from days to several months to implement. Firewall vendors usually roll these patches out automatically as a service. Although virtual patching provides a quick fix for a vulnerability, a more complete software patch is recommended because the virtual patch might not provide total protection. It is a greater risk to rely on the virtual patch than an actual patch or both of them.

2.3 Other security measures of WAFs

In addition to preventive features there are supportive features typical for WAFs that help administrators operate them. They are logging, reporting, and session tracking as explained in the following subsections.

2.3.1 Logging

Viewing log messages is an important feature in WAFs for maintaining and monitoring the system. There are two types of relevant logs: system logs, and traffic logs. A system log contains information on what events have occurred in the system, like for example what options have been changed or if there are any errors or warnings that require attention. Traffic logs are for auditing user and server packets. They are helpful when configuring and testing the WAF but they can also be used to examine attempted cyber-attack activities. Logging requests that have certain special characters or words might be insightful because the most prevalent attacks use them.

2.3.2 Reporting

Reporting or monitoring is a feature where a WAF shows graphical data on a timescale. These reports can show for example the hourly distribution of users or bandwidth. They can be used for analyzing security threats – there could be unusually high amount of traffic going on which might indicate that a DDOS attack is in progress. Reporting could also show the rate of attacks made.

2.3.3 Session tracking

A WAF can track user session by IP address, HTTP session or username. It is possible to force a login page the user must pass through to gain access to the application. This prevents forceful browsing. WAFs can also perform logging or blocking actions against users that exceed a threshold of violations.

2.4 WAF Security Models

There are two types of security models that can be used for screening normal behavior from abnormal. These are negative security model and positive security model, also known as blacklisting, and whitelisting respectively. The negative security model is based on the idea that we know what traffic is malicious and we want to block that traffic and everything else is passed through. The positive security model defines what kind of traffic is desirable and blocks everything else.

2.4.1 Signature Based Blocking

The negative security model relies on attack signatures, which are unique arrangements of information. This can be illustrated by the following example. An attacker tries to get user information from a web application by inserting this input into a username-field:

```
' OR '1'='1' –
```

thus causing an SQL injection attack to occur. A simple attack signature here could be

```
'1'='1'
```

which would recognize the input as an attack. Knowing that an attacker could replace the 1's with 2's or any other number so rather than having such a static signature a regular expression (regex) is often used [8]. A better attack signature using regex would be:

```
'\d+'='\d+'
```

The special sequence "\d+" would match any digit one or more times. The disadvantage with regex is that it requires more resources than string matching in both memory and processing power.

There are two disadvantages to signature based WAFs: they are resource intensive, and they cannot prevent new attacks unlike anomaly based WAFs. Because a blacklist-based system has to account for every known vulnerability and compare signatures to every network packet, it requires a lot of resources on high web traffic systems. Whitelists tend to be smaller than blacklists and they are capable of blocking unknown attacks as well, but they are more complex and more laborious to set up. The next subsection will explain why in more detail.

2.4.2 Anomaly Based Blocking

The positive security model constructs a ruleset of acceptable message content, like for example "a URL path can only contain alphanumeric and characters from group {&,?,/}" and "a value with the name 'username' can only contain alphanumeric characters and spaces". Thus the packet containing the SQL-injection attack from the previous subsection (2.4.1) would be dropped because it contains "illegal" characters. The more restrictive nature of whitelists causes false positives in higher rates than blacklists.

Because whitelisting only lets safe messages through it has the benefit of blocking zero-day vulnerabilities. Since web applications are all different they must have their own whitelists. Although possible, it is not practical to generate whitelists manually considering that web applications can be large and in constant development. Therefore, various learning methods have been developed to make whitelist rule generation easier: continuous learning and triggered learning. [9]

In continuous learning method the WAF collects data constantly and generates rules and performs adjustments based on the data on longer periods. If the users' behavior changes over time, there may be a larger amount of false positives in the transitional stage. This method is susceptible to attacks that target this learning method of the WAF because the attacker can make the WAF learn to accept malicious packets over time. This method requires less manual input than triggered learning and is more suitable to web applications that are not developed continuously. [9]

In triggered learning method the learning phase is triggered by the administrator and desirable traffic is generated manually or programmatically. For the latter case there are tools which can simulate humanlike traffic in large quantities. Once the learning phase is terminated, the WAF generates rules based on the traffic and is then switched into working mode where it blocks all anomalies. This method requires less traffic than continuous learning but the learning phase has to be partially redone when there are changes to the application. [9]

2.4.3 Security Model Summary

The negative security model is easy to setup since attack signatures are usually provided by the party that distributes the WAF but unless they run on dedicated hardware, they take a slice of the web server's resources especially when traffic levels are high. Table 2 summarizes the advantages and disadvantages.

Table 2. Summary of security model pros and cons.

Security model	Pros	Cons
Negative	• Easy to implement • Universal; can be applied to all web applications	• Resource intensive • No zero-day protection
Positive	• Resource light • Protects from zero-day vulnerabilities	• Requires machine learning to implement • Requires unique rules for web applications

3 XMPP

The extensible messaging and presence protocol is a protocol that was designed for instant messaging but because its presence capabilities and extendibility it has been adopted to machine-to-machine (M2M) for near real-time interaction. It is based on XML and has been used in a variety of chat implementations on the Internet.

3.1 Relevance to IoT and WAF

XMPP has been suggested as a good communications protocol for internet of things [10-13]. It is an open standard and software using it can be licensed with any software license, even commercial ones, which makes it suitable for researchers, the public sector, commercial companies and technology hobbyists. XMPP is standardized and extensible which means that anyone can propose an extension protocol and as a result it could be implemented in further XMPP applications if suitable.

Many XMPP extensions for internet of things are currently in early stages of development [14]. They aim to provide XMPP with functionality that is either missing or enhances its IoT-capabilities. Table 3 lists all the current IoT XEPs (XMPP Extension Protocol). Perhaps the most important XEP in the list is XEP-0322 because it deals with one of the issues that XMPP has – message size [11]. It would greatly reduce network overhead and memory footprint with serialization in binary code and by utilizing schemas.

As XMPP was originally developed for chatting, and because people are very familiar with chatting applications, it improves the H2M (Human-to-Machine) communication by providing a familiar interface. People are able to organize internet-connected devices in their contact lists, see their presence and status information, and subscribe to things they are interested in. The publish-subscribe model (as opposed to request-response) allows for filtering and displaying relevant information for users in near real-time. [13]

Table 3. XEPs for IoT [15].

