Cross Site Scripting (XSS) Web Attack Detection by Machine Learning

KAMRAN ALI KHAN
01-134132-076
RANA M. USMAN NASIR
01-134132-158

Bachelor of Science in Computer Science

Supervisor: Mr. Umar Khattak

Department of Computer Science
Bahria University, Islamabad

May 2017

© Kamran Ali Khan & Rana M. Usman Nasir, 2017

Certificate

It is to certify that the final year project of BS (CS) "Cross Site Scripting (XSS) Web Attack Detection by Machine Learning", was developed by Kamran Ali Khan, Rana Usman Nasir under the supervision of "Mr. Umar Khattak" and in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Science.

Approved by . . . :

Supervisor: Mr. Umar Khattak (Lecturer)

____

Internal Examiner:

____

External Examiner:

____

Project Coordinator: Dr. Arif Ur Rahman (Assistant Professor)

____

Head of the Department: Dr. Faisal Bashir (Associate Professor)

____

May 19^{th}, 2017

Abstract

In today's world, Web application security is a major issue for web application developers. There are several reasons behind that but the major cause of this problem is that most of the web developers don't have the adequate knowledge regarding secure coding. So, taking advantage of that carelessness of the developers, web applications have vulnerabilities and attackers take advantage of that.
In this project, we build a firewall which detects the XSS (Cross Site Scripting) vulnerabilities through machine learning and tries to block web attacks preventing web applications from serious troubles. The system makes decisions through a dataset which is provided to the system, we then use machine learning algorithm to classify each web request, whether it is a harmful request or not. There is no need of any headache procedures for the web application's security department as it is automatically configured and there is no need to define attack signatures. Web application security is very expensive field nowadays, companies use most of their budget in this field. Our system is free for the web application developers and users. Our system detects XSS automatically by classification of input using machine learning techniques. Our proposed features show the results which are a highly accurate classification of malicious inputs.

Acknowledgments

All praise is to Almighty Allah who bestowed upon us a minute portion of his boundless knowledge by virtue of which we are able to accomplish this challenging task. We are greatly indebted to our supervisor “Mr. Umar Khattak”. Without his personal supervision, advice and valuable guidance, completion of this thesis would have been doubtful. We are deeply indebted to him for his encouragement and continual help during this work.

KAMRAN ALI KHAN

RANA M. USMAN NASIR
Islamabad, Pakistan

May 2017

"We think someone else, someone smarter than us, someone more capable, someone with more resources will solve that problem. But there isn't anyone else."

Regina Dugan

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

XSS	Cross-Site Scripting
IDS	Intrusion Detection System
IDE	Integrated Development Environment
HTTPS	Hypertext Transfer Protocol Secure
DOS	Denial f Service
URL	Uniform Resource Locator
SQL	Structured Query Language
HTML	Hypertext Markup Language

Chapter 1

Introduction

Web Application security is very important for the web application's owner because nowadays most of the businesses are online. It is a reality that in two decades the web grown up so fast and build a platform to access and run complex web applications. These applications might be used in daily life, or in large scale for commercial services such as (gmail, outlook, twitter etc.). So, web applications are a common source of income and people depend on them that's why the security of the web application is a major issue. There are several reasons for the insecurity of the web applications. Our project basically deals with these issues of the web application's security. As web applications are a very common form of public service and they are easily accessible through HTTP protocol, that is the reason for being easy targets for hackers or penetration testers. Our application provides a solution that a common user can adopt to secure their web application from cross-site scripting attacks without having the extensive knowledge of application security [4].

1.1 Project Background

Current web application firewalls are mostly signature based firewalls. They use regular expression matching to identify malicious traffic. To use them we must define signatures of known attacks so, that firewall can match against those rules and block them. This way of operation is slow and produce high overheads, as there can be millions of attack signatures. Most of the firewall are manually configured that takes time and it is expensive too [1].

1.2 Problem Description

A web application that is built on PHP or ASP as a back-end programming language (which communicates with the database and underlying operating system) take inputs from the user. These inputs are being stored in a database or executed at the underlying operating system to perform various tasks. Web application developers mostly do not sanitize these inputs (because of time limitations, unconsciously or they don't know) and all the inputs (Legal and Malicious) gets processed by the application. We need to define another layer where all these inputs are first being filtered before they reach the application layer. Now all the input data will first pass through this layer if there is any invalid or undesirable data it must not pass through this layer (Firewall Layer). But the problem with available firewalls is they need to be configured and their rules must be manually defined by a developer (i.e. which input is legal and illegal). It's not an easy job because we cannot predict all kind of inputs a user can send, so our application will learn from log files generated by the server and then machine learning model will classify the future requests into two possible categories. So, that a web developer can only focus on the development part of the application.

1.3 Project Objectives

To design and develop an Application in which we implement a security system that provides security through machine learning while providing smooth access to legitimate users.

1.4 Project Scope

Basically, our objective in this project is to introduce the new way of securing the web applications. Our goals are to build a secure and intelligent Firewall for the Web applications that does not require any user input to work. All data inputs on the web application level must be validated on the firewall. Client side and server side validation is encouraged but not meant to be the only control for data input validation. The most sustainable strategy for data validation is accepting known good data (Legal Data) and so, our idea is to remove malicious input/strings through this layer. The other solutions available require constant maintenance and human interaction to keep the system secure and up to date.

Chapter 2

Literature Review

In this chapter, we will identify, evaluate and interpret any work or research that has been done related to our project. It will provide help and a layout for achieving our goals. As this project involves a lot of research work so, this chapter holds a lot of significance. This chapter identify the different features extraction techniques and algorithms that can help with our project.

2.1 Related Work

Author, proposed an approach which is based on Black box concept. In anomaly-based IDS (Intrusion Detection System) several approaches are used but author used Black Box approach. Through black box approach they send HTTP request and analyse the response, without the internal knowledge of how application is developed. IDS detect web attacks without the internal information of the application. Author also mentioned that they defined different features through which they identify whether the input data is malicious or not. First, they detect source and destination addresses by identifying whether its DOS attack or not, secondly which type of protocol is used in the request, with the length of URL (Uniform Resource Locator) they tried to identify whether its a SQL (Structured Query Language) or XSS attack, the time and date of the transfer data, number of logins and the request methods, HTTP request contain any SQL query, another feature they check is there any script tags, hexadecimal code, Request status from browser, the response body and its length. Through these features web application vulnerabilities can be checked, by securing these features web applications can be secured from attacks. The author concluded that with these features and machine learning algorithm in classification they achieved good results specially in XSS (Cross Site Scripting) [8].
Author introduced the new technique known as taint inference. In this technique, no need of knowing the source code and this method is efficient for low performance servers. Most of the Applications for this interference maybe accomplished through network layer

interposition. Then we defined some polices that can check whether the input is malicious or not. The results after checking through this technique are good and most attacks are easily blocked or filtered through it. In traint inference algorithm at a time single pair of data flow from one to another. Special names (N, I) has been given to each of them and the output of these also with another name. Then the taint inference is checked weather any substring has data from several algorithms might be used depend upon the data. In this algorithm, Author rely on signal type of data and retrieve the data from the cookies and use it further. A defined threshold used in it if data (N, I) is less than defined threshold then other several other algorithms use and check the data. Then there is an output syntax analysers that pass the data from the polices, which are defined. The authors concluded that with these features of taint inferring passively observed the input and outputs of the Web applications through this multiple language can be checked through normal manners [3].
In another study Authors proposed that most of the vulnerabilities occurs due to mistake of programmers this paper is based on solving the problems of the source code bugs. This hybrid method is used to detect bugs with less false positive. After the initial steps, they used taint analysis to highlight the applicant's vulnerabilities with the approach of Data mining. Then use a classifier (machine learning) to guess whether each applicant vulnerability is a false positive or not. Then correct it through the data we identify the false positive after the comparison of several replacements. it's very important to understand that the given techniques might not give 100 % correct results. These are the static analysis so, it must have issues. Data mining cannot avoid this undecidability that why it's only give probabilistic results. For classifications purpose the classifier use ten matric on the base of the four parameters of each classifier. Through that data is checked. Classify the vulnerabilities, then fixed it after correction insert it where the place they must inserted. Mostly vulnerabilities are false positive so, source code is modifying through removing them [5].
Author carries out the experiment for XSS detections security through two machine learning methods, Naive Bayes and Support Vector. In naive Bayes method data, must classified in different categories. It's an Arithmetical method which is based on Bayes rule. On this classification, decision is made by calculating the probability and the costs. This classifier adopts the feature conditional independence by assuming this feature value is totally different from the class of other features. For sample "X" the Bayes classifier calculates the posteriori probability of every given class "X". Through this they deal with the set of labels and by storing g this set of label data in database which included positive and negative samples whether the code is safe or malicious [6].
Author explains cross site scripting security vulnerability and its impact. Basically, attacks are classified into two major categories the persistent and non-persistence attacks. The persistent attacks are riskier as compared to the non-persistence attacks. With the passing of time and due to the advancement of technologies the XSS attacks has grown. So, Author

