# Web Application Firewall (WAF)

Akhilesh Gadde, Datt Goswami, Sai Santhosh Vaidyam Anandan, Vamsi Krishna Atluri

April 26, 2015

**Abstract**

A web application is open to a host of attacks, which might compromise the confidentiality, availability and integrity of the server. The prevention of these attacks is mostly dependent on the implementation by the user and the web browser. The aim of this project is to come up with a plug and play kind of module which will help the user provide basic protection against known attacks and indifferent traffic patterns without any change to the existing code base. This is done by building two rulesets, first is a global ruleset to identify common attacks like SQL injection and the second is built based on the prior usage patterns of the web pages to identify the illegitimate traffic. The user needs to first train the learning module for his website and then load the developed module as part of their web server in order to enable the security features provided by the firewall.

## 1 Introduction

Firewall [3] is a network security system that controls the incoming and outgoing network traffic based on an applied rule set. A firewall establishes a barrier between a trusted, secure internal network and external network that is assumed to be not secure or trustworthy. When Firewalls are configured properly, authorized users are not even aware of its presence, but unauthorized users are blocked at the perimeter[4]. This adds a layer of protection against general Internet risks Firewalls exist both as software to run on general purpose hardware and also as a hardware appliance. In some instances, they are also implemented as a combination of both hardware and software. Firewalls can be implemented at different layers of the TCP/IP protocol stack. For example [4], a network layer firewall like Cisco PIX offers stateful packet inspection and connection oriented protection. There are other kind of stateless firewalls which only look at individual data packets and do not keep track of connections already established, therefore every connection made is considered to be a new connection. However, a stateful firewall keeps track of more than just the information contained within a data packet. It attempts to keep track of the network connections at the application level in real time. As the firewall reviews the data packets flowing to an application, it classifies its properties and analyzes how it fits into the overall communication flow. As it identifies these properties, it stores the data in a database, which it later uses to compare new data traffic for maliciousness[4].

The top most layer of the TCP/IP model is the Application layer. The application layer [4,5] is responsible for managing, setting up, coordinating and terminating conversations between applications at each end of the communication. HTTP is an example of the web application that runs at the application layer. In a web client/server environment, the requests from the client are received by the HTTP application on the server side which sends a response to the client. The firewalling can be implemented at the application layer as an application proxy firewall or as a module in the end-host. An application proxy firewall acts as an intermediary between a web client and a web server translating the address, additional access control checking and logging as necessary. It then establishes a separate session to the actual web server to send the client requests. The module based firewall is present directly in the server and can monitor any application input, output and system service calls to determine whether a process should respond to any given connection.

We design a module based firewall on the web server that is implemented in two phases. In the training phase, the firewall sits passively between the user and web pages, quietly observing and recording the kind of requests the web server handle for different pages. In the protection phase, the firewall is now active and reads rulesets from two sources. The first is the global rule set where all the attacks with known signatures are stored. The second is a training ruleset obtained from the training phase of the firewall. Web Applications are vulnerable to number of known signature malicious attacks viz. Cross-site scripting, Bot, SQL Injection, Directory/Path Traversal.

Some of the types of known signature attacks that the firewall protects against are [9,10,11,12,13,14]:
**Cross-site scripting Attacks:** Cross-site scripting attacks enables attackers to inject client-side script into Web pages viewed by other users.
**Bot Attacks:** If the user agent in the header contains bot, which can cause Distributed Denial of Service attack to make the web application unavailable to the users.
**SQL Injection:** It is the insertion of SQL query to get data, or to perform some malicious activity via the input data from the client to the application.
**Directory Traversal:** If an attacker is accessing restricted directories and execute commands outside web servers root directory.

Any request from the client to the web application has some parameters which would be passed in the HTTP GET or a POST requests. Based on the values for different parameters in the learning phase, a ruleset will be generated. This is used for identifying future malicious requests and blocking them. A few of the examples in the training ruleset are: maximum number of parameters seen for all requests across all pages, maximum number of parameters for every specific page, average length of specific parameter of every page, character set of the parameters, etc.

