

Context Aware Honeypot for Cross-Site Scripting attacks using Machine Learning Techniques

A thesis submitted

in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Shubham Aggarwal

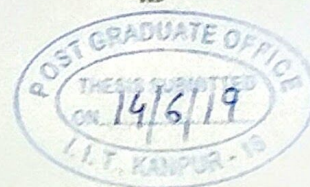


to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

June, 2019

CERTIFICATE



It is certified that the work contained in the thesis titled Context Aware Honey-pot for Cross-Site Scripting attacks using Machine Learning Techniques, by Shubham Aggarwal, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof Dr. Sandeep K. Shukla

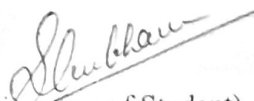
Department of Computer Science & Engineering

IIT Kanpur

June, 2019

Statement of thesis preparation

1. Thesis title CONTEXT AWARE HONEYPOT FOR CROSS-SITE SCRIPTING ATTACKS USING MACHINE LEARNING TECHNIQUES
2. Degree for which submitted M.TECH
3. The thesis guide was referred to for thesis preparation. Yes ~~No~~
4. Specifications regarding thesis format have been closely followed. Yes ~~No~~
5. The contents of the thesis were organized according to the guidelines. Yes ~~No~~


(Signature of Student)

Name : Shubham Aggarwal

Roll No. : 17111043

Department : CSE

ABSTRACT

Name of student: **Shubham Aggarwal** Roll no: **17111043**

Degree for which submitted: **Master of Technology**

Department: **Computer Science & Engineering**

Thesis title: **Context Aware Honeypot for Cross-Site Scripting attacks
using Machine Learning Techniques**

Name of Thesis Supervisor: **Prof Dr. Sandeep K. Shukla**

Month and year of thesis submission: **June, 2019**

In recent years, there has been an extensive research in Honeypot Technology, primarily to detect external threats and to collect threat intelligence. Honeypot can be defined as a system that attracts hackers towards itself. It is possible to deflect the hackers to a honeypot, monitor the ongoing and completed processes introduced by hackers. Basically, a honeypot is like a trap machine which pretends to be a real system so that an attacker is encouraged to break into it and deliver a payload on it. The purpose of the Honeypot is to analyze, understand, view and track hacker's behavior to create threat intelligence which will enable us to secure our system. If we deploy two or more honeypots on our network then it becomes a honeyfarm.

When it comes to an organization, the first line of defense is the intrusion detection system to defend against threats and after that a firewall to defend against external threats. If the intrusion detection system and firewall are well-configured then they can create a huge difference between a breach or a defense. When we use algorithms and statistical data to learn patterns from data without programming explicitly it is called Machine Learning. So in this thesis, we try to improve the intrusion detection system's capabilities by using some well-known machine learning techniques.

After a successful detection, we work on preventing the system from the attacker.

For that, we use an intrusion prevention system which produces a honey farm to neutralize the attack and along with which we try to extract quality payloads from the attackers so that we can prepare for a further attacks.

Dedicated to my parents

Acknowledgements

I stand hereby at the end of my two years M. Tech journey; a journey that has taught me so many things yet I do not have words to express my sincere gratitude to Dr. Sandeep Kumar Shukla, my thesis advisor during the course of the program. His constant counsel and support helped me steer through many impediments. His patience and motivation assisted me whenever my task felt too daunting. His immense pool of knowledge was always at my disposal as I could direct my smallest query towards him and receive prompt and precise answers. Dr. Shukla's unflagging faith in me was my biggest moral pillar, against which I learned the most during tough times. I am forever grateful to Dr. Shukla for being my mentor and seeing me through it until the end.

I'm also thankful to Mr. Rohit Negi for giving me ideas and working alongside with me. Without his passionate participation and input, I couldn't have completed it on time. My juniors Vishal Chourasia and Shriyansh Raj Mishra for helping me out with the Machine Learning part.

My sincere thanks to Ms. Aradhana Yadav for supporting me in the times when I wasn't sure that I can succeed.

Contents

List of Tables	xv
----------------	----

List of Figures	xvii
-----------------	------

1	Introduction	1
1.1	Motivation	1
1.1.1	Sony	2
1.1.2	Aadhaar	2
1.1.3	eBay	3
1.2	Problem Description	3
1.3	Goals	4
1.4	Overview of This Thesis Report	4
2	Background	7
2.1	Honeypot	7
2.1.1	What is a Honeypot?	7
2.2	Honey Farm	8
2.2.1	What is a Honey farm?	8
2.3	Intrusion Detection System	8
2.3.1	What is an Intrusion Detection System?	8
2.3.2	Types of IDS	9
2.4	Intrusion Prevention System	10
2.4.1	What is an Intrusion Prevention System?	10

2.4.2	Type of IPS	10
2.5	Cross-Site Scripting	11
2.5.1	What is Cross-Site Scripting?	11
2.5.2	Classification	11
2.6	Machine Learning	13
2.6.1	What is Machine Learning?	13
2.6.2	Support Vector Machine	13
2.6.3	Artificial Neural Networks	14
2.6.4	Recurrent Neural Network	15
2.7	Summary	17
3	Context Aware HoneyXSS	19
3.1	Design	19
3.1.1	Basic Design	19
3.1.2	Framework	19
3.1.3	Development Tool Setup	21
3.1.4	Data Collection	22
3.2	Implementation	23
3.2.1	Host Intrusion Detection Prevention System (HIDPS)	23
3.2.2	Context Aware Honeypot	24
4	Experiments	27
4.1	Vectorization	27
4.2	Classification	28
4.3	Comparison with other	32
4.3.1	Detecting XSS Using ML	32
4.3.2	DeepXSS	33
4.3.3	Multiclass Classification	33
4.3.4	XSSClassifier	34
4.4	Context-Aware Honeypot	34

4.5	Summary	37
5	Related work	39
6	Conclusions	41
6.1	What can be done next?	41
	References	43

List of Tables

3.1	Data Set	22
4.1	Accuracy by different Bag of Words	27
4.2	K-Nearest Neighbors	29
4.3	Accuracy by different Classifiers(1)	30
4.4	Accuracy by different Classifiers(2)	30
4.5	Accuracy by different Classifiers(3)	31
4.6	Precision of Linear SVM with count occurrences	32
4.7	Best result	32
4.8	Comparing with Fawaz A. Mereani's research	33
4.9	Comparing with Yong Fang's research	33
4.10	Comparing with S.Krishnaveni's research	33
4.11	Comparing with Shailendra Rathore's research	34
4.12	Attacks Blocked	35