provide an approach to solve this issue that is an API known as Enterprise Security API. Author mentioned that though this approach they have hand cross site scripting vulnerabilities happen when the attacker injects the malicious code whether it's in java script or in special character that would affect the internal of the web site. He also mentions some other tools to attack at web applications, might be they used previous session-id to inject the malicious code, with the help of these tools they can attacks the web application often bypassing HTTP. So, for the protection we must filter the input that can be achieved through applications filters. So, by validating the input data by verifying http request object for malicious content. Through ESAPI (The OWASP Enterprise Security API) based approach validates the input data but it's takes time but less than normal validates times.by concluding Author said that ESAPI helps to secure the web applications and handle web application security risks using this standard technique [2].
Author explained how the XSS vulnerabilities occurs, he explained how to tackle them. Through feature extraction which plays an important role to check the malicious code. Script content are used to delivering and hiding the malicious code by obfuscations (e.g. string size, word size, arguments size etc.). There are core contents through these content web application is target in web browser, plug in and operating system. Author also mentioned DOM object, this type of object is target on web application vulnerabilities (like document, URL, document. URL Un encoded, document. location, document. Referrer etc.). So, scan the XSS and identified based on the features. so, using this we can trace the contents and checks whether it's a normal http packet or there is any malicious code into. Through machine learning it will work more effective. Save these features and through machine learning by using Naive Bayes algorithm classified the data set which is statistical method based on it calculating the probability and costs related to each decision. Author also Discuss Decision tree in which they organised the classification scheme. The tree begins with the root node which is considered as a "parent node" and then the other nodes. So, tree comparing each node value with the constant values.so these steps caring on until the leaf node. This method of classification is like the classification of data the machine learning algorithms by concluding Authors said that the experimental result shows the same results as compare to Naive Bayes classifier results [7].
In the below table, we sum up all the problem that are defined above and their proposed solutions. Authors mentioned different techniques to solve those issues and at the end we proposed our solution how we implement the solution in our project.

Table 2.1: Literature Review in Tabular Form

Author	Problem Description	Solution
• El Moussaid. • Nadya ElBachir. • Ahmed Toumanari.	Authors described different attacks and why they occur. Web application hacked by hackers through SQL injections, XSS etc.	In this research authors defines a solution they named it intrusion detection system. In anomaly- based IDS several approaches are used but author used Black Box approach
• Medeiros. • Nuno F. Neves. • Miguel Correia. • IbÃľria.	Many programmers do not have adequate knowledge about secure coding	Author defined an approach that use Source code static analysis to find the bugs in Code. This paper explores the use of hybrid methods to detect the vulnerabilities. • Taint analysis. • Machine learning. Code analysis (Taint analysis) Machine learning (Data mining)
• R. Sekar.	Over the past few years, injection vulnerabilities have become primary target for remote exploits.	Author discovered the new approach for defending that exploits. Taint-tracking is the name of that approach. Taint propagation by passively observing the inputs and outputs of the protected application.
• Angelo Eduardo Nunan. • Eduardo Souto.	Advanced features in web browser increase the user and increased security risks and attacks since they allow malicious codes injection.	This paper focus automatic classification of XSS attacks on web pages by extracting and predicting features of web document content on URL. Naive Bayes and SVM classifier techniques used in this paper.
• S.Krishnaveni. • K.Sathiyakumari.	When a security mechanism is failed then the user may download malicious code from a trusted web site. In this case, the malicious script is contracted to full access with all assets belonging to that legitimate web site. These types of attacks are called Cross-Site Scripting (XSS) attacks.	Finding the malicious web pages is a difficult task for the security team of the web application. In this paper classification of XSS attacks through some techniques like data mining, documentation scanning. From web pages features are extracted and then data is scanned using Naïve Bayes algorithm.
• Bhanu Prakash Gopularam. • N. Nalini.	Mostly the attacks are classified as persistent and non-persistent attacks.	In this paper, Author mentioned some libraries for maintaining the security of the web applications. The security library like OWASP, ESAPI would help web applications to handle security risks using standardized methods.

In the above table, we have explained all the work has been done on this field. Their ideas, proposed solution all are mentioned in the table.

Proposed Solution: As we have mentioned above in a tabular form, the problems and there solutions provided or proposed by other Authors and researchers. They all are fully experts in their field. But according to our proposed solution, those solutions are more expensive and they need more powerful system for their operations. Our Solution is free for all web application users, easy to understand, and user friendly. In our project, we classified the data through Naive Bayes machine learning algorithm. All incoming requests will be checked by our machine learning model and if there is any problem in the request firewall will block the request (Cross Site Scripting Attack) and maintain the web application security. Data Acquisition, Feature Extraction, Classification are the main modules in our project.

Chapter 3

Requirement Specifications

3.1 Proposed System

The system we have proposed would be a Linux based Application that will be used as a security module in NGINX. In our proposed system, the user enters the data in given fields, that could be a login ID, password or anything which is required by the particular application being run on the server. That entered data will be filtered through Firewall before it reaches the web application. If the traffic is marked not harmful it is then passed to the web application. This filtration mechanism is performed through machine learning. Classification will be performed through a labeled data-set which will be obtained from preprocessing the log file generated by the server. After that machine learning model will be trained using that data and web requests will be classified into two categories and they are spam or not spam. As the time passes more data will be available and we can use that data to frequently train our model to improve its performance.

3.2 Existing System

Current web application firewalls are mostly signature based firewalls. They use regular expression matching to identify malicious traffic. To use them we've to define signatures of known attacks so that firewall can match against those rules and block them. This way of operation is slow and produce high overheads, as there can be millions of attack signatures. Due to memory and other limitations we can also not define all possible combinations of attack signatures. On the other hand, they are also hard to install and configure, because a lot of manual configuration is required for setting attack signatures. However, our proposed system does not need any manual configuration, user does not need to define attack signatures because we use machine learning to classify the traffic.

3.3 Requirement Specifications

In requirement specification, we must specify functional and non-functional requirements of our system. Functional and non-functional requirements of our proposed system is given below.

3.3.1 Functional Requirements

Functional requirements specify what inputs are given to the system and what corresponding output is produced and how it behaves on input. Following are the functional requirements of our system:

- User installation script (makefile).
- User will have option to upload their own training data.
- Perform model evaluation.
- Script for manual training of model from user side.
- Configuration file to turn on or off the software as NGINX module.
- A trained model that will perform prediction of traffic.

3.3.2 Non-Functional Requirements

Non-functional requirements are the requirements that are mentioned to grade system performance. The system can work without them, but they can increase the performance of the overall software application. Below are our non-functional requirements.

1. **Availability:**
   Can be easily turned on or off from the configuration file.
2. **Efficiency:**
   Our focus for this module to be efficient, for that only the minimum required part is coded in python. Rest is coded in C.
3. **Usability:**
   This NGINX module is very easy to use since it requires no manual further configuration from its user.
4. **Robustness:**
   Our module will recover from NGINX crashes since servers are prone to crashes we make sure it come back to its normal operation after a restart.

5. **Reliability:**
   Sinceuser can also perform manual training of the model, but our default data set provides 80 % plus positive results.

3.4 Use Cases

A use case is a list of actions or events which define the interaction between the actors and system to achieve a goal. The main use case diagram of our system is shown in 3.1 The actors in this use-cases are the persons who will be using the system.