## 2 Architecture

The Web Application Firewall (WAF) is loaded as an additional component in the server hosting the web applications. In addition to the check for known malicious signatures, it consists of training module and protection module for anamoly detection. The training module just observes and records the legitimate traffic in a file in the server. The recorded values are passed to a Python parser which creates meaningful rules from the recorded data like the maximum number of parameters for any page, range of values observed for any parameters in the requests, etc. The protection module would read these rules from the file and decide whether a new request from the client needs to be responded by the server or blocked.

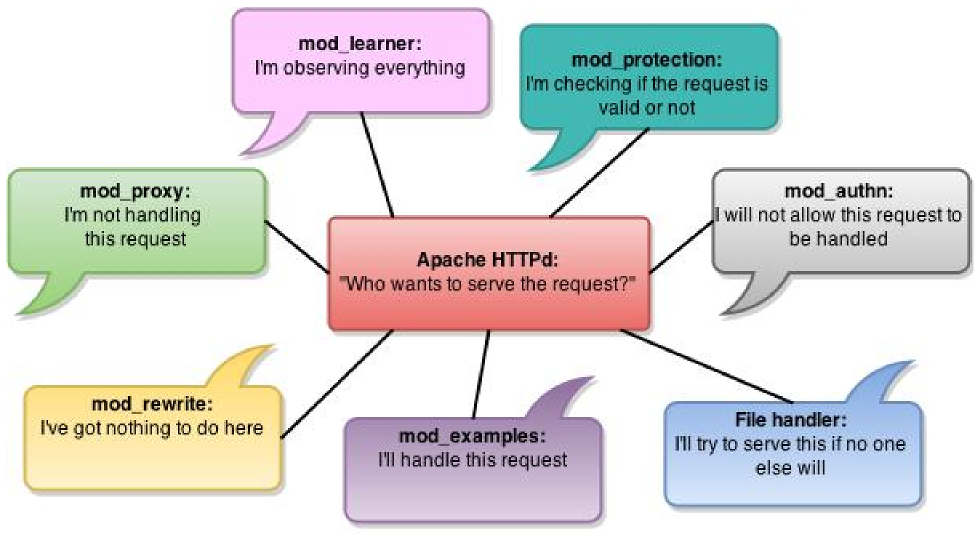
mod_learner: I'm observing everything
mod_protection: I'm checking if the request is valid or not
mod_proxy: I'm not handling this request
Apache HTTPd: "Who wants to serve the request?"
mod_authn: I will not allow this request to be handled
mod_rewrite: I've got nothing to do here
mod_examples: I'll handle this request
File handler: I'll try to serve this if no one else will

**Fig1.** Modules of HTTP web server.

The training and protection modules are coded in C language and are compiled and loaded as new modules in the Apache Server. The server is restarted with the new modules and verified for functionality. In $Fig1$, the $mod_learner$ module does the training for the firewall during the learning phase. After learning, this data is passed to a parser that generated meaningful rulsets, called training ruleset. The $mod_protection$ module does the actual protection of the web server from the malicious traffic or known signature attacks. The protection phase verifies the request from the client for each general signature attacks followed by the rules obtained from the training phase for each parameter. At any stage of the verification process, if the firewall determines that the request is not genuine, it is dropped.

## 3 Methodology

The WAF protects the server from known malicious signatures and from illegitimate traffic from the clients. The firewall functionality can be classified into two broad categories: Signature Detection and Anomaly Detection.

### 3.1 Signature Detection

The global ruleset file consists of the signatures for the possible server side attacks. The file is a simple text file where rules can be added dynamically on the fly. The rules have a specific format which the user needs to follow while adding new signatures. For example, the signatures like ¡script¿ string in User-Agent parameter helps the firewall in detecting the basic XSS attack through the header from the client requests. These malicious requests are dropped at the firewall and the server is protected from these attacks. If the firewall determines that the client request is not malicious, the request is passed to the next part of the code that checks it against the trained ruleset.