List of Figures

2.1	Block diagram of honeypot	8
2.2	Honeypot Farm	9
2.3	Network IDS and Host IDS	10
2.4	Host IPS and Network IPS	11
2.5	Reflected XSS	12
2.6	Stored XSS	13
2.7	SVM	14
2.8	ANN	15
2.9	RNN	16
2.10	LSTM	16
3.1	General Design of Honeypot	20
3.2	Framework of Main System	20
4.1	K-Nearest Neighbors	30
4.2	LSTM results	31
4.3	Classifiers Accuracy	31
4.4	Attacks Blocked	36
4.5	At 80% similarity in ContextAware Honeypot	36

Chapter 1

Introduction

With the introduction and expansion of Internet and its services, our lives have changed a lot. But this popularity has also attracted a small community which has exposed the need for the security of the internet. Unfortunately, this need has been increasing progressively. The perception of linking the world through a basic structure has brought the computer systems to be targeted through malicious activities which threaten the security of internet communications and transactions. Overtime, numerous tools have been developed by security researchers to protect the user and their system from the members of this community but it was easily realized that 100% security can never be achieved. Still, both the communities have been evolving and finding innovative ways to take the lead. While security researchers can learn from the past and use it to prevent the main system from attack, the attackers always try to find new vulnerabilities to take things under their control. And that is why the idea of Honeypots was introduced in the first place.

1.1 Motivation

In recent years, many network security tools like firewall were developed to tackle the threat which an organization faces on a daily basis through network based attacks. These tools have a specific problem that they can not successfully detect new threat which give attackers with an upper hand. In a sense, they are passive tools. Tools

like firewall divides internal and external network with some protocols which prevents internal resources from external threats. Whereas, IDS/IPS can monitor the internal network and after identifying the threat it can take appropriate actions accordingly. For any organization, they are the first line of defense against any sort of a network attack. However, it lacks the ability to monitor the attacker's behavior or to gather concrete payloads(without harming the system) for further improvement. Some of the past cyber attacks which put forward these problems to the limelight are described below:

1.1.1 Sony

Sony didn't use the firewall on PlayStation Network's servers and also used an old Apache server without applying patches for their PlayStation Network which resulted in compromising sensitive information of 101 million users. This notwithstanding the fact that the company claimed the data to be secured by cryptographic hash [1]. It was one of the biggest data leak of its time which could have been easily prevented. It was also stated that they were already made aware of this problem couple of months ago before the attack, but to save money on simple security measures Sony did not pay any attention to the alerts.

1.1.2 Aadhaar

Aadhaar is one of the largest databases as it is India's national identification system, which means that the database roughly has more than a billion unique entries which also links to other sensitive information such as bank accounts numbers, phone numbers, etc. And it was leaked simply because the API didn't have a secure access [2]. Centre for Internet and Society confirmed that around 130 million Aadhaar numbers along with the sensitive information is available on the Internet [3]. It was later found that it was hacked using a patch for the software which deactivated important security protocols and major issue was that it was freely available at a cost of Rs. 2500 [4].

1.1.3 eBay

A very popular e-commerce site which operates in over 30 countries and has more than 14,000 employees could not stop Cross-Site Scripting (XSS) attacks for 4 years. In 2014, it was posted that attackers were modifying the listings on the site with JavaScripts redirects and proxies. The JavaScript was embedded in the description of the fraudulent ads posted by the attacker and it redirected the victim's browser automatically to the attacker's website where it fetched the same listing from a UK server [5]. Till 2017, same thing was done but now with the large number of account information they collected in 3 years, it has become impossible to remove [6].

To overcome this, honeypots were introduced for which Lance Sptizner said *“A honeypot is security resource whose value lies in being probed, attacked, or compromised.”* [7]

Honeypots were one of the biggest wins for security researchers because instead of spending all the time in making a system secure they could now focus more on learning the complex behavior of the attacker, by giving him a false system to attack. It is like reversing the rules where security holes are left deliberately for the attackers to allow them to take control, which forces them to leave their signatures so that their modus operandi can be understood.

Honeypot is not a security answer to a certain question, but it can be seen as a more adjustable tool which can be used in unique ways.

1.2 Problem Description

As we have seen in the above examples, that organizations rely more and more on their computer networks for speedy growth and capturing the big market which advances in technology keeps opening up. Their dependence on computer systems has opened the gate for a better network environment which should be secure enough so that these organizations can prosper. Traditional network security equipment,

which has a largely passive role, does not adequately protect the current network and meet the required security level in modern times. That is why honeypots were introduced to fill in this essential gap where one can interact and learn from the attacker while safeguarding its resources.

The scope of this study has been has taken a jump further in the development of network security tools, especially honeypot. We want to design a computer network system which can safeguard itself from known and unknown attacks by learning and analyzing the behavior of the attackers. A honeypot unique enough that can make an attacker use his signature payloads over common payloads.

1.3 Goals

Honeypots are the traps for attackers but depending on the complexity of the honeypot we can't decide that if it is too easy for an attacker to escape or too hard to try. So the goal of our research is to develop a honeypot which can set the mark according to the attacker. We want him/her to try again and again but also not want common attacks or an easy escape, so with every possible attempt, we increase the complexity of our honeypot so that we can get good data for further analysis.

There is also a list of sub-goals which is done in this study. Making a better Intrusion Detection System since honeypots are not good enough if we are unable to redirect suspicious traffic to them. For that, we need to find better and faster ways to detect these suspicious packets in the first place. We need an updated data-set consisting of previously known attacks and predict what can be new possible attacks.

1.4 Overview of This Thesis Report

The structure of the remaining thesis is as follows. Chapter 2 provides the background of Honeypot and other tools required or used in this thesis along with a basic explanation of the XSS attack and the modern techniques to classify it. In Chapter 3, we have discussed the design and tools used to make this project and how we

implemented the project using those tools. In Chapter 4, we have analyzed and provided the reasons to select these tools by showing the results of our experiments. Chapter 5 is dedicated to the previous/related work of our security researchers which formed the basis of our work. Chapter 6 concludes the report and give some ideas for the improvement that can be done to this project.

Chapter 2

Background

2.1 Honeypot

2.1.1 What is a Honeypot?

Overtime, people and security researchers have tried to give numerous definitions of honeypot but the best till date has been given by Lance Sptizner, who said “*A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource*” [7]. It is self-explanatory that a honeypot is an “information system resource” that can range from a printer to a server, from a network device to a whole network, anything can be a honeypot. Basically, the aim of designing a honeypot is to make a system that can be attacked in the first place with fake data. But honeypot is not a solution for a lot of problems. As explained earlier, it requires certain tools to transfer suspicious traffic to them and then they will only help you in monitoring and understanding the behavior of an attacker. Surely they also give signatures of an attacker to safeguard further similar attacks.