XEP	Description
xep-0000-IoT-BatteryPoweredSensors	Defines how to handle the peculiars related to battery powered devices, and other devices intermittently available on the network.
xep-0000-IoT-Events	Defines how Things send events, how event subscription, hysteresis levels, etc., are configured.
xep-0000-IoT-Interoperability	Defines guidelines for how to achieve interoperability in Internet of Things, publishing interoperability interfaces for different types of devices.
xep-0000-IoT-Multicast	Defines how sensor data can be multicast in efficient ways.
xep-0000-IoT-PubSub	Defines how efficient publication of sensor data can be made in Internet of Things.
xep-0000-IoT-Chat	Defines how human-to-machine interfaces should be constructed using chat messages to be user friendly, automatable and consistent with other IoT extensions and possible underlying architecture.
XEP-0322	Defines how to EXI can be used in XMPP to achieve efficient compression of data. Albeit not an Internet of Things specific XEP, this XEP should be considered in all Internet of Things implementations where memory and packet size is an issue.
XEP-0323	Provides the underlying architecture, basic operations and data structures for sensor data communication over XMPP networks. It includes a hardware abstraction model, removing any technical detail implemented in underlying technologies. This XEP is used by all other Internet of Things XEPs.
XEP-0324	Defines how provisioning, the management of access privileges, etc., can be efficiently and easily implemented.
XEP-0325	Defines how to control actuators and other devices in Internet of Things.
XEP-0326	Defines how to handle architectures containing concentrators or servers handling multiple Things.
XEP-0331	Defines extensions for how color parameters can be handled, based on Data Forms (XEP-0004)
XEP-0336	Defines extensions for how dynamic forms can be created.
XEP-0347	Defines the peculiars of Thing discovery in Internet of Things. Apart from discovering Things by JID, it also defines how to discover Things based on location, etc.

Like the other messaging protocols, XMPP is not without its drawbacks. As mentioned

earlier XMPP has a large message size that results from its verbosity and text based communication. There is no support for Quality of Service in the protocol itself and it has to rely on TCP's (Transmission Control Protocol) methods. These drawbacks have been addressed in extension protocols but they have not yet completed the full approval process.

XML and HTML are based on the same language called Standard Generalized Markup Language or SGML. Because they are similar and because XML is sometimes used in the web for representing arbitrary data structures for example in an AJAX (Asynchronous JavaScript and XML) requests, web application firewalls are able to read it.

The fact that XMPP is essentially XML makes it possible for WAFs to inspect and modify XMPP messages. It enables a WAF to protect an IoT device in a similar manner as it would protect a web server from malicious users. Especially on low powered sensory devices it is important to prevent the so called sleep deprivation attacks which cause the battery to drain quicker than normal leading to denial of service eventually. A WAF could also perform SSL offloading on behalf of IoT devices that do not have SSL acceleration or SSL capability so that a secure connection would be possible.

Since XML documents can be well defined there is a technology called schema that defines what an XML document should be like. For example, a schema could describe what elements a document should have and what attributes they should have and so on. A WAF could easily enforce that the data moving through it is according to the schemas since they are available on the internet. It would of course require that these schemas are followed in the first place.

3.2 XMPP Messaging

The basic units of communication in XMPP are called stanzas. There are three types of stanzas, all of which are used for different purposes and are processed differently. The stanzas are as follows:

- **Message** is a push type communication unit similar to email. As the name suggests,

it can be used for sending messages, but also for notifications or alerts. Errors are also sent as messages. A message does not necessitate a response from a server or a client.

- **Presence** is a publish-subscribe type method where the user can see the status of people on their contact list, also known as roster. The server keeps a record of everyone's presence status. Clients can subscribe to each other so that they can receive near real-time presence information.
- **Iq** is a request-response type communication method that has four methods: get, set, result and error. A get or set –request must always be responded with result or error and request-response pairs must always have the same id-value that differentiates them from other pairs.

XMPP connection is persistent and uses TCP to transmit messages. Persistency in this case means that the connection allows the client or the server to push messages without initiating a new connection. When the client connects to the server it initiates the connection with a `<stream>`-tag and the connection stays on until the client disconnects either by sending a `</stream>`-tag or by closing the socket.

The following example in figure 6 from [16] demonstrates a brief exchange of XMPP stanzas as well as connection begin and end. The messages from client are prefaced with "C:" and from server with "S:". In this example the client requests its roster, sends a message to "madhatter", receives a message from "king", and then leaves. The client also indicates its availability with the `<presence>`-tag, which all the subscribed clients then receive from the server.

```
C:   <stream:stream>

C:     <presence/>

C:     <iq type="get">
         <query xmlns="jabber:iq:roster"/>
       </iq>

S:     <iq type="result">
         <query xmlns="jabber:iq:roster">
           <item jid="alice@wonderland.lit"/>
           <item jid="madhatter@wonderland.lit"/>
           <item jid="whiterabbit@wonderland.lit"/>
         </query>
       </iq>

C:     <message from="queen@wonderland.lit"
                to="madhatter@wonderland.lit">
         <body>Off with his head!</body>
       </message>


S:     <message from=”king@wonderland.lit”
                to="party@conference.wonderland.lit">
         <body>You are all pardoned.</body>
       </message>

C:     <presence type="unavailable"/>

C:   </stream:stream>
```

Figure 6. Example showing client-server interaction [16]

3.3 BOSH

When a client cannot initiate or maintain a long-lived bidirectional TCP connection due to network restrictions or otherwise, there is an alternative protocol that utilizes the HTTP. It is called Bidirectional-streams Over Synchronous HTTP or BOSH for short. [17] It is very convenient because it can be configured to work over the TCP port 80, which is one of the most used internet ports so firewalls usually keep it open.

BOSH emulates a long-lived, bidirectional stream by using multiple synchronous HTTP connections without requiring the use of frequent polling or response chunking. It is designed to transport data efficiently and with minimal latency in both directions. BOSH achieves this by utilizing long polling with multiple synchronous HTTP request/response pairs. BOSH is otherwise similar to XMPP, but it wraps each message in an HTML body-element with session and request IDs. Session ID manages the authenticated session lifetime and request IDs keep track of the right order of messages. [17]

4 LITERATURE REVIEW

The motivation of this chapter is to chart what research have been done on web application firewalls. In the first section, a metasearch is conducted to find out how the research on WAFs has grown and what the main publishers of articles are. Then the main research categories are mapped out and further expanded upon. The next chapters would have included the examination of XMPP or XML in the context of web application firewalls but no such scientific research was found. The reason for this might be that their attack vectors are not at the level where they are interesting to the scientific community.

4.1 A metasearch

To establish a base for WAF research, a metasearch was conducted. The site nelliportaali.fi [18] was used with a search keyword "web application firewall", in quotation marks to catch the term explicitly, and then the keyword was set to match any field. The common abbreviation "WAF" was not used as it can also refer to web application frameworks. The following databases were selected for the search: ACM, Emerald Journals, IEEE, SpringerLink and Wiley Blackwell Online Library. The results have been collected into tables 4 and 5 and visualized in figure 7. Table 4 shows articles by databases and table 5 shows articles annually with the rest of 2016 projected by linear extrapolation. This data was collected on 18th of August 2016. The data included journal articles and book chapters, which was a limitation of the search engine.