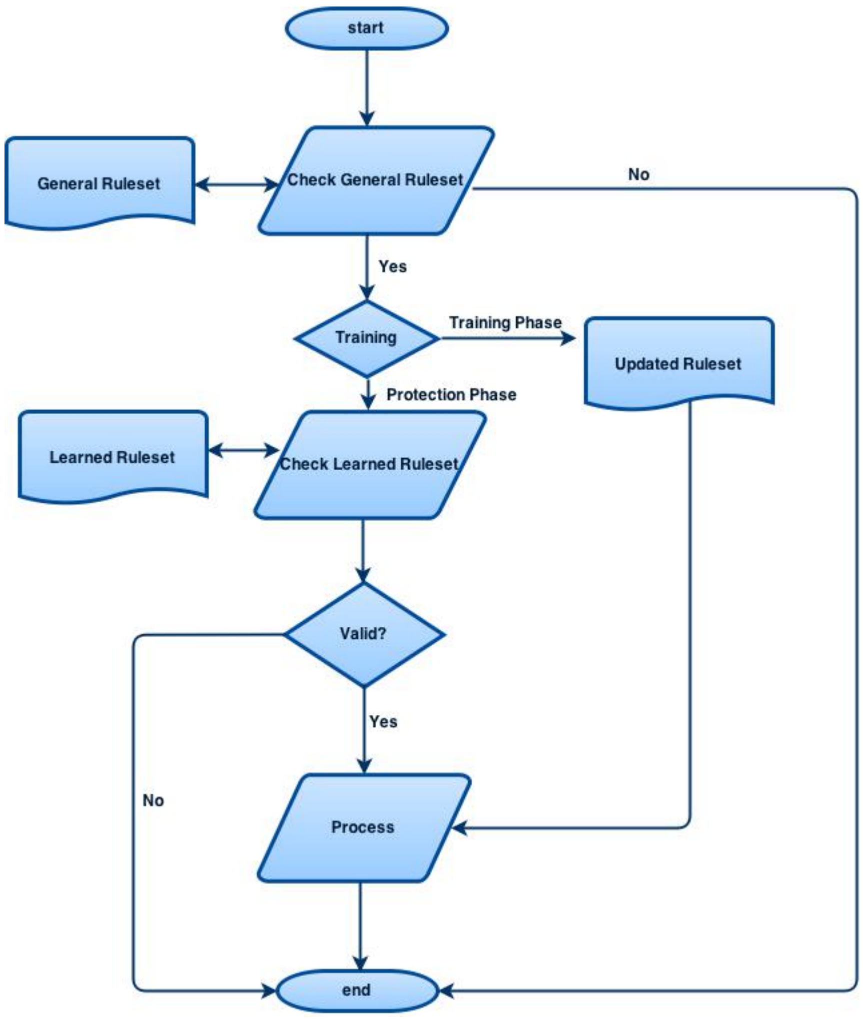

**Fig2.** Flowchart of WAF

## 3.2 Anomaly Detection

The global ruleset helps us in identifying the known malicious attacks but cannot stop attacks that do not have corresponding known signatures. We learn how the legitimate traffic looks like during the training phase and if we observe that the future traffic is too different from what we have learned, we do not allow the requests to proceed for further processing. The functionality is split into 2 phases - Training and Protection.

### 3.2.1 Training Phase

During the training phase, the WAF just collects the information about all the information passing through it. The user tries to use the website as much as possible, click on all possible links, and perform all possible actions. When the user is satisfied that he has exercised the web application as much as possible, it asks the WAF to create a normality profile. A python parser is called which reads the parameters from the recorded client requests and creates the normality profile for them.