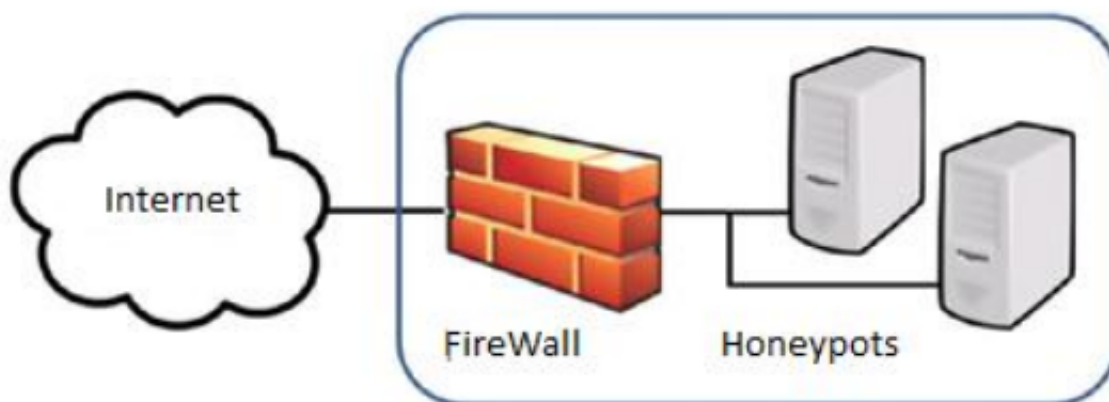


Figure 2.1: Block diagram of honeypot

2.2 Honey Farm

2.2.1 What is a Honey farm?

On a network, two or more honeypots make a honeynet. Generally, it is used to monitor a large or/and more diversified network in which a single pot is not enough. They are generally applied as part of large network intrusion detection system(NIDS). So instead of deploying a large honeypot, we deploy simplified small honeypots which is known as honeypot farming. All the attackers are redirected to one central location which makes it easy to handle. So basically we can say that a honey farm is a combination of multiple analysis tools and different honeypots.

2.3 Intrusion Detection System

2.3.1 What is an Intrusion Detection System?

An IDS is a system which monitors the flow of network traffic. If there is any sort of suspicious activity it prompts a quick alert whenever such an activity is found. But anomaly detection is considered a primary function, some of them can also take action like blocking or forwarding whenever anomalous traffic or a malicious activity is detected. But this also makes them prone to false positives. So an organization

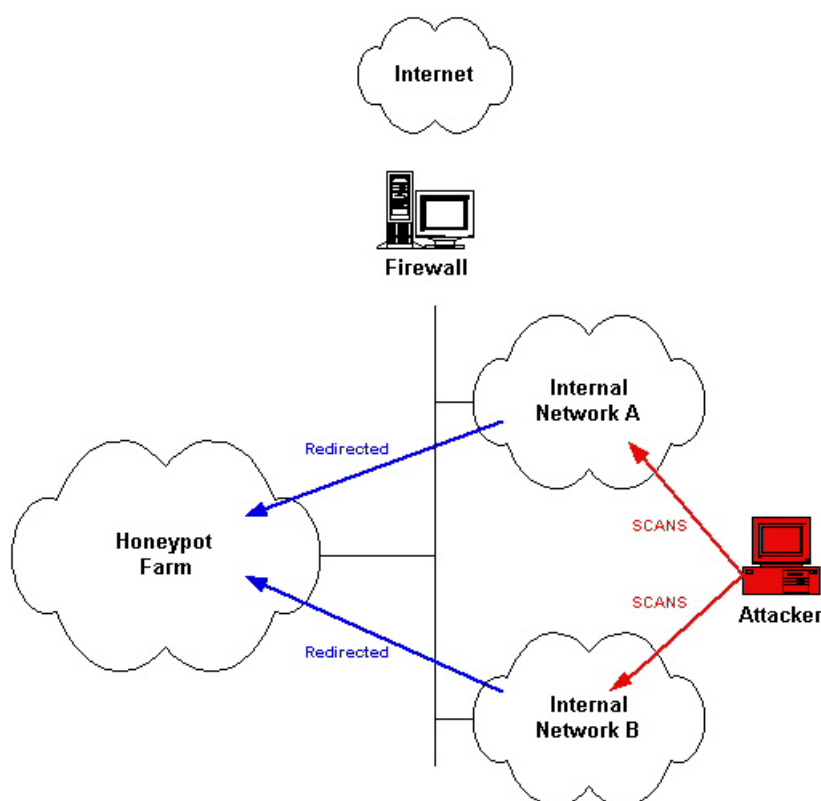


Figure 2.2: Honeypot Farm, source: <https://www.symantec.com/connect/articles/honeypot-farms>

has to work hard to minimize these false positives.

2.3.2 Types of IDS

- **Network IDS (NIDS)** monitors the whole or a part of a network by capturing all the incoming and outgoing traffic. It is usually called passive IDS because it informs the admin after finding/detecting any anomaly in the captured traffic.
- **Host IDS (HIDS)** monitors only the activity of a single host. It is highly customizable according to the host which increases the chances of detecting any suspicious activity compared to NIDS. It also holds advantage of detecting any anomaly which is originated in the internal network which is sometimes missed by NIDS.

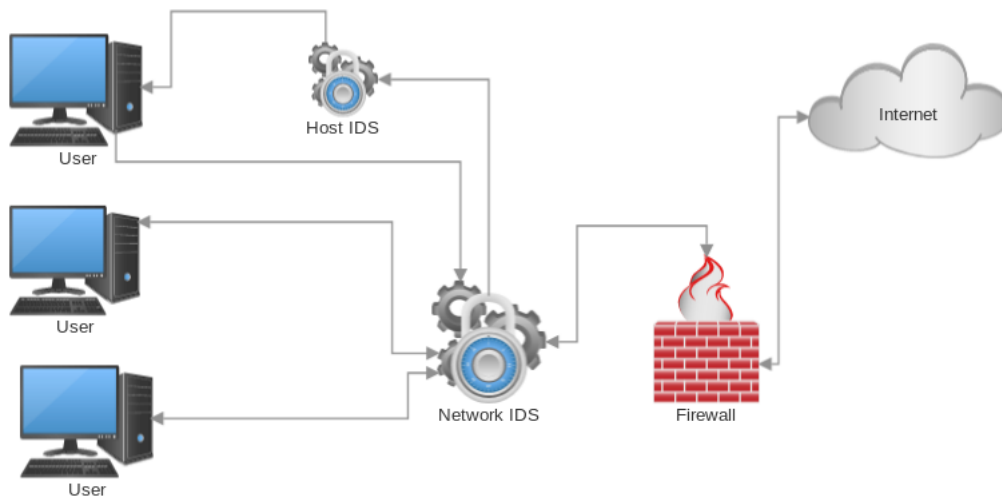


Figure 2.3: Network IDS and Host IDS using SmartDraw

2.4 Intrusion Prevention System

2.4.1 What is an Intrusion Prevention System?

An IPS is a system which monitors the flow of network traffic as well as protect it. It is usually used to protect the system against unauthorized access and DoS attack so that the system should be up for the most of the time.