We can see from the data that IEEE and SpringerLink had the most articles on WAFs. Perhaps their published journals are more connected to the security and networking fields or maybe they have more articles and journals on Computer Science. The data also shows that WAF research has begun in 2005 at the latest. For reference the first version of ModSecurity, the most popular open source WAF, was released in the late 2002 [19].

Tables 4 and 5. Results of a database metasearch using the keyword "web application firewall". Table 4 (on the left) has the results grouped by database name and table 5 annually. The projection column is an estimation of yet to be published articles this year.

Database	Results
ACM	0
Emerald Journals	3
IEEE	101
SpringerLink	92
Wiley Blackwell Online Library	4

Year	Results	Projection
2005	3	
2006	1	
2007	4	
2008	10	
2009	13	
2010	15	
2011	21	
2012	23	
2013	23	
2014	29	
2015	40	
2016	17	10

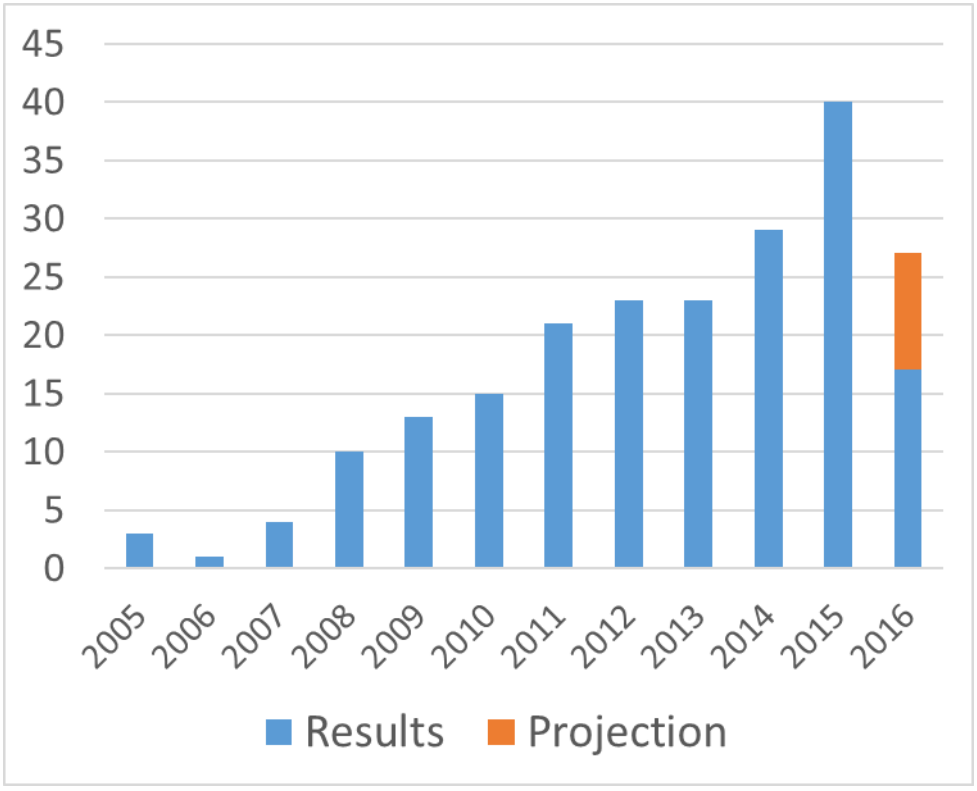

Figure 7. Metasearch data as a chart.

Since 2006, web application firewall research seems to have been growing steadily and will probably continue its growth. Internet security has been a rising trend and WAFs play a big part in mitigating cyber threats. According to Symantec's 2016 Security Report [20] crypto-ransomware was up 35% in 2015 from 2014, there were 36% more new malware variants and in 2015 there were nine breaches where more than 10 million identities were exposed. Zero-day vulnerabilities more than doubled (+125%) between 2014 and 2015 with 54 new vulnerabilities, ten of which were related to Adobe® Flash® Player. Symantec estimates that the Flash® Player will gradually fall out of common usage over the year 2016. Akamai Technologies has similar figures on their Q1 of 2016 "akamai's [state of the internet] / security" report [21]. Comparing the first quarter of 2015 and 2016, DDoS attacks have risen 125% and mega attacks (greater than 100 Gbps) 137%. As figure 8 shows, all web application attacks from Q4 2015 to Q1 2016 have grown in quantity except for PHP injection attacks [22].

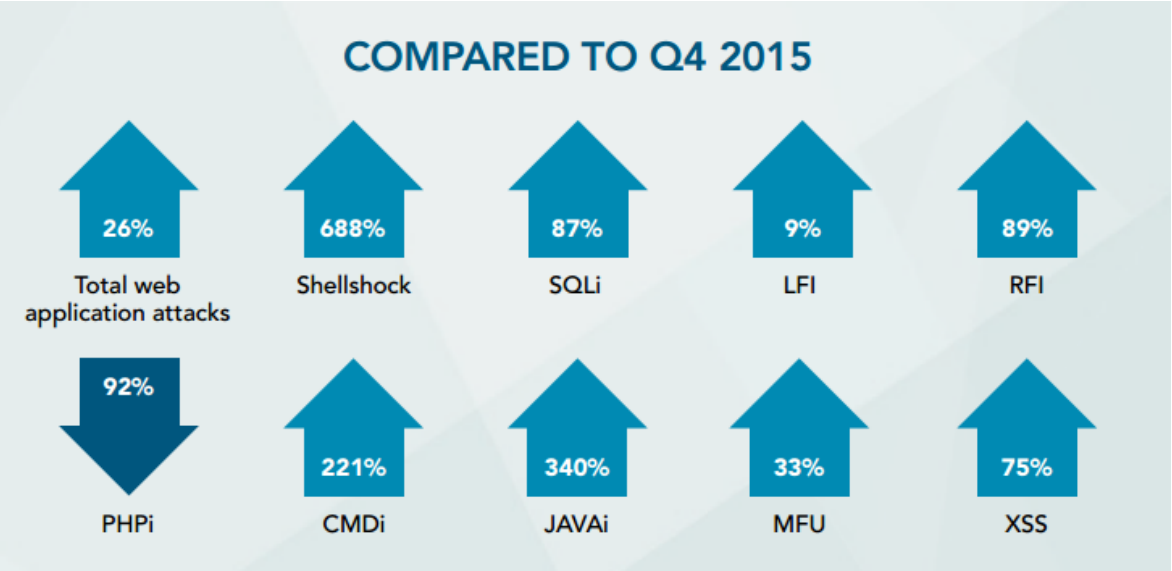

Figure 8. Web application attacks have risen from Q4 2015 to Q1 2016 [22].