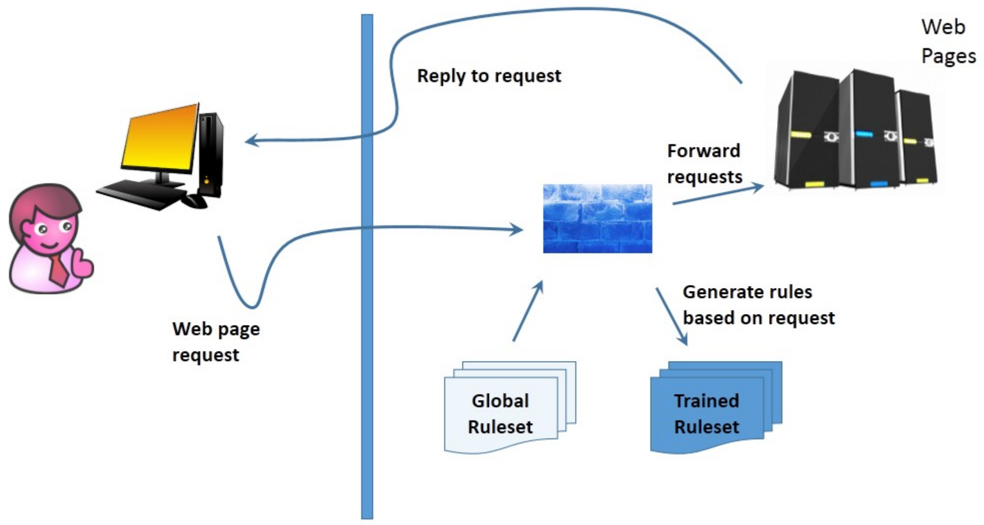

**Fig3.** Anomaly Detection: Training Phase

The normality profile consists of rules like the maximum number of parameters for any specific page of the website, character set of any specific parameter for a page, the minimum and maximum possible values of a parameter based on the standard deviation of the values observed in the learning phase,etc. When the values are normalized, three times the standard deviation from the mean on both the sides of the mean would encompass 99% of the genuine requests. The rules obtained from this phase are passed to the protection phase where incoming requests are validated.

### 3.2.2 Protection Phase

The rules generated from the learning phase are provided as input against which the incoming requests are validated by the firewall in protection phase. The firewall is in live phase now where it is no more passively observing the requests but instead actively checking them and deciding whether the requests are genuine. The firewall would first check for anomalies using the known signatures and then read the trained rule set. The header parameters for the incoming requests are checked and the rules from the rule set file for the corresponding parameter value are fetched and checked against the incoming values for legitimacy of the incoming requests. If the values for any parameter are within the the specified range of the rule set, the request is sent for further processing to the server. Else, the request is dropped and a log entry is made in the file specifying the details on why the request was dropped. This can be used for later analysis to find the cause of denial of response from the server.

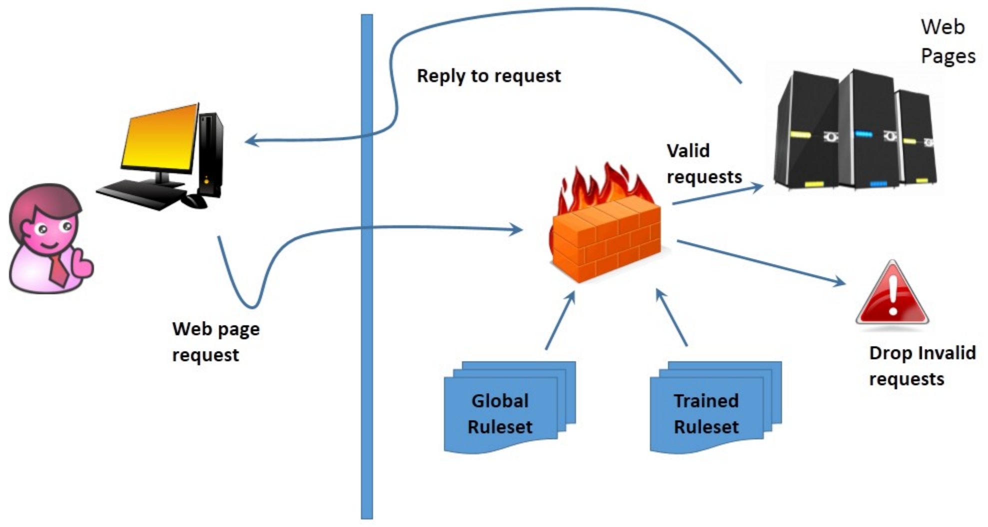

**Fig4.** Anomaly Detection: Protection Phase

In summary, the detection should happen as follows:

- For profiles involving maximum number of parameters, drop requests that exceed that maximum number.
- For profiles involving averages, assume a normal distribution, compute the standard deviation, and drop all values that are not part of the mean+-3*sd.
- For profiles involving character sets, drop requests with parameters containing character from sets not seen during training phase.