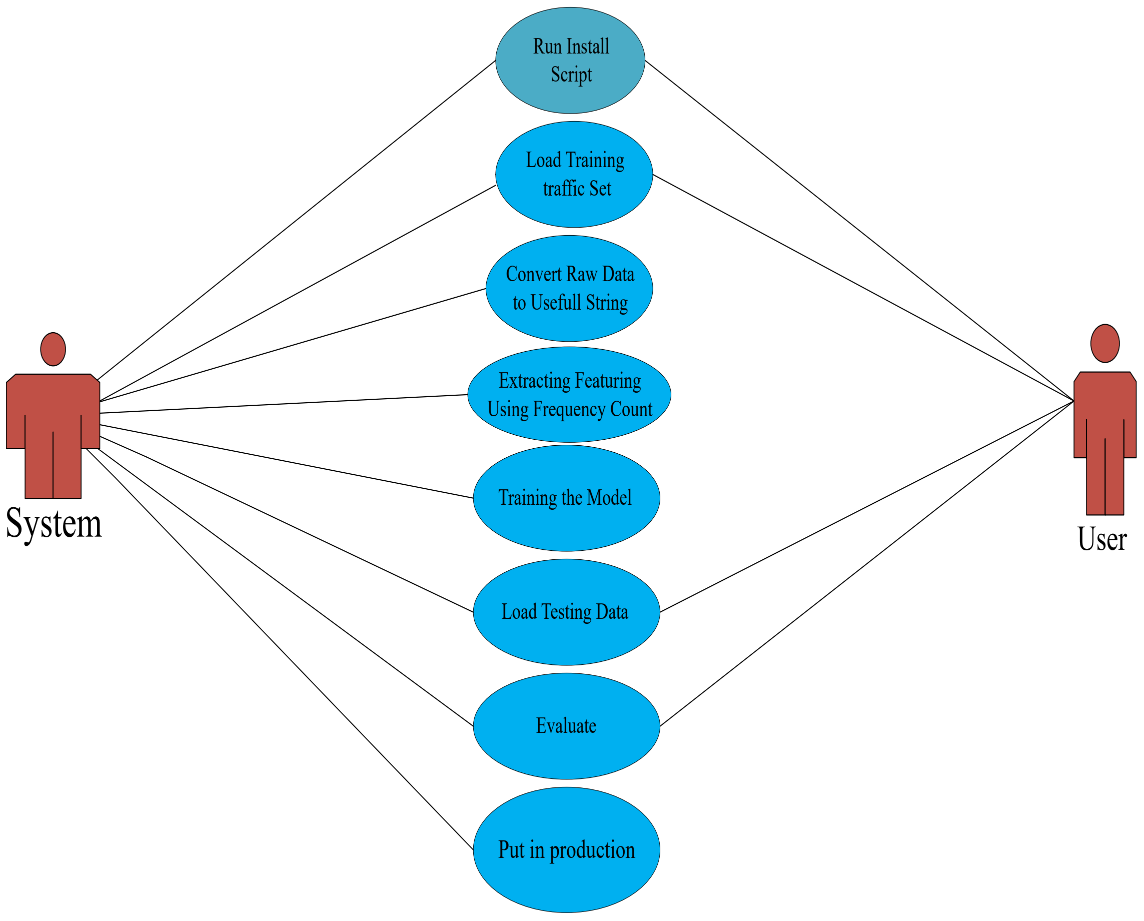

Run Install Script
Load Training traffic Set
Convert Raw Data to Usefull String
Extracting Featuring Using Frequency Count
Training the Model
Load Testing Data
Evaluate
Put in production
System
User

Figure 3.1: Main Use Case Diagram

The individual use cases are presented below, starting from figure 3.2 to figure 3.9 with their respective specifications table.

Table 3.1: Main Use Case

Use case ID	UC-1
Title	The Main Use Case Diagram of the System.
Description	Shows the whole working of the System.
Primary Actor	System and user.
Pre-Condition	System is Installed properly
Post-Condition	Entered Data is fully checked
Success Scenario	Entered Data is normal.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

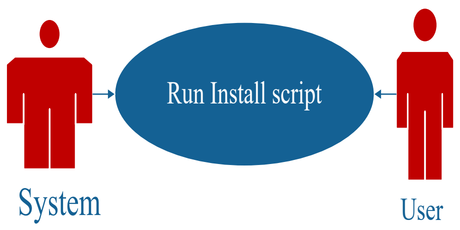


Figure 3.2: Install Script

In figure 3.2 shows the system gets the command or data from the user and install the script. Detail is in the Table below.

Table 3.2: Install Script

Use case ID	UC-2
Title	Run Install Script
Description	Get data from user and Run install Script in system.
Primary Actor	System and User.
Pre-Condition	Training data Should be Stored in system folder.
Post-Condition	System is installed as desired.
Success Scenario	Application Installation.
Exceptions	Data is not normal.
Assumptions	User don't know how to enter the data.

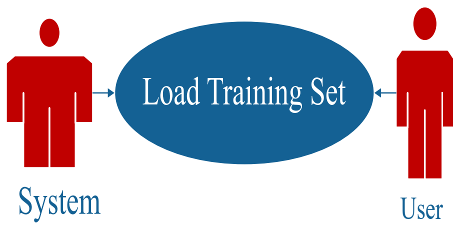


Figure 3.3: Load Training Traffic Set

After the installation, the use case figure 3.3 Load the training set to train the Model.

Table 3.3: Load Training Traffic Set

Use case ID	UC-3
Title	Load Traffic set
Description	After the install script training traffic is loaded.
Primary Actor	System and User.
Pre-Condition	Training data Should be Stored in system folder.
Post-Condition	Data set loaded for the feature extraction.
Success Scenar io	User's input is normal data.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

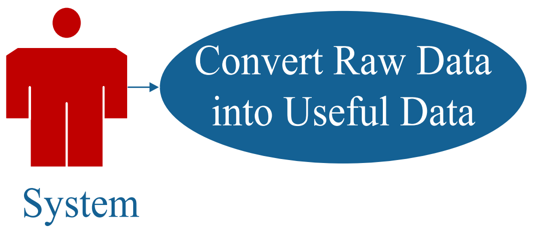


Figure 3.4: Convert Raw Data into Useful Data

After the Training Set, Conversion is performed in figure 3.4. System convert the Raw data into useful data. Detail in table below.

Table 3.4: Convert Raw Data into Useful Data

Use case ID	UC-4
Title	Convert Raw data into useful data.
Description	Raw input data is converted to useful data.
Primary Actor	System.
Pre-Condition	Training data is loaded into the application.
Post-Condition	Useful data is extracted.
Success Scenario	Useful data.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

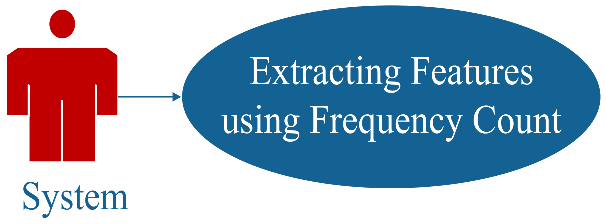


Figure 3.5: Extracting Featuring

It is an individual use case 3.5 of the system. Find the frequency of the data.

Table 3.5: Extracting Features Through Frequency Count

Use case ID	UC-5
Title	Extracting featuring.
Description	Extracting the features from useful data.
Primary Actor	System.
Pre-Condition	Availability of useful data.
Post-Condition	Features extracted.
Success Scenario	Features Set.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

Training the Model
System

Figure 3.6: Training the Model

After the frequency, we train the model in 3.6 to check the probability of the data.