As to why the year 2016 projected article releases are much lower than 2015, there might be many reasons. The sample size in the metasearch is low therefore variation is possible. There might also be a period of time until a published article appears in the metasearch in which case the extrapolation is inaccurate. There is no reason to draw any conclusions to current WAF research popularity based on the metasearch because of the unknown factors.

Some of the research on WAFs crosses with research on IPS's (Intrusion Prevention System) so it would be advisable to know some of the differences and similarities of these systems. IPS is an inline packet inspector that has the ability to drop packets or terminate connections. Thus it relies mainly on identifying attack signatures and has no understanding of web application protocol logic like WAFs do. Signatures are less flexible than WAF rule-systems that can for example restrict attribute value lengths. Below is a table from [23] that compares the feature list of WAF and IPS. As can be seen from table 6, IPS features are very limited compared to WAF. [23, 24]

Table 6. Comparing IPS and WAF features [23].

Security feature	IPS	WAF
Injection attack protection (XSS, SQL)	Limited	Yes
CSRF protection	No	Yes
Normalize encoded traffic	No	Yes
Inspect HTTPS traffic	No	Yes
Session tampering/hijackin/riding protection	No	Yes
Forceful browsing prevention	No	Yes
Data theft protection, cloaking	No	Yes
Brute-force protection	No	Yes
Web services projection	No	Yes
Virus/malware upload protection	Yes	Yes
Application layer DoS protection	No	Yes
Rate control protection	No	Yes
Request, response rewrite	No	Yes
Application access logging and user audit trails	No	Yes

4.2 Blacklist and whitelist algorithms

The majority of research work in WAFs is concentrated on creating and improving algorithms to identify and prevent cyber-attacks. They are further divided into two categories: improving attack signature patterns to be more comprehensive and/or efficient and self-learning algorithms. The largest research motivation in this category is to prevent SQL injections and quite justifiably so. Injections have been the number one threat according to OWASP Top 10 in both 2010 and 2013 surveys. The next survey will be

published some time in late 2016 or early 2017.

A good example of new algorithms for detecting injection attacks can be found in “SQL Injection Attack Detection Method Using the Approximation Function of Zeta Distribution” [25]. This learning algorithm creates a zeta distribution profile for user submitted strings and tries to determine if it is harmful or not. For example, in normal data the symbol “SP”, also known as “space”, is the most used symbol but in injection strings the most used symbol is the asterisk, or “*”. Some training data is required to create distinct categories for profiles.

The results of the study showed that the detection rate of harmful strings was 96.1% based on the test data. It performed well on strings of the type “personal information” (99.3%) but for some string types that contained a lot of symbols the performance was suboptimal with results like 93.8% for “mathematical formulas” and 92.3% for “other” types. This algorithm could be useful if it performed better on symbol abundant strings. Additionally, the detection rate of the normal test data was 98.3%, which means that the rate of false negatives was 1.7%, which sounds really high. Character distributions can also be tampered with if the attacker knows that a defense system is analyzing his input. The paper did not address this issue. [25]

There is an interesting study by Bremler-Barr and Koral [26] which shows that it is faster to do signature matching on gzip-compressed (GNU zip) data than normal uncompressed data even though a decompression has to be performed. This is also the first solution that performs multipattern matching “on the fly” for compressed data.

4.3 Distributed systems and EDoS Armor

The cloud computing environment is quite different from the traditional web application environment where the server would physically exist near the application developers and therefore all of the systems would be configurable. The cloud service vendors provide often abstracted data on their systems because the customer does not need to know about

the details and disclosure of specifics can lead to getting more interest from malicious parties.

Despite its restrictions, the cloud datacenter solutions are attractive especially for web entrepreneurs and small to medium size businesses because of their pay-as-you-go model and scalability. Since smaller e-Commerce applications are often built on top of cloud services from different vendors, they are vulnerable to DDoS-attacks. If auto scaling is turned on, the attack not only causes losses of sales or ad revenue, but also the cost of the service increases thus causing an Economic Denial of Sustainability (EDoS). [27]

In [27] the authors created a system for preventing EDoS called EDoS Armor. The system is quite sophisticated and has five different modules: Challenge server, Admission control, Congestion control, Classification of users, and Priority calculation. The architecture of the EDoS Armor is demonstrated in figure 9.

Before a user can access the destination server it has to respond to a challenge issued by a Challenge server to distinguish it from illegitimate users such as bots. The challenge can be cryptographic or image based. Once the challenge has been completed Admission control issues the client a unique "hide port" through which it can access the web server. All of this is done transparently with JavaScript to not disturb the user. Congestion control then divides the users into good and bad clients and serves the good ones with higher priority, meaning more bandwidth and shorter response time. The request rates are also filtered at the IP-level, so that if a client starts requesting too many resources per time unit it gets dropped and blocked. EDoS Armor then continually classifies the user by which application routes it traverses. Variables like client's purchases, request processing time and session information affect the classification of the user. Good behavior leads to better classification. After classification the client priorities start updating based on browsing behavior. The new priority is based on the old priority and is additive and multiplicative thus priority rises constantly on good behavior and diminishes on bad behavior. [27]

EDoS Armor is a promising tool based on its results, but the research paper does not

address its scaling [27]. It was mentioned that it runs on a reverse proxy server in front of the web application but can it serve multiple clients? The paper also mentioned that ports are forwarded with NAT (Network Address Translation) forwarding, which makes it unclear if this system should be deployable by the cloud server customer or the vendor since NATs are network middleware.