2.4.2 Type of IPS

- **Network IPS (NIPS)** monitors the whole or a part of a network for malicious activity. By using protocol activity it partitions the network to contain harmful malwares.
- **Host IPS (HIPS)** monitors the activity of a single host. It is highly customizable according to the host which increases the chances of protecting it against any suspicious activity.

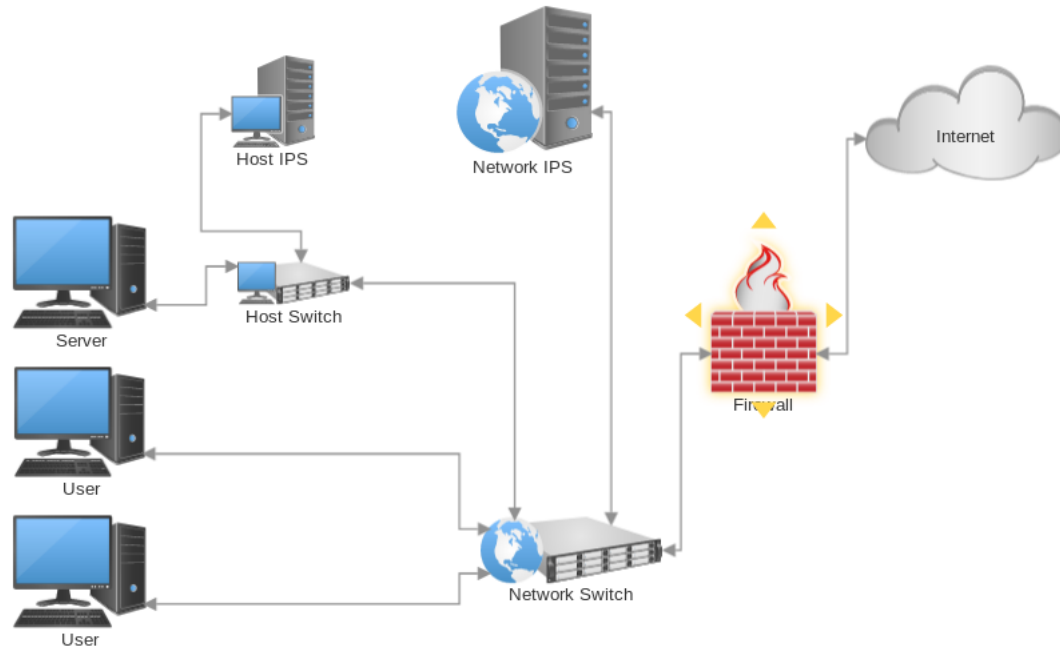


Figure 2.4: Host IPS and Network IPS using SmartDraw

2.5 Cross-Site Scripting

2.5.1 What is Cross-Site Scripting?

XSS attacks are one of the most popular cyber attacks which mostly target web sites and applications in which one can inject code into the web page which is then viewed by other users. Hence an innocent user will execute the code of the attacker. Using this one can bypass access controls by exploiting XSS vulnerability. In most of the cases, they are used to steal cookies containing sensitive information or installing malware. But these days they are used for phishing attacks and browser exploits.

2.5.2 Classification

2.5.2.1 Reflected XSS

Reflected XSS or Non-persistent attack is usually done to steal cookies and other sensitive information by executing the script in the victim's device. This is done by appending the end of the URL with the non-sensible term and script. So when the user clicks the link it will show an error for that non-sensible term and send back the

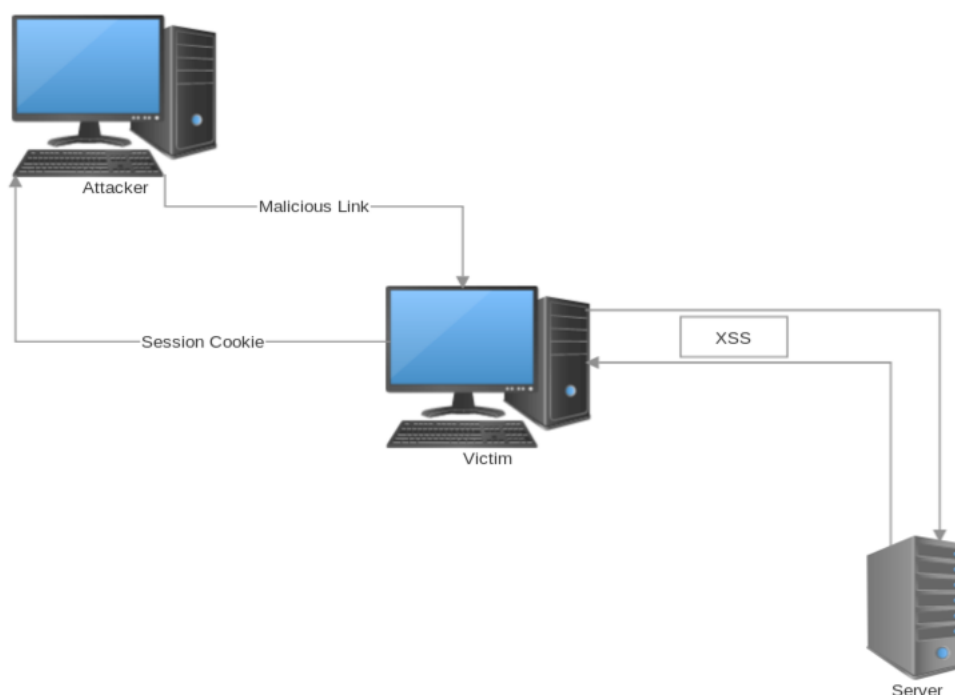


Figure 2.5: Reflected XSS using SmartDraw

request with the malicious script which then executes itself on the user's device.

2.5.2.2 Stored XSS

Stored XSS is also known as Persistent XSS and as the name suggests this attack is done by storing the payload or a script in the page of a website, so when a user loads the page with the script, the script will also be executed on the user's browser which opens a window of numerous attacks.

2.5.2.3 Document Object Module-Based XSS

Document Object Module based XSS or DOM-based XSS attack is done by injecting malicious scripts into DOM by using a malicious URL. In this as well the end of the URL is appended and instead of sending it to the server, it is processed by the browser since these days most of the part is processed on the user side compared to the server side. So after clicking URL, DOM is infected by script and browser executes the malicious code.