Table 3.6: Model's Training

Use case ID	UC-6
Title	Training the Model.
Description	System is trained through extracted features.
Primary Actor	System.
Pre-Condition	Extracted useful features.
Post-Condition	Trained model.
Success Scenario	Trained model.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

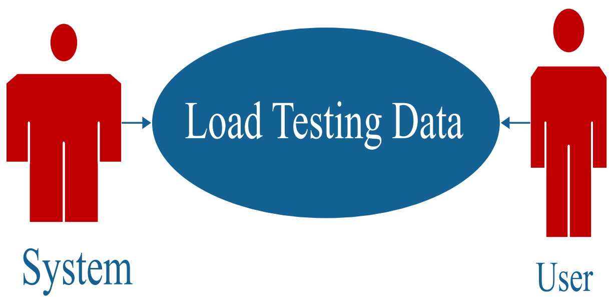


Figure 3.7: Load the Testing Data

After the data occurrences of data probability, we will test the data in 3.7. Detail is in table.

Table 3.7: Load the Testing Data

Use case ID	UC-7
Title	Load testing data.
Description	System Load the testing data.
Primary Actor	System and User.
Pre-Condition	Training data should be stored in system folder.
Post-Condition	Evaluation.
Success Scenario	Test data loaded and feature extracted.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

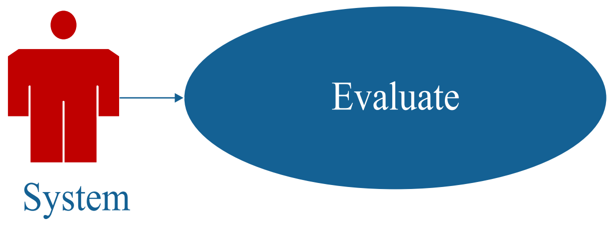


Figure 3.8: Evaluate

After that in 3.8 system will evaluate whether the input is malicious or normal.

Table 3.8: Evaluate

Use case ID	UC-8
Title	Evaluate.
Description	Evaluate
Primary Actor	System.
Pre-Condition	Training data set with features extracted.
Post-Condition	Evaluation result.
Success Scenario	Evaluation result is 90% or above.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

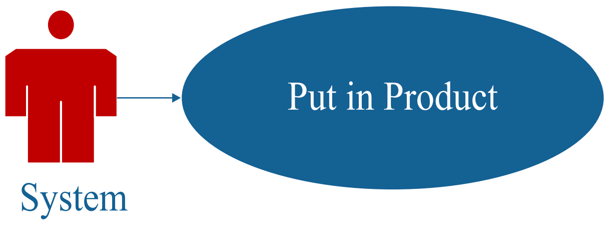


Figure 3.9: Product

The last use case of our system is Put in product 3.9

Table 3.9: Put in production

Use case ID	UC-9
Title	Put in Production.
Description	Model is than integrated with NGINX.
Primary Actor	System.
Pre-Condition	Model evaluates to 90% and above.
Post-Condition	Model is behaving as expected.
Success Scenario	Successful filtering of attacks.
Exceptions	Data is not normal.
Assumptions	User know how to enter the data.

Chapter 4

Design

This chapter is related to architecture and design. There are three phases of our project. There are three basic modules of our project are.

- Load Traffic.
- Labelled Traffic.
- Firewall.

4.1 System Architecture

System architecture is very important component for the representation of the application because it's a high level logical representation of any application and it's shows what are the components of the system and how they relate to each other. In figure 4.1 the System architecture of our application is shown which is actually a web application firewall.

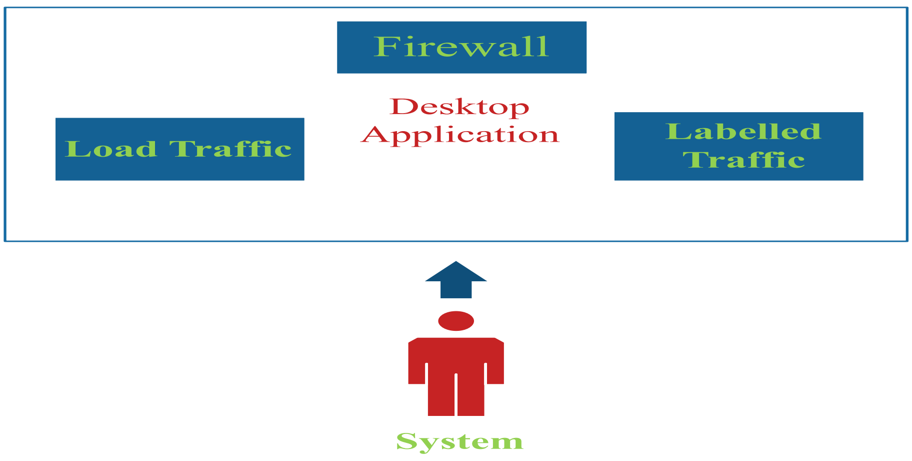


Figure 4.1: System Architecture

Our system architecture has four major components i.e. user's input, load traffic, firewall, and labelled traffic. Our system doesn't need any external database like SQL etc. Data set is stored in a log file for feature extraction.

4.1.1 Design Methodology

Our system's methodology is expressed in figure 4.2 where two separate independent modules are working. First the training data set of sample as a training web traffic will be loaded and their feature vectors will be stored into the system. Now they will be available for matching.

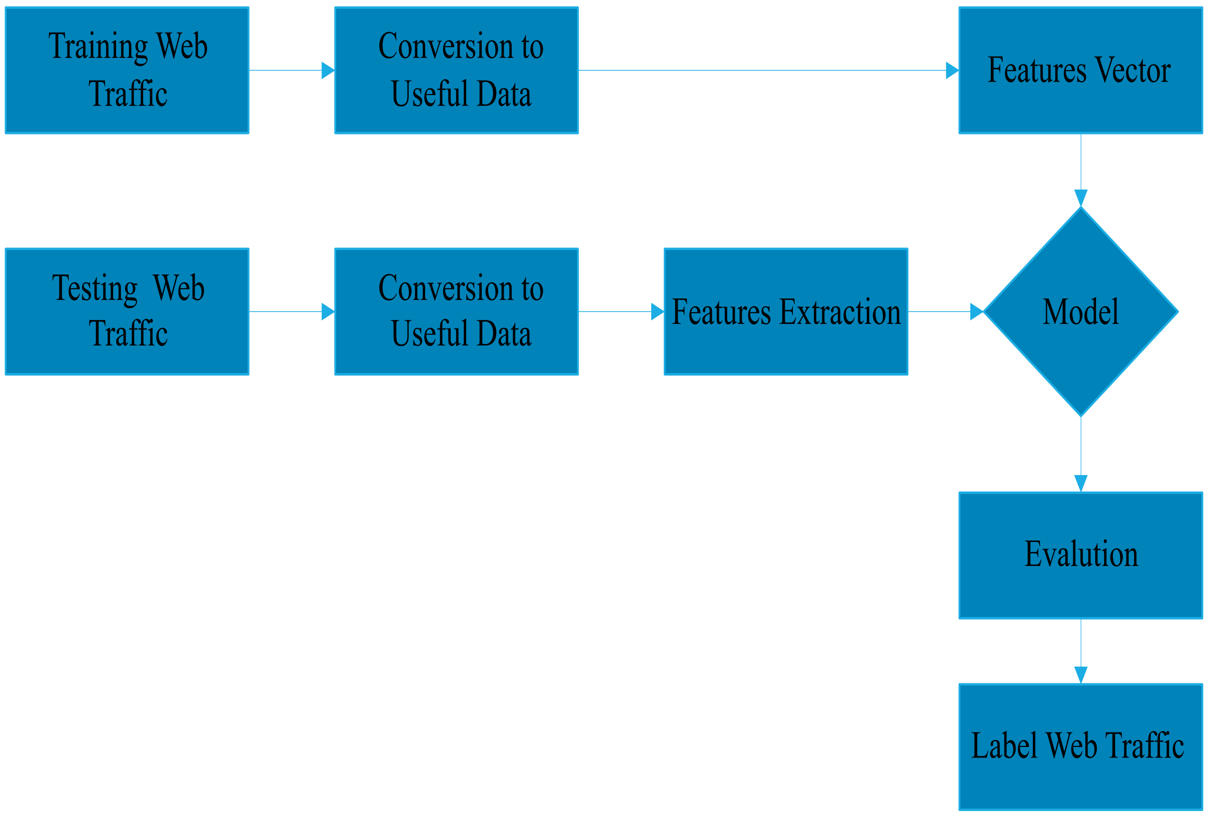


Figure 4.2: Diagram of the Methodology of the Application

4.2 High Level Design

This section describes in further detail elements discussed in the previous section. In Figure 4.4 shows the data flow diagram of the application. Browser send request to NGINX web server and from web server data pass through our firewall. Working is in Figure.