## 4 Implementation

The implementation of the project involves the following areas -

### 4.1 Web Pages (Apache, PHP, HTML/CSS)

Simple web pages are built using HTML/CSS/PHP. The web site contains a login page, registration page and the main home page. They are hosted on Apache Tomcat server locally and accessible at http://localhost/$< pagename >$.html

### 4.2 Learning Module (C, APXS)

Custom HTTP modules for training phase is written in C language and converted to the Apache .so module using the APXS utility. Once the module is loaded to the server and restarted, it registers itself with the web server saying it now wants to process all request first hand before being processed by the HTTP response module. This Apache module, once invoked, performs the following operation:

- Read the general rule set and verify if the request is valid based on the known signatures in the global ruleset.
- The various GET and POST parameters are identified and written out to the file along with the destination URI.
- Returns the request to the web server for further processing.

The different parameters recorded in this phase are written to the file 'my_log.txt'.

### 4.3 Parsing Captured Parameters (Python)

After the learning phase is done, the parameters in 'my_log.txt' needs to be parsed in order to generate the statistics and meaningful trainning ruleset that will be used in the protection phase. This is done using the python file 'learner.py'. The python code takes the 'my_log.txt' as input and stores the statistics in various files whose location is based on the URI heirarchy.

### 4.4 Protection Module (C, APXS)

Similar to the learning module, this is another custom module called protection, written in C and the corresponding .so module is registered with the Apache web server. The module generates two linked lists of rules - one for the general ruleset and the other for the learned rules from the training ruleset. Every incoming request parameter is verified with the rules from these linked list and does one of the following

- If the request is valid, send it to the server for further processing.
- If the request is invalid, decline and log the reason for the action.

## 5 Conclusions

The global ruleset helps the firewall in detecting the known signatures for malicious attacks from the clients. But this would only protect us against the known signatures and hence we have used anomaly detection using the training and protection phases where the learning phase helps the firewall in identifying the legitimate requests by storing the different parameters, their values, the possible character set, etc and the protection phase would enable the firewall to detect malicious traffic using the training ruleset from the learning phase. A python parser is used by the WAF to create a normality profile for different pages of the website and create

training ruleset from the recorded data. The global ruleset and training ruleset are implemented as files and can be extended as needed. New rules can be added in the correct format to both the files and the firewall doesnt need to be restarted. Any new requests coming in from the client would be checked against all the rules specified in the file.
This provides a simple plug and play mechanism for the firewall that can be implemented on any web server and the extensibility is made possible due to the dynamic addition of rules in the ruleset files. This would help the user in protecting the server from any known signature attacks as well as unknown signature attacks like Denial of Service (DDoS), invalid parameters in the requests,etc. This would also help the user to just plug this module to an existing server and not bother about the protection mechanisms at client browser or at the server implementation code.

## References

[1] Developing modules for the Apache HTTP Server - http://httpd.apache.org/docs/2.4/developer/modguide.html

[2] Project Description - http://securitee.org/teaching/cse508/projects/project2.html

[3] Firewall wikipedia, http://en.wikipedia.org/wiki/Firewall

[4] TCP/IP Protocol suite, http://www.thegeekstuff.com/2011/11/tcp-ip-fundamentals/

[5] An Overview of Defense in Depth at each layer of the TCP/IP Model, http://www.giac.org/paper/gsec/2233/overview-defense-in-depth-layer-tcp-ip-model/103817

[6] Host vs Network based firewalls, http://science.opposingviews.com/hostbased-vs-networkbased-firewalls-2000.html

[7] An introduction to type of firewalls, http://searchnetworking.techtarget.com/tutorial/Introduction-to-firewalls-Types-of-firewalls

[8] Advantages of proxy based firewalls, http://searchsecurity.techtarget.com/answer/The-benefits-of-application-proxy-firewalls

[9] XSS attacks, https://www.acunetix.com/websitesecurity/xss/

[10] Botnet DDoS attacks, https://www.incapsula.com/ddos/ddos-attacks/botnet-ddos.html

[11] SQL injection attacks, https://technet.microsoft.com/en-us/library/ms161953(v=sql.105).aspx

[12] SQL injection prevention cheat sheet, http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet

[13] Directory Traversal attacks, http://projects.webappsec.org/w/page/13246952/Path_Traversal

[14] Path traversal attacks, http://www.acunetix.com/websitesecurity/directory-traversal/