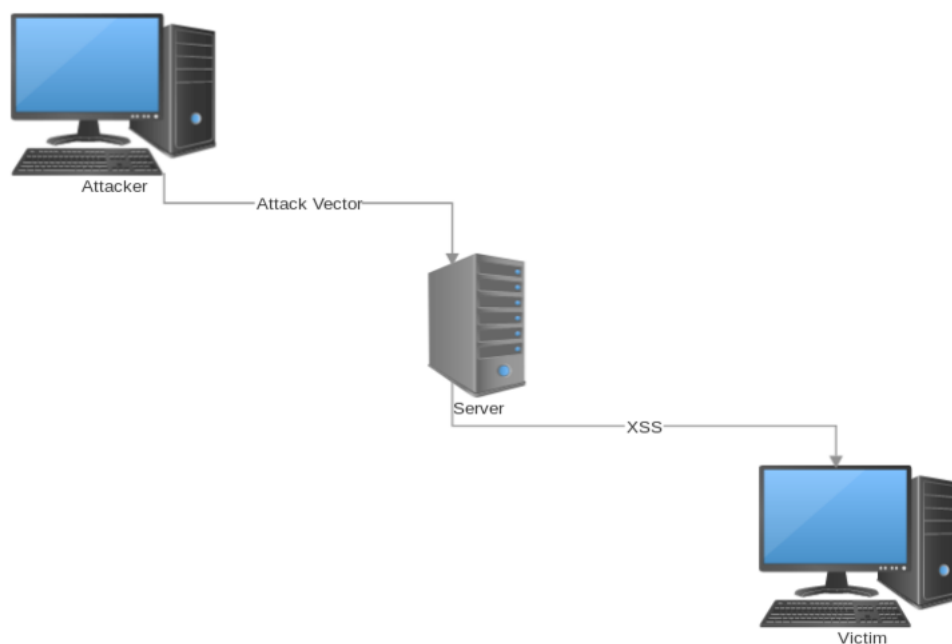


Figure 2.6: Stored XSS using SmartDraw

2.6 Machine Learning

2.6.1 What is Machine Learning?

ML is defined as the use of algorithms and statistical data to learn patterns from data without programming explicitly. This is a subclass of the artificial intelligence domain within computer science. In recent years, humongous amount of raw data(un-structured) has been generated, and machine learning algorithms can help us extract useful knowledge from the data.

2.6.2 Support Vector Machine

SVM are used for classification problems. Generally, it is used to make predictions in a non linearly separable dataset. SVMs fall in the class of supervised learning algorithms, conceptually we project data into some other dimension where we can easily find a plane which linearly separates data into different categories. SVMs use a set of mathematical functions known as kernels, which take the data as input and transform it as necessary. There are different types of kernel functions, for example,

Principle of Support Vector Machines (SVM)

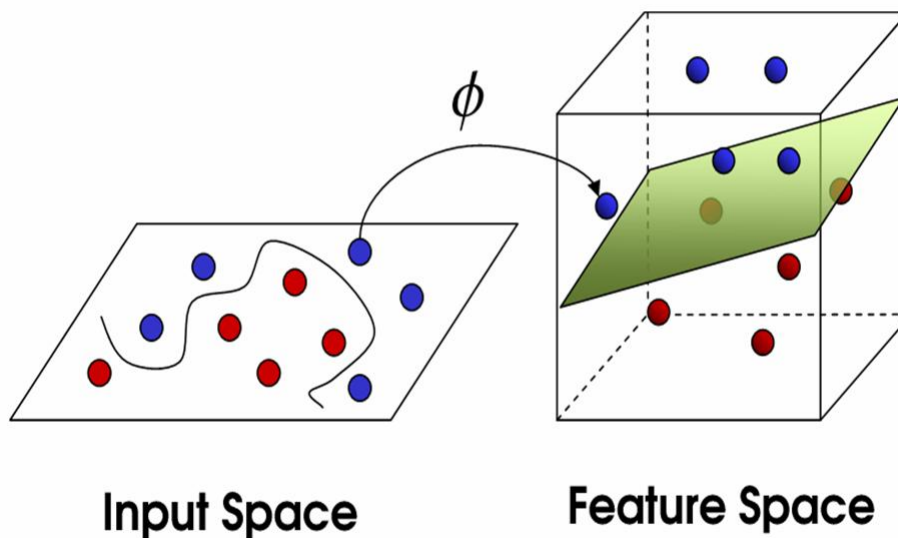


Figure 2.7: SVM, source: <http://crdd.osdd.net/raghava/rbpred/svm.jpg>

linear, nonlinear, polynomials, radial base functions (RBF) and sigmoid.

2.6.3 Artificial Neural Networks

ANN is a mathematical model based on the human neural network. It can be thought of as a set of neurons connected in a layered fashion and each of the neurons has a weight associated with them, which are learned during the training phase. The original idea behind the neural network was to mimic the learning process in the brain. A generic ANN consists of at least 3 layers namely:

- input layer
- 1 or more hidden layer(s)
- output layer

Every neuron in a layer performs weighted summation of its input signals followed by non-linear activation such as sigmoid, tanh, etc. As the network trains, the data flow through the network (feed-forward and back-propagation) adjusts the associated

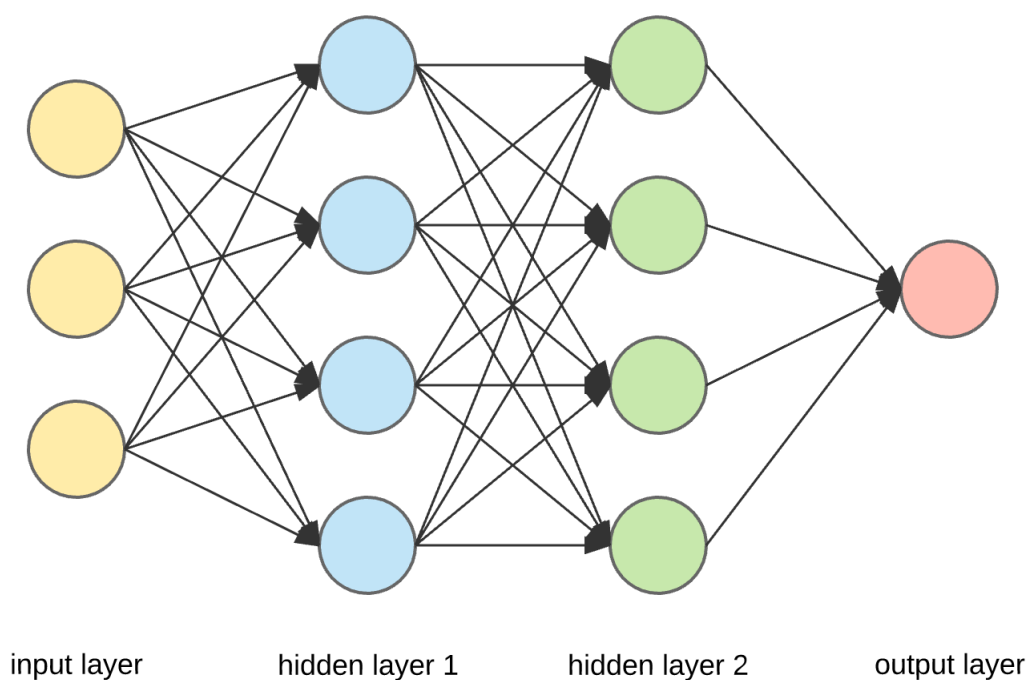


Figure 2.8: Artificial Neural Network

weights which allow the network to have a deeper understanding of the data and performs tasks such as classification, regression, etc.