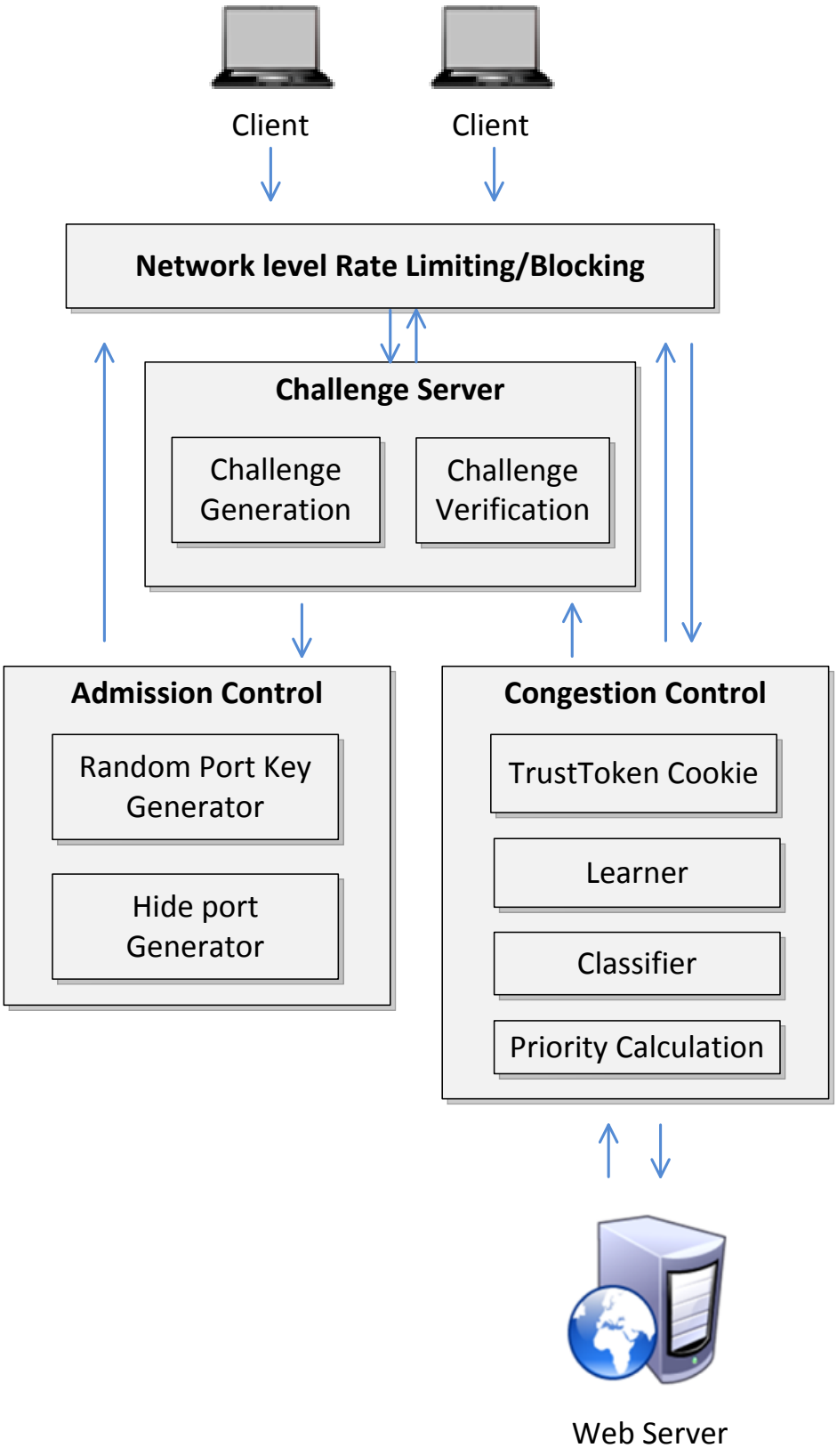

Figure 9. Architecture of EdoS Armor [27]

4.4 Comparison of existing WAFs

There are very few studies that compare WAFs or measure their effectiveness. Among them is this study [28], which compares the three most popular open source WAFs: ModSecurity by Trustwave SpiderLabs [29], AQTRONIX WebKnight [30] and Guardian@JUMPERZ.NET [31]. It concluded that ModSecurity was the best one of them, generating less false negatives than WebKnight and blocking more attacks than Guardian.

The study was a bit limited because all the WAFs were used with their default settings except for ModSecurity where the "Base" OWASP rules were installed. WebKnight seemed to block all POST-requests affecting the results significantly. In retrospect the study measured the default configurations of three different WAFs more than it measured their capabilities. How a WAF is tuned has a significant effect on its performance as the study "Estimates on the effectiveness of web application firewalls against targeted attacks" [32] discusses. There were four countermeasures that increased the effectiveness of a WAF based on expert knowledge. Those were WAF operator experience, the effort spent on tuning the WAF, automated black box testing tools, and whether an operator was monitoring the WAF.

4.5 New WAF Implementations

Several new WAF implementations have been suggested by the research community [33-35]. They appear to be quite experimental and have not been refined later to my knowledge. The first one [33] is implemented as a plugin for WebScarab, which is a java-based web application testing tool from OWASP that intercepts and can alter HTTP requests. This WAF blocks control flow tampering attacks targeted at a web application. Control flow tampering means that a URL is requested in the wrong sequence which might result in an exploitation. To prohibit attackers from control flow tampering the WAF has to build a dependency graph that represents the relation of web pages and later on when someone tries to access a page from the wrong location the request is blocked. The difficulty of building an accurate dependency graph increases as websites become more

dynamic in their content and when there are changes to the structure of the web site.

To benefit from both, the speed of signature based detection and the ability to prevent zero day attacks with anomaly detection, Tekerek et al. [35] have developed a hybrid web application firewall. The WAF first checks if a request contains any malicious signatures and if it doesn't it proceeds to anomaly detection. The latter is accomplished by statistical methods since it is known that malicious attacks usually have certain properties that normal requests do not have, anomalous requests are less frequent than requests generated by normal user traffic, they are longer than normal requests because they usually contain attack vectors which can be quite lengthy, and their letter frequency is also different. By combining these three analyses it can get an anomaly rating.

5 IMPLEMENTATION AND EVALUATION

This chapter describes the implementation of a ModSecurity plugin and discusses the problems and issues that were conceived during the implementation. The resulting plugin is evaluated.

5.1 Goal

The goal of the practical part of this thesis is to engineer a proof of concept web application firewall for the XMPP protocol, which is one of the more widely used internet of things technologies. It will be developed for the home environment. Because of time constraints, the WAF will be implemented as a plugin to the popular ModSecurity, which already has an XML processing engine. ModSecurity is also the most widely used open source WAF at the moment and its documentation is comprehensive.

The purpose of this WAF is to filter harmful network traffic and to prevent different types of cyber-attacks. The following features should be implemented:

- DOS prevention
- Geoblocking
- Session tracking
- Request ID tracking
- Application level compression prevention
- Optional configuration by user
- And dropping packets without dropping the connection.

It can be tempting to implement features in a WAF that overlap with the functionality of the XMPP server. There is some filtering that is more beneficial to be done on the WAF end to not burden the XMPP-server. In this implementation the WAF checks if request XML is “well formed” and drops non-valid requests because they could be used in Denial of Service –attacks.

5.2 Method

The ModSecurity plugin has been created as set of rules in one or more Apache configuration files similar to how the OWASP ruleset is implemented. The ModSecurity rule engine provides a well-documented and powerful framework for inspecting, altering and blocking web requests and responses. There is also an option to execute Perl scripts in case more flexible scripting language is needed but this implementation did not use Perl scripting.

DOS prevention works by keeping a cumulative count of requests from each IP address. When the count reaches a certain amount requests per minute, all the consecutive requests are dropped. The IP address is allowed to make requests again once the ban timer has expired. These rules effectively prevent DoS attacks from a single source but do not handle Distributed DoS very well.

Geoblocking was implemented with ModSecurity's Geolookup-methods. `SecGeoLookupDb` reads a geolocation database into the memory, which contains two rows of data: an IP address and a country code. Every request's IP is compared to the list and on a match, it is assigned a country code. IP Addresses can thereafter be blocked based on these country codes.