Following are the different views of high level design.

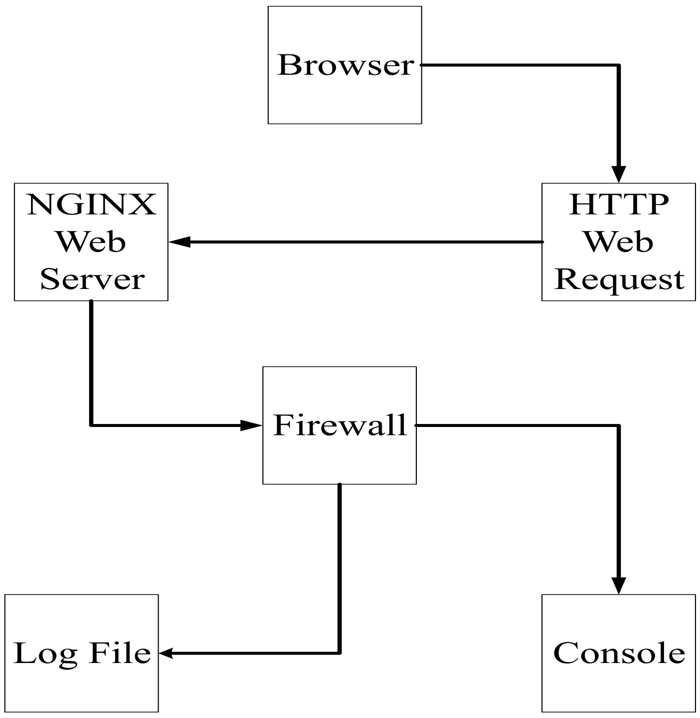


Figure 4.3: DFD diagram of Project

4.2.1 Conceptual or Logical Design

It shows high level work flow between different components of the system and its modules. It also describes how users will be influenced with it. The figure 4.4 describes the components of the modules and their relationship with the log file and within modules. Package Diagram of our application is given below:

4.2.2 Process

This is basic work flow of the system. It describes the run time view of the system. It shows the interaction of user with the system and inter-system interaction of different subsystems. In figure 4.5 shows the detailed process of interaction between the different components of the application.

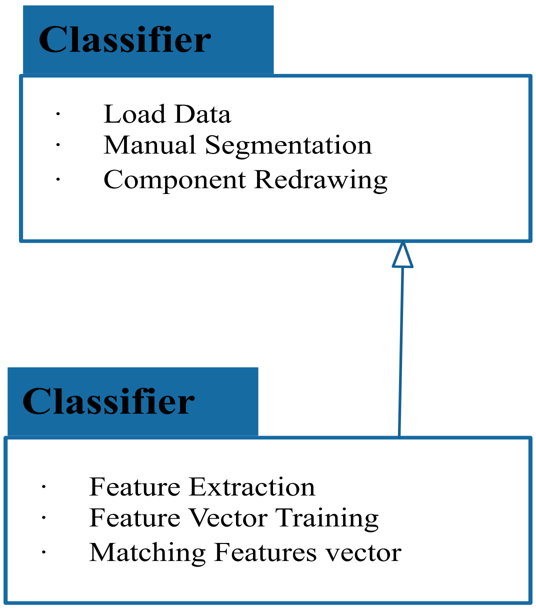


Figure 4.4: Package Diagram of the Application

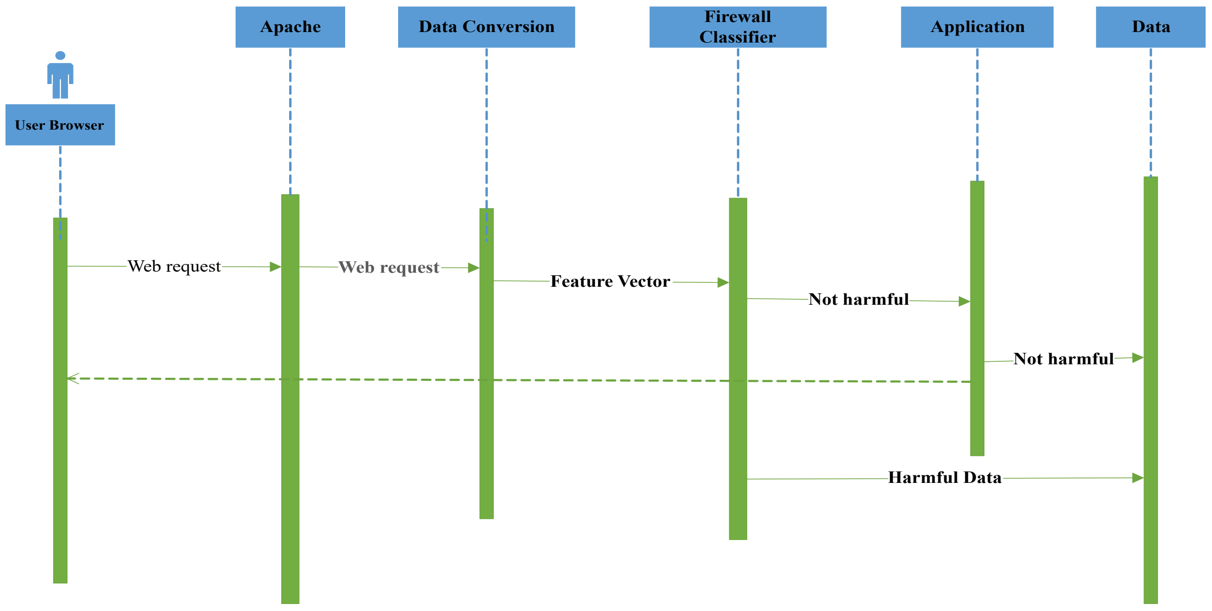


Figure 4.5: Sequence Diagram

In process stage there is a Class figure 4.6 that shows the links between the classes of our system and their functionality in different modules.

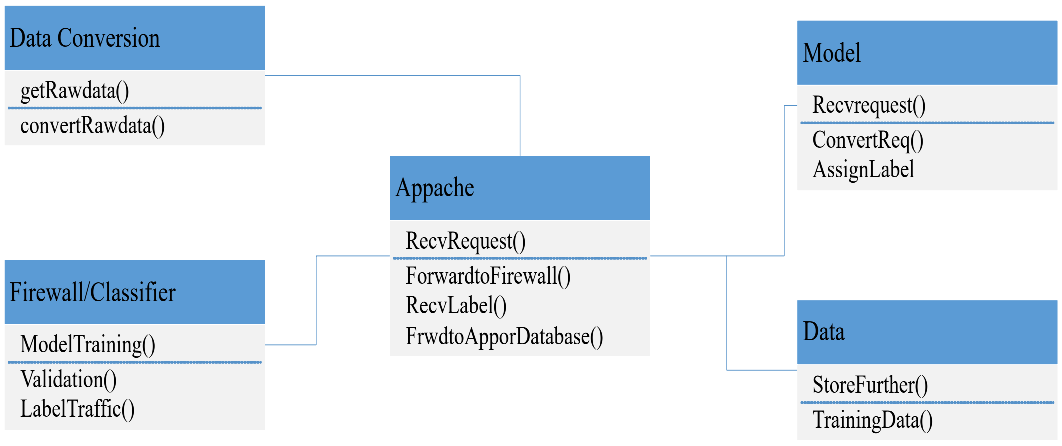


Figure 4.6: Class Diagram of the Application

4.2.3 Physical

Physical diagrams give the physical view of the system, external components and how they are connected with each other. Deployment Diagram of our application is given in figure 4.7:

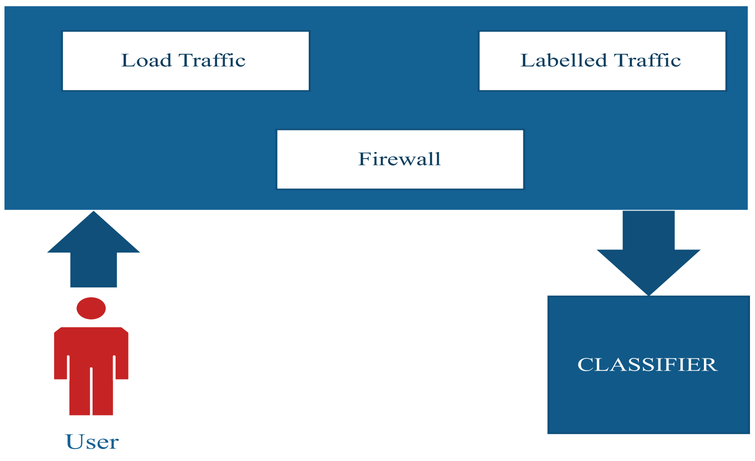


Figure 4.7: Deployment Diagram of the Application

4.2.4 Security

As firewall is itself a security application so we do not need any external security mechanism. However, application user must make sure that firewall is only being downloaded from the official source because a small tampering with the source code can lead to big issues for web application developers. Normally all the data is also secured because of Linux file permissions, and it is recommended to set a strong root password for the Linux machine.