2.6.4 Recurrent Neural Network

RNN is a category of ANN where a directed graph is formed based on a sequence of connections. They contain loops to persist information and do a similar event for all the elements in the input sequence. The elements of the output sequence in RNNs depend on previous elements or state. Recurrent neural networks are used where any model needs context to generate output for a given input. The API of Vanilla Neural Network (and also Convolutional Neural Network or ConvNets) is too constrained: the input is a particular-sized vector such as an image or a video frame and the output is also a particular-sized vector like a probability vector for other classes. Unlike VNNs, RNNs use their memory to compile input sequences. In RNNs all input are related to each other. This makes them applicable to sequence prediction problems in fields of speech recognition and NLP.

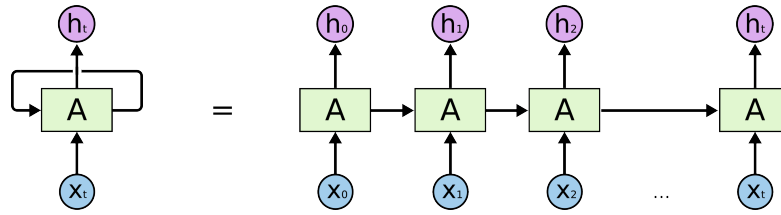


Figure 2.9: RNN, source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

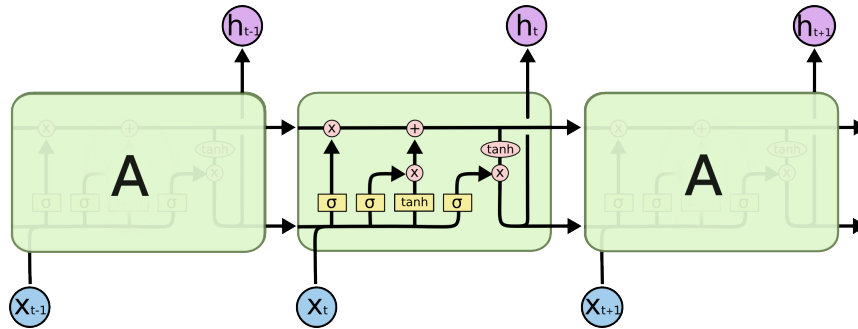


Figure 2.10: LSTM Network, source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

2.6.4.1 Long Short Term Memory Networks

RNNs digest sequential data and the models accuracy by remembering previous relationships among data. But these are designed with limited memory capacity which becomes unreliable when long term context is to be considered. To understand the context better Long Short Term Memory Nets modify the internal structure of the neural unit by partially vanishing shield because LSTM units allow the gradient to be unchanged. In LSTMs, data passes through a mechanism called cell states which helps it to remember relevant features and forget the rest. There are 3 dependencies in the data of a particular cell state.

To generalize, any problem is dependent on,

- **Previous cell state-** data is already in the memory after the last stage
- **Previously hidden state-** it is similar to the output from last cell
- **Input at the current time stage-** it is the new information given at that time

2.7 Summary

The objective to explain all these things is to make a base idea of what our research is all about. We detect the malicious traffic or a possible XSS attempt using a pre-trained LSTM model which will be embedded in our HIDS. Then we combine our HIDS with our HIPS which transfers all that traffic to our honeypot server. Our honeypot server has some analysis tools out of which one is be a pre-trained SVM model which increase the complexity of our honeypot after every attempt.

Chapter 3

Context Aware HoneyXSS

3.1 Design

3.1.1 Basic Design

Here we are trying to protect our main server. The attacker can be present on an internal network or an external network. If he/she is on the external network then there is a high probability that he/she will be blocked by the firewall but if he/she is not blocked by the firewall then both he/she or anyone sitting in the internal network are not preventable by the firewall. For them, we have Host Intrusion Detection-Prevention System (HIDPS) attached to our main server. If anyone wants to access our main server then they have to pass our HIDPS. If HIDPS detects any sort of malicious activity then it will forward it to context switch which is attached to honeypot to prevent the main server from any sort of attack.

3.1.2 Framework

Kali Linux 2019.1 is used to host the virtual machine which also uses Kali Linux 2019.1 on Virtual Box 5.2.22. Since we are using Host IDS/IPS, it is also on our host machine whereas the context switch is on the virtual machine which makes it easier to control the honeypot. And as shown we have kept everything open-sourced.

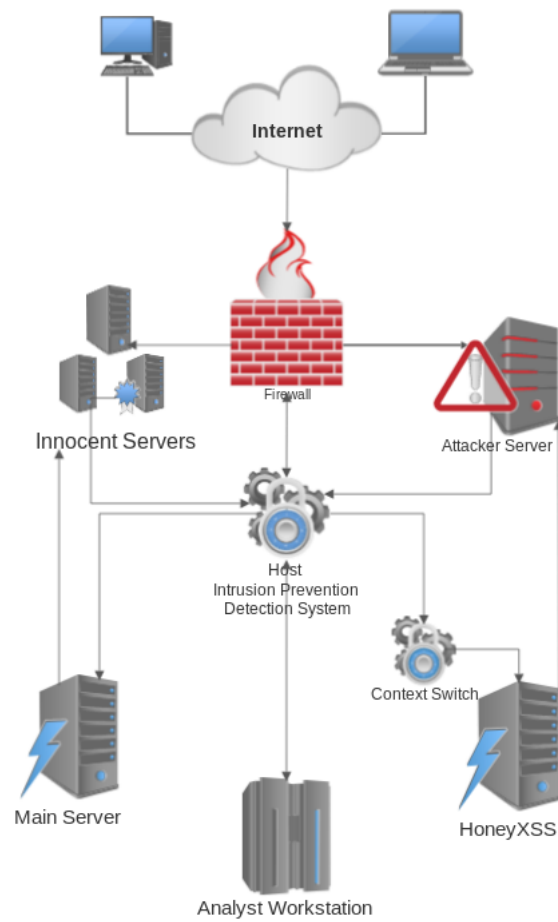


Figure 3.1: General Design of Honeypot using SmartDraw



Figure 3.2: Framework of Main System using draw.io

3.1.3 Development Tool Setup

3.1.3.1 Python Libraries Used

- **Scikit-learn** is a libre software commonly used for machine learning. Most of the models used for classification is done using this library and with its dependency on NumPy and SciPy.
- **Keras** is an open-source library commonly used for neural-network. We get the best result using this library but tensor flow is needed to run this.
- **OS** is an open-source library commonly used for executing operating system command from the python scripts so that dependency from bash scripts can be removed.