Because IP addresses are subject to change, it is recommended to update the geolocation database once a month. This can easily be automated in Unix-based systems with a scheduler tool, such as Cron. The database used in this implementation is MaxMind's GeoLite database that can be found in [36]. This WAF blocks the ten countries where most DDoS attacks originated from in the first quarter of 2016 according to the content delivery network provider Akamai Technologies [22]. An example rule that does geoblocking can be seen in figure 10.

Session tracking in XMPP over BOSH is easy, since after establishing a session, it is always included in every message's body-tag. Session tracking enables the WAF to block a session if the client violates the communication. It also allows the WAF to restrict some

features if the client has not begun a session.

```
# China
SecRule REMOTE_ADDR "@geoLookup" \
"id:32100,\
drop,\
tag:geoblocking,\
chain"
    SecRule GEO:COUNTRY_CODE "@streq CN"
```

Figure 10. Example of a geoblocking rule.

The request ID tracking feature is an important one because it enables the WAF to drop arbitrary requests from the client without disrupting the connection. To keep track of the right order of client messages the protocol defines that the client must include a sequential Request ID in its messages, except for empty keep-alive messages. There is a specific window of values the server accepts that are above the current Request ID. That window's size is the same as the simultaneous requests allowed. If the Request ID is above or below this window, the connection manager breaks up the connection.

When the WAF wants to drop a request from the client it has to keep an internal count of the Request ID because the client just keeps incrementing the RID, but the when the WAF drops a request from the client, the RID that the server expects is no longer the same as what the client is sending. Figure 11 illustrates how the WAF can suppress a client request without disrupting the connection. In this example when the client makes a request for compression with some specified method, the WAF changes the message into an "empty body" message. When the connection manager responds with its own empty message, the WAF modifies it into "unsupported method" response. The WAF has to record the amount of messages it has dropped from the client because in the future it has to decrement the RID by that amount because the connection manager disconnects otherwise.

A WAF administrator can customize the ruleset by changing values in the "xmpp-0-setup.conf" file included in the plugin. He can disable geoblocking and set the amount of messages per minute after which the client gets blocked. He can also determine how many seconds the client is blocked for.

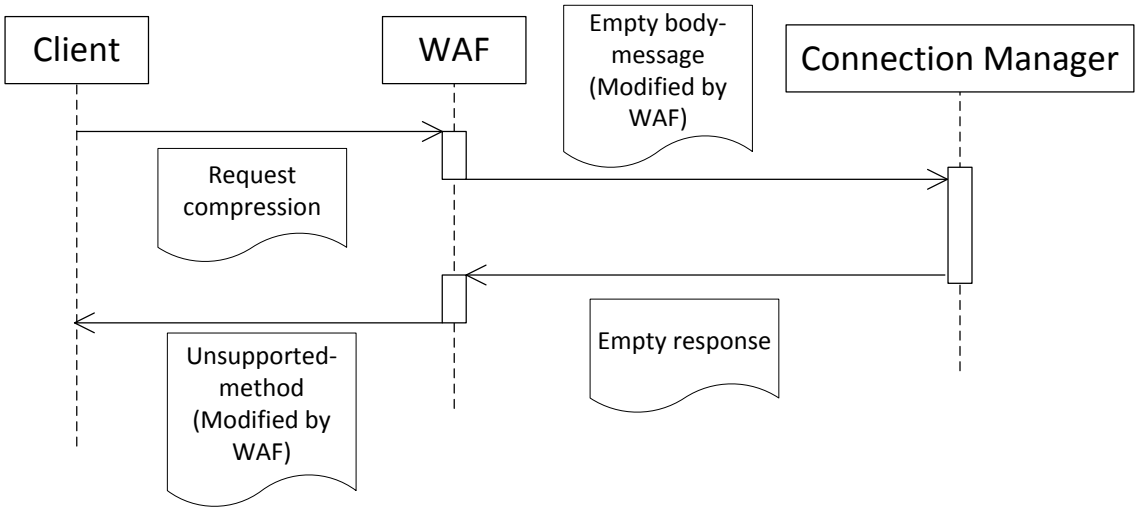

Figure 11. WAF suppresses the compression request of the client.

5.3 Complications

Some features of ModSecurity version 2.9.0 are still unstable which caused some complications during the development of this plugin. There were two problems with ModSecurity which caused the XML parsing engine to be ineffectual in the case of XMPP.

The first complication was that the ModSecurity's XML processor libxml2 does not work well with default namespaces at least in conjunction with ModSecurity. Default namespaces are present in nearly all XMPP messaging which renders the XML engine useless.

The second problem was identified when trying to bypass the first problem. ModSecurity has a way of modifying the request body. If all of the default namespaces could have been converted temporarily into normal namespaces, then the XML processing engine could have been utilized. Unfortunately it was discovered that the message could not be passed to the XML engine after modification.

The aforementioned problems meant that the XML engine was unusable in this plugin until the default namespaces could be handled correctly by ModSecurity in the future

versions. Because XML parsing is not available, the plugin must utilize regex to locate tags, their attributes and contents in the message body. Using only regex has the disadvantage that it cannot utilize XML schema validation. XML schemas define how XML documents are constructed and what tags are accepted. This kind of validation could have made some types of exploits obsolete but would have induced heavier CPU load.

5.4 Testing and debugging the WAF

Confirming that the WAF works as intended is no easy task because there is no clear indication. Therefore ModSecurity's log-level should always be high during rule development, which unfortunately also increases the amount of unneeded messages. To remediate the scarcity of a lower log level one can attach debugging messages to rules, which are output to error log. The debugging messages allow the developer to determine which rules were activated and what were the values of variables at the time of execution.

The development on the WAF was done in a manner similar to test-driven development. Every rule was implemented and tested separately so that coding errors could have been detected as soon as possible. To further verify that the plugin worked when all rules were implemented and turned on, an XMPP client was run and its features were tested to be working correctly. Geoblocking was not testable in the testing environment but it was configured according to the documentation.

5.5 Discussion

During the thesis work, it occurred that it is quite difficult to shield an application from targeted attacks. With a large botnet, an attacker can render a server useless for several days if he really wants unless the defender is using some kind of cloud service for preventing DDoS. But those services cost money so in the end outcome of the attack depends on which party is willing to use more resources.

Restricting access from certain countries with high cyber crime activity can also alleviate

the risk of ending up as a victim. Because a big portion of cyber attacks originate from a handful of countries, Geoblocking is a good preventive measure that stop vulnerability scanning from blocked countries. It should be mentioned that a hacker from a blocked country can easily bypass such restrictions because of virtual private networks (VPN). They act as kind of tunnels that mask where the traffic is originated from. Therefore geoblocking acts as a good camouflage but once the enemy has noticed you, the cover is gone.

5.6 Results and evaluation

The result of this study is a working proof of concept web application firewall for XMPP that implements such features as DOS prevention, geoblocking, session tracking, request ID tracking and compression prevention, and is configurable by the user. It was implemented as a plugin for ModSecurity. This proof of concept shows that it is possible to implement WAF functionality with existing tools that operate on top of HTTP.