4.3 External Interfaces

External interface that deals with the user of this application is command line, as it is a Linux based application it does not need any graphical user interface, everything is managed through the command line. Our application has following external interfaces requirements.

- Linux Server to run the firewall.
- Remote management through Putty on windows.
- Remote management through the console on Linux.
- Internet connected is required to access the server machine remotely.

Chapter 5

System Implementation

In computer science, system implementation is a process of defining how the system should be built, its physical and system design. It is the process of realization of an application, execution of a plan, ideas, algorithm, design, and model etc. This system has been implemented on Linux and it is a web application firewall that use machine learning to detect malicious traffic. Our system use training data specific to each server, on which it gets trained. In figure 5.1 shows the Architecture view of the system.

5.1 System Architecture

The high-level logical representation of the application is shown in System Architecture. In System Architecture, the components of the system and how they are related to each other are shown in pictorial form. How the user interacts with the system and how the system responds to some valid and invalid inputs will also be discussed in this paragraph. The development environment of the application is also discussed in System Architecture. The system is Web Application firewall for Linux based server, can be used with one server instance at a time. The architecture of our system is clearly described in figure 5.1 as discussed earlier Our Application is a firewall for the web applications. Once it is installed there is no need for any configuration and no need of any extra work for the web site security team related to XSS attacks. Our application is automatically installed through auto-install script provided with the install package and becomes fully functional provided there is enough data available on the server for model training. Our system does not need any external database like MySQL, Oracle database. The four major components of our System is Load Training Traffic, Clean the traffic, Train the model and classify the requests. As soon as the model is trained for classification, we enter the second phase of the application. In the second phase, we sniff each web request coming towards the server and put a label, whether it is a malicious request or not.

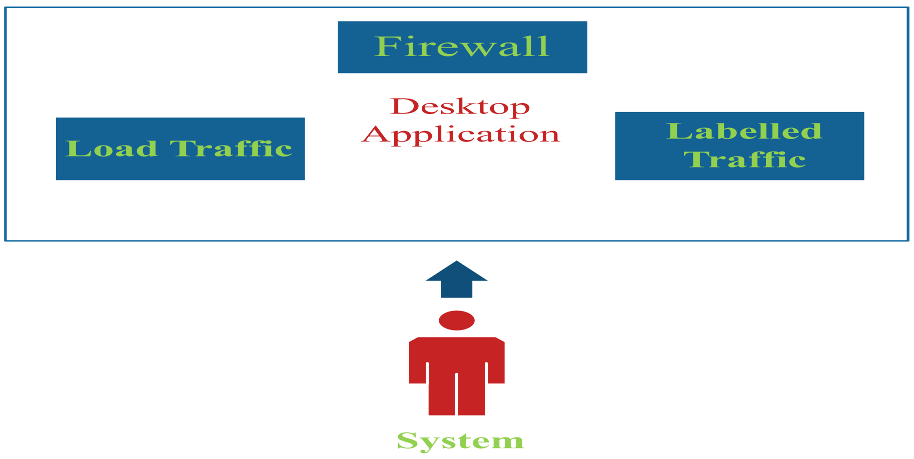


Figure 5.1: System Architecture

5.2 System Internal Components

The system has four major components and every component is linked with each other. Four internal components of our system are Data Acquisition, Feature Extraction, Classification, and Logging. How these components are linked and its processes are shown in detail in figure 5.2 as a System internal component diagram.

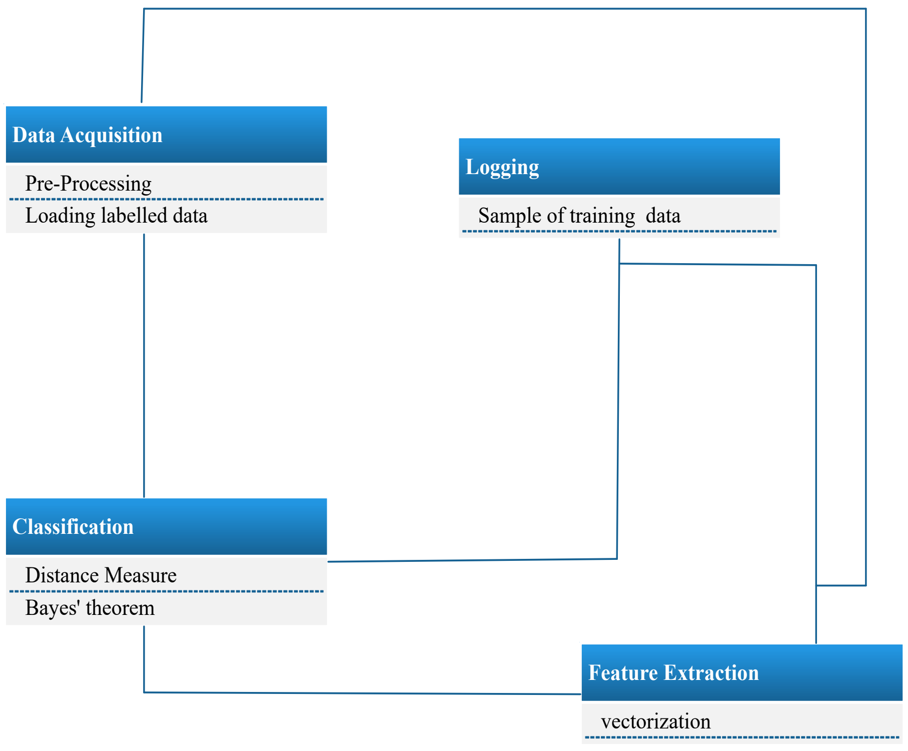


Figure 5.2: System's Internal Diagram of Application

5.2.1 Data Acquisition

As soon as the install script is initiated our application start accumulating website traffic of that specific server in a log file in a pre-defined format. Once the enough data is collected it then triggers the machine learning phase. The log file is in a ModSecurity log file format, so that format of a log is same on different kind of servers.

5.2.2 Feature Extraction

As soon as the data is accumulated, we then convert the raw HTTP traffic and filter useless data from web requests. As this is a case of text classification we use Vectorizer for feature extraction, cleaned web requests are then passed to Vectorizer and it extracts the features based on the frequency of each word present in each document. We also have a predefined data set of XSS traffic which we have used to extract features of a malicious traffic.

5.2.3 Classification

Classification is the core of this application, once we've data available from the server we train our model. Now classification is done based on the words a certain web request can contain. If the frequency of words is closer to the requests of malicious traffic the traffic is labeled as malicious. For example, if web request contains the word "script" 4-5 times it means it is not a good web request. We've used Naive Bayes as classier for this problem because it is a good fit for text classification. With normal classification, we have had some bad results, so we've used a manual enhancer, which enhances the spam of a request if it contains some words which were also present in the malicious traffic.