3.1.3.2 Other Languages Used

- **Hypertext Preprocessor** is the most commonly used scripting language which is most suitable for the development of the website and can be easily embedded into HTML.
- **Hypertext Markup Language** is the most common markup language used for making web pages and applications. Along with Hypertext Preprocessor, it was used to create the main server.
- **Shell Script** is a scripting language which runs on Unix shell (command-line interpreter). It was mostly used in the proper functioning of ost intrusion detection prevention system and context switch.

3.1.3.3 Software and OS

- **Kali Linux** is an open-source Debian-derived Linux distribution OS and widely used for penetration testing and forensics. It also has a pre-configured LAMP stack which gives us Apache server to use.

- **VirtualBox** is virtualization software that installs an operating system as an application on another operating system. It allows running other operating systems in the virtual environment on any operating system.

3.1.4 Data Collection

Data collection was an important job because it was needed for our Host Intrusion Detection-Prevention System (HIDPS) and context switch classifiers. The data was collected from 4 different popular sources.

- **xssed.com** is the biggest data set of all the known Cross-Site Scripting (XSS) attacks done. So we crawled the site and gathered most of the http commands that are known XSS attacks.
- **ismailtasdelen/xss-payload-list** is a cheat sheet of Cross-Site Scripting (XSS) attacks publicly available at Github. It consists of unique and new XSS attacks possible.
- **HTTP data set CSIC 2010** is a very popular data set used by many researchers. We only took the benign data for training our model.
- **Morzeux/HttpParamsDataset** is a labelled data set publicly available at Github. We only took the normal data and Cross-Site Scripting (XSS) attack for training our model.

Then we removed the duplicates and combined all the data in one file.

XSS	Non-XSS	Total
34873	28613	63486

Table 3.1: Data Set

3.2 Implementation

3.2.1 Host Intrusion Detection Prevention System (HIDPS)

HIDPS was implemented on our main server which consists of some vulnerabilities. Our HIDPS constantly checks the logs for any sort of malicious activity and if it detects anyone trying to exploit the induced vulnerability then it forwards that Internet Protocol to honeypot server which is just a replica of the main server.

3.2.1.1 Detection

We are continuously checking the access log for any new entry, and for every new line, we do the following things.

1. **URL Decoding** is a method to decode the encoded information from a Uniform Resource Identifier. URL encoding is also called percent encoding because the character can be encoded using the percent “%” sign, followed by its hexadecimal digits which makes it harder to understand and can be used as URL obfuscation. For example “&” can also be written “%26” or *https://extract.pw* can be written as *https://%65%78%74%72%61%63%74%2e%70%77*
2. **Tokenization** First, the payload is separated from the URL which then is passed through regular expression so that we can remove all the special characters. Then we take pre-trained tokenizer of Keras which we trained at 10000 tokens at a time and p payloads padded till 1500 words.
3. **Classification** After tokenization, we are left with a sequence which we pad to 1500 and generate sequence matrix. Now we pass this matrix to our pre-trained long short-term memory (LSTM) model which predicts that it is a Cross-Site Scripting (XSS) attack or not. Both the model is trained by using the whole data set.

3.2.1.2 Prevention

When the alert is generated we use Internet Protocol Tables to forward the malicious packets from that Internet Protocol address to the Internet Protocol of Context-Aware Honeypot. Here we use the concept of white-listing in Internet Protocol Tables. After finding the vulnerability plenty of new Internet Protocol also start attacking which makes our system under threat since if there is a sudden increase in traffic, it will definitely cause a delay in detection which can give away the window some new attacker needs to exploit our vulnerability. To avoid this we start forwarding all the Internet Protocols to honeypot server except for the ones who are trusted and are in the white list of Internet Protocol Tables.

3.2.2 Context Aware Honeypot

At Context Aware Honeypot we have introduced context switch which checks the malicious logs of the honeypot. The main idea of the context switch is to constantly increase the complexity of the honeypot. It can be easily done by blocking the type of attacks which are too common and gradually increasing those numbers.

3.2.2.1 Detection

We are continuously checking the access log for any new entry, and for every new line, we do the following things.

1. **URL Decoding** is a method to decode the encoded information from a Uniform Resource Identifier. URL encoding is also called percent encoding because the character can be encoded using the percent “%” sign, followed by its hexadecimal digits which makes it harder to understand and can be used as URL obfuscation. For example “&” can also be written “%26” or *https://extract.pw* can be written as *https://%65%78%74%72%61%63%74%2e%70%77*
2. **Tokenization** First, payload is separated from the URL which then is passed through regular expression so that we can remove every special character. Later

we split each word and convert them into separate tokens.

3. **Classification** First, we pass all the tokens to our pre-trained model of Bag of Words where each token is given a specific value. After getting these values we then pass it through the pre-trained model of linear Support Vector Machine (SVM) which gives the probability of how common the payload is. Both of these models are trained on the cheat sheet which consists of different Cross-Site Scripting (XSS) attacks.

3.2.2.2 Extraction

After getting the probability of the attack we change the vulnerability accordingly. If the value of probability is crossing the certain threshold then it is executed and the threshold is updated. Else the attack is blocked and the threshold remains the same. All the attacks are saved in a backup which after classification can be used to train our existing model to predict better.

Chapter 4

Experiments

4.1 Vectorization

We first tested some basic approaches in Bag of Words with Linear Support Vector Machine [8].

1. **Count Occurrence (CO)** is used where the value of the word is relative to the occurrences of that word.
2. **Normalized Count Occurrence (TF)** is used to make a bias model where the high frequency can dominate the result.
3. **Term Frequency-Inverse Document Frequency (TF-IDF)** makes a bias model where the low frequency can dominate the result.

From Table 4.1 we can say that Count Occurrence is performing marginally better than the other 2 approaches, so we will use Count Occurrence as our Vectorizer.