The WAF system can be evaluated with the help of OWASP's "Web Application Firewall Evaluation Criteria" [3]. The system that is evaluated consists of the following configuration: Ubuntu 16.04 operating system and Apache web server with mod_proxy and ModSecurity installed. The intended use for the system would be for home usage, which is also the context during the evaluation. According to OWASP a full evaluation would require a skilled technician, therefore an applied, more lightweight evaluation is produced in this study.

The evaluation will look at each section described in the OWASP's document and provide a summary for the reader on the areas that are covered by the system and on the areas that are deficient according to the context that was defined in the previous paragraph.

5.6.1 Deployment architecture, performance and protocol support

The system is deployed as a reverse proxy, which operates as an inline network device. It can intercept requests from clients and the HTML rewrite functionalities are extensive. It

can be configured to terminate SSL traffic but not conveniently. Overall, the deployment architecture is sufficient but requires a dedicated server and someone who knows how to install and configure software on Linux-based systems.

As the OWASP evaluation criteria document says, the issue of performance evaluation is difficult. It can be said though that software-based systems like this one perform worse than a WAF that is delivered as an appliance. Appliances can utilize optimized hardware and specialized hardware components. For home usage, the performance should be sufficient because of the amount of network traffic generated.

The system has good support for HTTP specifications and it can restrict what HTTP-methods are allowed, although XMPP over BOSH only requires the POST-method. It has the ability to restrict requests by element content, length and byte range. The system handles HTTP and HTML support well. The system performs XML validation but otherwise the XML processing engine has problems with anonymous namespaces as described in subsection 5.3. Other XML features are not relevant for the system.

5.6.2 Detection and protection techniques

The detection system uses the negative security model, which is not as desirable as the positive security model. It is a rule-based system and it can be configured to update new rules from an internet database. The WAF has great content normalization methods and extension APIs (application programming interface) [37]. For a negative security model based system, it has extensive detection features but a positive security model would be easier to use and would also stop zero-day attacks.

The system protects against brute force and DOS attacks by blocking an IP that makes requests too frequently and prevents some distributed attacks with geoblocking in cases where the bot nets are located in blocked countries. There is no session attack mitigation, but in the context of XMPP there is not much that a WAF can do other than encryption. When it comes to protection, the system can deal with random attacks but it cannot protect

from a determined attacker who targets the system.

5.6.3 Reporting, management and logging

The two of the most lacking areas in the system are the lack of graphical user interface and lack of reporting, which is hard to implement without the former. Any reporting functionality would have been desirable because without it the user has no knowledge of any suspicious activity or how much traffic throughput there is. The lack of graphical user interface also makes the management of the system difficult because it requires some technical skill from the administrator. Management is performed by editing different configuration files in the system, which are not all located in the same directory. The administrator can also make his own rules but knowledge of the ModSecurity API is required. There is some granularity within the configuration options, as for example the geoblocking or blocking in general can be turned off (this is called detection only mode). The management capabilities in the system are very deficient and require technical skill.

The system has configurable transaction logs, access logs and error logs with unique transaction IDs and ModSecurity can utilize logging on a separate server. Filtering sensitive data out of the logs is supported but not easily configurable. Log retention options are also limited and the system does not discriminate between suspicious and non-suspicious transaction. Transactions that matched a rule are written on separate logs that can be preserved for longer periods of time. Overall, the logs in the system are not very accessible although they are thorough, and configuration options are somewhat deficient.

5.6.4 Evaluation summary

The system has good HTML and HTTP support and it handles detection well, although it does not utilize the positive security model. The areas that it could improve upon are the deployment architecture, protective measures and logging. On these areas, the system is difficult to use, require technical skill and does not provide a graphical user interface. The system was the most deficient on reporting and management.

6 CONCLUSION

The goal of this thesis was to develop web application firewall filtering capabilities for an internet of things protocol as a proof of concept. The motivation for this was for the author to learn IoT and WAF technologies. ModSecurity was chosen as the firewall because it is open source, it is popular and it has a powerful application programming interface. XMPP was chosen as the protocol because of its textual presentation and verbosity. The firewall was implemented as a plugin since the development time was limited. The two of its main features are geoblocking and prevention of small-scale DoS attacks.

For a clear picture of the technologies that were present in the thesis, the principles of WAF and XMPP were introduced to the reader and the trends and areas of WAF research were recognized. The starting point was to analyze the functionality of firewall technology by examining its different components. The components were categorized as preventive security measures and other, which could have as well been named as supportive measures. The two security models, signature based blocking and anomaly based blocking were discussed and their pros and cons were summarized. Furthermore, the XMPP was identified as a good but yet developmental protocol for enabling machine-to-machine internet of things communication. The strengths of XMPP are that it is actively developed for IoT capability, it is an open standard and extensible. Its drawbacks are that the message size is large and that it has no innate quality of service features. Both of those are addressed in future protocol extensions. On the literature review, the metasearch shows that since 2005 the research on application firewalls has been trending upwards up until 2016. It is uncertain if the trend will continue in the future but judging from the historical data it is likely. A wide range of problems was found on WAF research but no XML research was found on the WAF context.

In the implementation section, the web application firewall plugin was implemented. During the implementation there were some issues with the XML parsing and in the end the complications prevented the usage of the XML parser. Instead, regular expressions were used. The plugin was evaluated with the help of an external evaluation document. The evaluation pointed out that technical skills are required for deploying and using the

WAF plugin and that some functionality like reporting and graphical user interfaces were missing. The detection and logging features were extensive. When it comes to the future development of the plugin, the first priority should be fixing the broken functionality in ModSecurity's XML library. As an alternative, other web application firewalls could also be evaluated for better support on XML and XMPP. More generally, as the XMPP usage rises in the future as IoT becomes more prevalent there are bound to be new vulnerabilities that need virtual patching.

In the future, the web application firewall research should branch out on the internet of things realm. IoT devices played a big part in DDoS attacks last year and we are only in the beginning of the IoT era. When IoT devices use mobile connections, like many use today, they are not behind a firewall, which makes them more vulnerable. Therefore, solutions that encrypt and filter data need to be introduced in the consumer market space.