5.3 Tools and Technology Used

The tools and technology we are using in our system are:

5.3.1 Pycharm

PyCharm is an Integrated Development Environment(IDE) from Czech company Jet Brains used for Python programming language. It's used for unit testing, graphical debugger and supports Web development. PyCharm is cross-platform, supports Windows, MacOS, and Linux versions. Our application is also a Web Application Firewall and language which is used for development is Python, so, we have also used Pycharm for the development of our system.

5.3.2 Clion

CLion. (pronounced "sea lion") is also cross-platform. It's used for languages C and C++ Integrated Development Environment(IDE) for Linux, OS X, and Windows. Our application has a module in C language, so we used CLion for the development of our system.

5.3.3 CPNginx

It's a cPanel NGINX integration plugin. We needed this plugin so that NGINX can be integrated with cPanel server. After that, we run our install script to combine NGINX with the firewall module.

5.3.4 NGINX

It's an open source software for web serving, reverse proxying, caching, load balancing, media streaming. It also works for a proxy server for Emails. Our system has a server so for the proxy works of the web server we used NGINX in our system.

5.3.5 Apache

It's server. It's a world's most used web server software. Our system used it as a server.

5.3.6 Anaconda

It's a freemium open source for the python and R programming language.

5.3.7 Naive Bayes Classifier

We used machine learning Algorithm for classification. In machine learning, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

5.4 Processing Logic Flow

In figure 5.3 our system logic flow starts from the process of loading Raw HTTP request clean the request and make a vector. After the feature extraction, data is classified based on those features. After that process we got the results and maintain the log file for future requests.

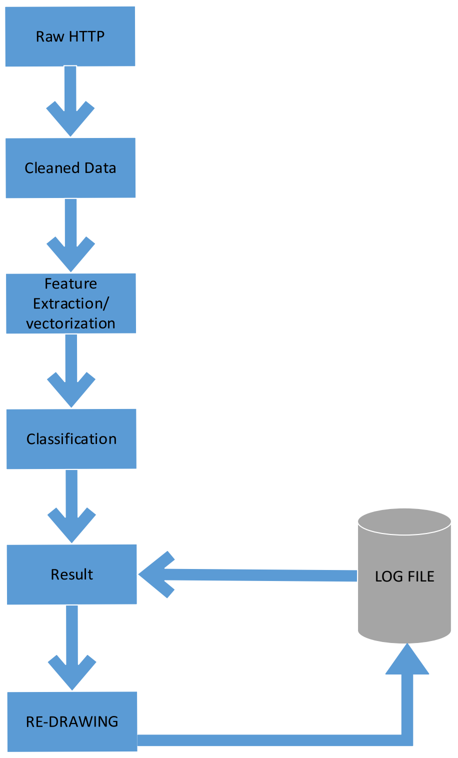


Figure 5.3: Processing Logic Flow Diagram of Application

Chapter 6

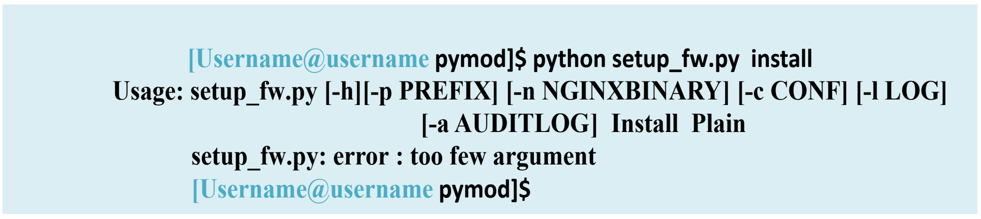
System Testing and Evaluation

6.1 System Testing and Evolution

In this chapter, we discuss the system performance and evaluation. Evaluation of system/application is done by using various system testing techniques. This is the main part of an application as it provides validation about working of the systems and the requirements of the system. To find out if any errors exist or proper exception handling is included. We will discuss a few testing techniques in this chapter. It is a very important component of any application development because we get to know how much of the original specification is met. And we get to find any errors and fix them in a way so that they won't create any issues while application starts running in a production environment. We will discuss the testing that we've applied on our application in this chapter.

6.2 Command Line Testing

Our web application firewall is a Linux based application. As it is installed and run through command line we must make sure proper command line testing is in place. Figure 6.1 show the default run of the install/un-install script.



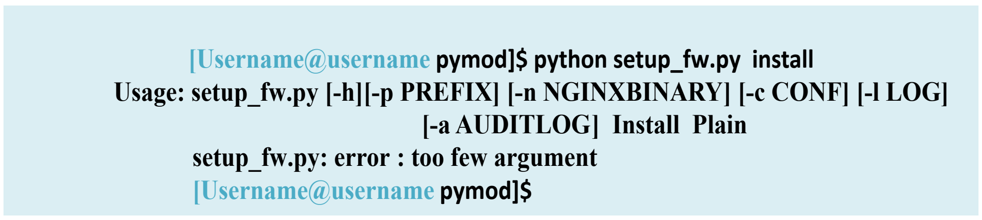
```
[Username@username pymod]$ python setup_fw.py  install
Usage: setup_fw.py [-h][-p PREFIX] [-n NGINXBINARY] [-c CONF] [-l LOG]
                   [-a AUDITLOG]  Install  Plain
setup_fw.py: error : too few argument
[Username@username pymod]$
```

Figure 6.1: Command Line

As we can see this script have few required parameters due to which we've been displayed with an error saying: "too few arguments". Install script has two mandatory arguments.

- Install or un-install.
- Install type.

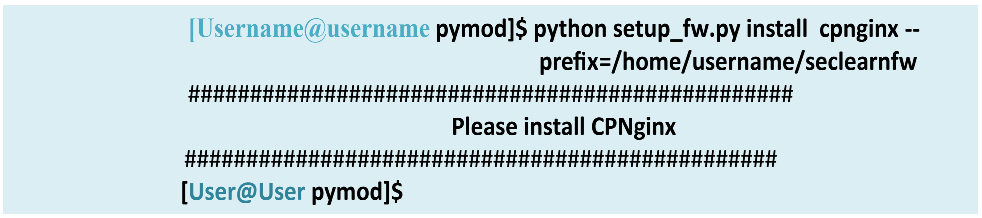
The first argument is whether you need to install or un-install and the second argument is plain or CPNGINX install type.



```
[Username@username pymod]$ python setup_fw.py  install
Usage: setup_fw.py [-h][-p PREFIX] [-n NGINXBINARY] [-c CONF] [-l LOG]
                         [-a AUDITLOG]  Install  Plain
setup_fw.py: error : too few argument
[Username@username pymod]$
```

Figure 6.2: One Argument

In figure 6.2 we've supplied one argument but the script still demands one more argument before it starts running.



```
[Username@username pymod]$ python setup_fw.py install  cpnginx --
                              prefix=/home/username/seclearnfw
#####################################################
                    Please install CPNginx
#####################################################
[User@User pymod]$
```

Figure 6.3: All Arguments

In figure 6.3 we've supplied all mandatory arguments plus one optional argument. But the script first checks if our requirements are met or not. As CPNginx is not installed, it displays the text that CPNginx should be installed first.

6.3 Functional Testing

We've started our testing with functional testing. In functional testing, we've tested our application in various conditions of the server. Different installation scenarios are performed, we've also performed white box, black box and integration testing.

6.3.1 Run Installer

Software: Firewall Setup Script.
Modulation: Installing the Firewall.

First test case is the installation of our application without its pre-requisite CPNginx not installed. As on this stage CPNginx is not installed so it should prompt for the installation of CPNginx and does not continue the installation of firewall.

Table 6.1: Test Case 1:Run

Test case ID	TC-01
Description	The installation of our application without its pre-requisite CPNginx not installed. If not then install the CPNginx
Application for	Web Application users.
Requirements	CPNginx is not installed.
Steps to be Taken	Install CPNginx.
Expected Results	Prompt for CPNginx installation.
Actual Result	Firewall not installed.
Status	Success
Remarks	N/A

6.3.2 Run Installer

Software: Firewall Setup Script.
Modulation: Installing the Firewall.
Second test case is the installation of our application with its pre-requisite CPNginx installed. As on this stage CPNginx is installed so it should successfully install the firewall application.