REFERENCES

1. Akamai's State of the Internet Security Team, "'Q4 2016 State of the Internet – Security Report," [online]. Available: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2016-state-of-the-internet-security-report.pdf [Accessed 5th April 2017].
2. Frost & Sullivan, "High-Profile Data Breaches Demand Web Application Firewall Adoption," [online]. Available: http://ww2.frost.com/news/press-releases/high-profile-data-breaches-demand-web-application-firewall-adoption/ [Accessed 14th July 2016].
3. Web Application Security Consortium, "Web Application Firewall Evaluation Criteria," [online]. Available: http://projects.webappsec.org/w/page/13246983/WAFEC%201%20HTML%20Version [Accessed 29th June 2016].
4. The Open Web Application Security Project, "OWASP Best Practices: Use of Web Application Firewalls," [online]. Available: https://www.owasp.org/index.php/Best_Practices:_Web_Application_Firewalls [Accessed 30th June 2016].
5. J. Pubal, "Web Application Firewalls," [online]. Available: https://www.sans.org/reading-room/whitepapers/application/web-application-firewalls-35817 [Accessed 17th July 2015].
6. PCI Security Standards Council, "Information Supplement: Requirement 6.6 Code Reviews and Application Firewalls Clarified," [online]. Available: https://www.pcisecuritystandards.org/pdfs/infosupp_6_6_applicationfirewalls_codereviews.pdf [Accessed 10th July 2016].
7. R.C. Barnett and J. Grossman, "'Web Application Defender's Cookbook: Battling Hackers and Protecting Users," ed. 1, Indianapolis, IN: John Wiley & Sons; 2013.
8. F. Yu, Z. Chen, Y. Diao, T. Lakshman, R.H. Katz, "'Fast and memory-efficient regular expression matching for deep packet inspection," *In: Proceedings of the 2006 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, San Jose, California, USA, 2006, pp. 93-102.

9. D. Pałka, M. Zachara, "'Learning Web Application Firewall - Benefits and Caveats," *In: Proceedings of the IFIP WG 8.4/8.9 International Cross Domain Conference and Workshop on Availability, Reliability and Security for Business, Enterprise and Health Information Systems,* Vienna, Austria, 2011, pp. 295-308.
10. S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann, M. Ameling, "'A service infrastructure for the Internet of Things based on XMPP," *In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom),* San Diego, CA, USA, 2013, pp. 385-388.
11. V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud Computing,* vol. 3, no. 1, pp. 11-17, 2015.
12. M. Kirsche, R. Klauck, "'Unify to bridge gaps: Bringing XMPP into the internet of things," *In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom),* Lugano, Switzerland, 2012, pp. 455-458.
13. R. Klauck, M. Kirsche, "'Chatty things-Making the Internet of Things readily usable for the masses with XMPP," *In: Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom),* Pittsburgh, PA, USA, 2012, pp. 60-69.
14. XMPP, "XMPP | Specifications," [online]. Available: https://xmpp.org/extensions/ [Accessed 28.7.2016].
15. P. Waher, "XEP 0323: Internet of Things - Sensor Data," [online]. Available: https://xmpp.org/extensions/xep-0323.html [Accessed 30.7.2016].
16. P. Saint-Andre, K. Smith and R. Tronçon, "'XMPP: the definitive guide," ed. 1, Sebastopol, CA: O'Reilly Media, Inc.; 2009.
17. I. Paterson, D. Smith, P. Saint-Andre, J. Moffitt, L. Stout and W. Tilanus, "XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH)," [online]. Available: https://xmpp.org/extensions/xep-0124.html [Accessed 8.11.2016].
18. MetaLib®, "MetaLib® - Home," [online]. Available: http://www.nelliportaali.fi/ [Accessed 19.8.2016].
19. I. Ristic, "'ModSecurity Handbook," United Kingdom: Feisty Duck; 2010.

20. P. Wood, "'2016 Internet Security Threat Report," [online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf [Accessed 22.8.2016].
21. Akamai's State of the Internet Security Team, "'Q1 2016 State of the Internet – Security Report," [online]. Available: https://www.akamai.com/es/es/multimedia/documents/state-of-the-internet/akamai-q1-2016-state-of-the-internet-security-report.pdf [Accessed 22.8.2016].
22. Akamai Technologies, "DDoS and Web Applications Attack Stats & Trends," [online]. Available: https://www.akamai.com/uk/en/multimedia/documents/state-of-the-internet/akamai-q1-2016-state-of-the-internet-security-report-infographic.pdf [Accessed 22.8.2016].
23. Z. Ghanbari, Y. Rahmani, H. Ghaffarian, M.H. Ahmadzadegan, "'Comparative approach to web application firewalls," *In: 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2015, pp. 808-812.
24. F5, "Security 101: BIG-IP ASM and IPS Differences Defined," [online]. Available: https://f5.com/resources/white-papers/security-101-big-ip-asm-and-ips-differences-defined [Accessed 25.8.2016].
25. T. Oosawa, T. Matsuda, "'SQL injection attack detection method using the approximation function of zeta distribution," *In: 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014, pp. 819-824.
26. A. Bremler-Barr and Y. Koral, "Accelerating Multipattern Matching on Compressed HTTP Traffic," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 970-983.
27. M. Masood, Z. Anwar, S. A. Raza, M. A. Hur, "'EDoS Armor: A cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments," *In: Multi Topic Conference (INMIC), 2013 16th International*, 2013, pp. 37-42.
28. S. Prandl, M. Lazarescu, D. Pham, "'A Study of Web Application Firewall Solutions," *In: Proceedings of 11th International Conference on Information Systems Security (ICISS 2015)*, Kolkata, India, 2015, pp. 501-510.

29. Trustwave, "ModSecurity: Open Source Web Application Firewall," [online]. Available: https://modsecurity.org [Accessed 19.8.2016].
30. AQTRONIX, "AQTRONIX WebKnight - Open Source Web Application Firewall (WAF) for IIS," [online]. Available: https://www.aqtronix.com/?PageID=99 [Accessed 7.11.2016].
31. J U M P E R Z . N E T, "Home," [online]. Available: http://www.jumperz.net/index.php [Accessed 7.11.2016].
32. H. Holm and M. Ekstedt, "Estimates on the effectiveness of web application firewalls against targeted attacks," *Info Mngmnt & Comp Security,* vol. 21, no. 4, pp. 250-265, 10/07; 2016/11.
33. M. Sharifi, M. Zoroufi, A. Saberi, "'How to Counter Control Flow Tampering Attacks," *In: 2007 IEEE/ACS International Conference on Computer Systems and Applications,* 2007, pp. 815-818.
34. F. Fangmei, C. Shao, D. Liu, "'Design and Implementation of Coldfusion-Based Web Application Firewall," *In: Computer Science & Service System (CSSS), 2012 International Conference on,* 2012, pp. 659-662.
35. A. Tekerek, C. Gemci, O. F. Bay, "'Development of a hybrid web application firewall to prevent web based attacks," *In: Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on,* 2014, pp. 1-4.
36. MaxMind, "GeoLite Legacy Downloadable Databases," [online]. Available: http://dev.maxmind.com/geoip/legacy/geolite/ [Accessed 28.2.2017].
37. Trustwave, "Reference Manual · SpiderLabs/ModSecurity Wiki · GitHub," [online]. Available: https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual [Accessed 19th April 2017].

APPENDIX A

Link to the web application firewall plugin that was implemented in this thesis:
https://github.com/WAF-XMPP/modsecurity-xmpp-rules