Table 6.2: Test Case 2:Run

Test case ID	TC-02
Description	On this stage CPNginx is installed so, it should successfully install the firewall application.
Application for	Web Application users.
Requirements	CPNginx is installed.
Steps to be Taken	Run install script.
Expected Results	Firewall successfully installed
Actual Result	Firewall installed.
Status	Success
Remarks	N/A

6.3.3 Run Installer

Software: Firewall Setup Script.
Modulation: Installing the firewall.
User can also run even if the firewall is already installed. So, script must not re-run the install, it should first check if firewall is already installed or not. Because re-installing the firewall on already installed machine can create problems.

Table 6.3: Test Case 3:Run

Test case ID	TC-03
Description	if the firewall is successfully installed then don't run the script.
Application for	Web Application users.
Requirements	Firewall is already installed.
Steps to be Taken	User should run the installer.
Expected Results	Prompt that firewall is already installed.
Actual Result	Firewall is installed.
Status	Success
Remarks	N/A

6.3.4 Setup Connection:

Software: Setup Connection Script.
Modulation: Setting Up connection.
Once the firewall module is installed on NGINX, it will start sending traffic to setup connection script through Unix domain sockets, so we've to run this setup connection script which will train the model and setup Unix domain socket for accepting web requests.

Table 6.4: Test Case 4:Setup

Test case ID	TC-04
Description	If the firewall is successfully installed then we've to run this setup connection script which will train the model and setup Unix domain socket for accepting web requests
Application for	Web Application users.
Requirements	Firewall is already installed.
Steps to be Taken	User should run the installer.
Expected Results	Start accepting connections.
Actual Result	Unix domain socket accepting connections.
Status	Success
Remarks	N/A

6.3.5 Setup Connection

Software: Setup Connection Script.
Modulation: Setting Up Connection.
Setup connection script can also run even if the firewall is not installed, even though it is fine to run setup connection script. So, it should prompt to first install firewall.

Table 6.5: Test Case 5:Setup

Test case ID	TC-05
Description	Setup connection script .
Application for	Web Application users.
Requirements	Firewall is already installed.
Steps to be Taken	Run the setup connection script.
Expected Results	Start accepting connections.
Actual Result	Unix domain socket accepting connections.
Status	Success
Remarks	N/A

6.3.6 Un-install Firewall

Software: Un-install Script.
Modulation: Un-installing Firewall.
We also have a software un-install script, on Linux we've to manually un-install script. That is why a separate un-install script is needed to un-install the firewall. There can be many possible errors while un-installing which should be taken care off. It should check if firewall is already installed or not, and make sure all related files are completed removed and old setup is back as it was before the installation of firewall.

Table 6.6: Test Case 6: Un-install

Test case ID	TC-06
Description	We also have a software un-install script, on Linux we've to manually un-install script.
Application for	Web Application users.
Requirements	Firewall is already installed.
Steps to be Taken	Run the un-install script.
Expected Results	Firewall un-installed.
Actual Result	Firewall is successfully un-installed.
Status	Success
Remarks	N/A

6.3.7 Un-install Firewall

Software: Un-install Script.
Modulation: Un-installing Firewall.
Un-install script can also be executed even if the firewall is not installed and can delete potentially important Linux related files, so it should only run if the firewall is properly installed.

Table 6.7: Test Case 7: Un-install

Test case ID	TC-07
Description	Un-install script can also be executed even if the firewall is not installed.
Application for	Web Application users.
Requirements	Firewall is not installed.
Steps to be Taken	Run the un-install script.
Expected Results	Prompt firewall is not installed.
Actual Result	Prompted that firewall is not installed.
Status	Success
Remarks	N/A

6.4 Software Performance Testing

Performance testing checks the overall performance of the software application. In this testing, we've taken data from two production servers and ran the metrics accuracy test. We will see that how these two data sets performed with our model.

6.4.1 System Accuracy Percentage Rate

Accuracy percentage is used to check how our machine learning model performed on the training data.
Data from server 1: Without Spam Enhancement: 91%
With Spam Enhancement: 96 %
Data from server 2: Without Spam Enhancement: 95%
With Spam Enhancement: 99%

6.5 Exception Handling

Exception handling helps the application to remain unaffected in case the normal flow of the program is not executed. Since the firewall is a network application Exception handling was a must, exception handling done in our system is explained below.

- If for some reason application is not able to obtain a Unix domain socket, then normally it throws an error. We have enclosed it in a try/catch block to avoid any unexpected behavior.
- If any of the required files are not present (Data-set file, NGINX Configuration files) application should be able to produce their own copies from predefined formats.
- When data transformation is going on, there can be some data which is binary or not UTF-8, for that we've first converted every piece of data in UTF-8 format manually, and if conversion is not successful we discarded the input.
- On the NGINX module part of the application, we had to check if any IO operations are successful or not. Files are being opened properly because in C there is very less unbuilt testing so we had to manually check if everything is in order.

6.6 Usability Testing

Usability testing is by the user of the application. That how easy was it to install, use and un-install the application. We also get to know if application met the user requirements.

6.6.1 User Friendly Command Line Interface

Command line interface make it easy for the first-time user, it provides an easy to understand help message if any of the required inputs are missing.

6.6.2 Easy to Use

Our Firewall is very easy to use, unlike other firewalls there is no need to define signature rules because it works on machine learning to detect malicious traffic.

6.6.3 Easy to Learn

It is very easy to learn because most of the hard work is done by the install script. Any new or old user can get used to it easily.

Chapter 7

Conclusions And Future Work

The project " Cross Site Scripting (XSS) Web Attack Detection by Machine Learning " is designed for the help of Web Applications users/developers to insure their web application security through automatic system. This application is very useful for the web applications users/developers and computer sciences students. It has been great learning experience for us as developers specially the module in which we used machine learning. Our application is very simple and easy to understand for common web application users. People can easily handle this application because it is very user-friendly, hence there is no need to have any special training and no need of any extra work to install that application as installation is performed through a single command. After working on this project, we learned about different kind of possible attacks a web application can face and how easy it is for an attacker to find a loophole and steal important information. It is a vast field and with the passage of time our application (machine learning mode) will get more mature and we will expect models which result in the good classification of attacks.

7.0.1 Future Enhancements

Currently, this firewall only stops XSS (Cross Site Scripting) attacks which are only one of the many web attacks present these days. In future, we are planning to train models on SQL Injection Attacks, DOS (denial-of-service) Attacks and many other web attacks. We are also looking to develop this module not only for Apache, NGINX but for other servers like Caddy Web server and Lite Speed Web. In future, we would also like to develop a GUI based log file analyzer where a user can easily analyze log files because it is hard to analyze log files for a naive web server user. Not only on the web server level, we can also create a firewall of this type on network interface level, that may require a lot of time and testing but it can be done and in future, we may try to implement it on interface level. Currently, our machine learning model depends on the data collected from the server. In future, we are planning to gather data from all the servers it is installed on (normal and malicious data) to make it data independent as well. That could be a really great improvement because if we’ve malicious data from various sources it can improve the accuracy of the model as well.

Bibliography

[1] Fernando Magno Quintão Pereira Roberto S. Bigonha Andrei Rimsa, Marcelo D'amorim. "efficient static checker for tainted variable attacks", February. Cited on p. 1.

[2] Ahmed Toumanari. El Moussaid, Nadya ElBachir. Web application attacks detection: A survey and classification., nov 2014. Cited on p. 5.

[3] K. Sathiyakumari. Krishnaveni. Multiclass classification of xss web page attack using machine learning techniques., November. Cited on p. 4.

[4] Hee Beng Kuan Tan Lwin Khin Shar. "predicting common web application vulnerabilities from input validation and sanitization code patterns, ", September. Cited on p. 1.

[5] Nuno F. Neves Medeiros, Ibéria and Miguel Correia. Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. Cited on p. 4.

[6] Angelo Eduardo Nunan. Automatic classification of cross-site scripting in web pages using document-based and url-based features,. Cited on p. 4.

[7] Nalini N. Opularam Prakash Bhanu. Cross site scripting security vulnerabilities and risk mitigation using enterprise security api for web-apps. Cited on p. 5.

[8] R. Sekar. An efficient black-box technique for defeating web application attacks, nov 2009. Cited on p. 3.