	CO	TF	TF-IDF
1st fold	0.9961815665	0.9950837669	0.994749654
2nd fold	0.99575179	0.9955608592	0.994749654
3rd fold	0.99575179	0.9944630072	0.9960381862
4th fold	0.9961815665	0.9949403341	0.9954176611
5th fold	0.9955610711	0.995322419	0.9953701494
Average	0.9958855568	0.9950740773	0.9952650609

Table 4.1: Accuracy by different Bag of Words

4.2 Classification

We have used Count Occurrence Vectorizer for 7 different classifiers and Long short-term memory is using keras default tokenizer where testing is done by splitting data in a ratio of 67:33 which is almost 2:1. The classifiers we tested on are

1. **Support Vector Machine (SVM)** is an algorithm which tries to find a maximally separating hyperplane for two separable classes.
2. **K-Nearest Neighbors** is a pattern recognition algorithm which tries to predicts based on the values of the neighbors.
3. **Random Forest** as the name suggests it is a combination of multiple random decision trees which are combined to give a more accurate prediction.
4. **Gradient Boost** is a boosting technique which combines weak classifiers into a strong classifier by sequentially adding the previous model's under-fitted predictions to the ensemble, which makes sure to correct the errors made previously.
5. **Multilayer Perceptron** consist of many perceptrons in a layerwise fashion which is used to learn a highly nonlinear classification boundary for two or more classes.
6. **Decision Tree** is a graph that is branched with values provided and predicts the outcome at the end.
7. **Adaptive Boost** is a boosting technique which combines weak classifiers into a strong classifier by giving preference to under-fitted training instances in the last model.

K	Accuracy(%)
1	98.92
2	98.89
3	70.74
4	75.08
5	97.13
6	97.01
7	74.96
8	74.45
9	73.19
10	74.24
11	98.43
12	98.89
13	99.17
14	99.08
15	98.98
16	99.24
17	73.06
18	74.22
19	99.28
20	98.99

Table 4.2: K-Nearest Neighbors

We tried to get an idea of the accuracy with respect to different number of neighbors. After this analysis we selected two and five because

1. Fawaz A. Mereani's[9] research has used two nearest neighbors to produce a staggering result of 99.75% accuracy but on a different smaller data set.
2. Since three and four did not produced good result so we took the next best which is five for further analysis.

K-Nearest Neighbors

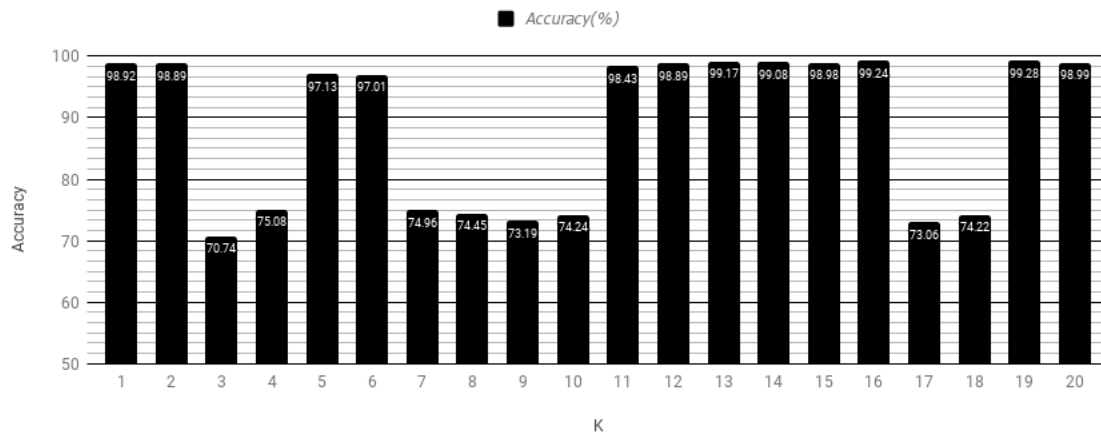


Figure 4.1: K-Nearest Neighbors

	SVM(poly)	2-NN	5-NN
1st fold	0.5568230633	0.9859195265	0.7555725264
2nd fold	0.5531478211	0.9889742733	0.9647272207
3rd fold	0.5509785203	0.7246778043	0.7560381862
4th fold	0.5475156317	0.9843914081	0.7397136038
5th fold	0.5488998139	0.9881622912	0.9712171838
Average	0.5514729701	0.9344250607	0.8374537442

Table 4.3: Accuracy by different Classifiers(1)

	Random Forest	Gradient Boosting	Multilayer Perceptron
1st fold	0.6959095031	0.9871599045	0.9954178798
2nd fold	0.6812085342	0.987112787	0.994988306
3rd fold	0.693952556	0.9864445611	0.9956088015
4th fold	0.7091646778	0.9857756563	0.9957519927
5th fold	0.696004964	0.9863491003	0.9953701494
Average	0.695248047	0.9865684018	0.9954274259

Table 4.4: Accuracy by different Classifiers(2)


```

Using tensorflow backend.
[{"siteMessageMsg=<script>alert(1)</script>" 'xss'}
{"itemId=3220mousseover=alert(1)&document.cookie=29k20bad=x22" 'xss'}
{"uLang=enK2ZkE3cscriptK3EalertK2&document.cookie=29K3C/scriptK-br/>3E'
 'xss'}
{"msg=<script>alert('LastRider-cyberBellona')</script>" 'xss'}
{"q=><script>alert('Xss By Atm0n3r')</script>&submit=Rechercher'
 'xss'}]
ol (63486, 2)

Layer (type)           Output Shape           Param #
-----
Inputs (InputLayer)    (None, 1500)           0
embedding_1 (Embedding) (None, 1500, 50)       500000
lstm_1 (LSTM)           (None, 64)             29440
fc1 (Dense)             (None, 256)            16640
activation_1 (Activation) (None, 256)            0
dropout_1 (Dropout)     (None, 256)            0
out_layer (Dense)       (None, 1)              257
activation_2 (Activation) (None, 1)              0
-----
Total params: 546,337
Trainable params: 546,337
Non-trainable params: 0

Train on 34028 samples, validate on 8567 samples
Epoch 1/10
2019-05-06 15:27:24.443708: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
34028/34028 [=====] - 243s 7ms/step - loss: 0.0558 - acc: 0.9872 - precision: 0.9941 - val_loss: 0.0190 - val_acc: 0.9953 - val_precision: 0.9980
Epoch 2/10
34028/34028 [=====] - 244s 7ms/step - loss: 0.0164 - acc: 0.9957 - precision: 0.9984 - val_loss: 0.0194 - val_acc: 0.9955 - val_precision: 0.9994
2085/20851 [=====] - 69s 3ms/step
Test set
Loss: 0.0218
Accuracy: 0.9953
precision: 0.9995
(myenv) temp@temp:~/Desktop/XSS_analysis/RNN5

```

Figure 4.2: LSTM results

Classifier Comparison

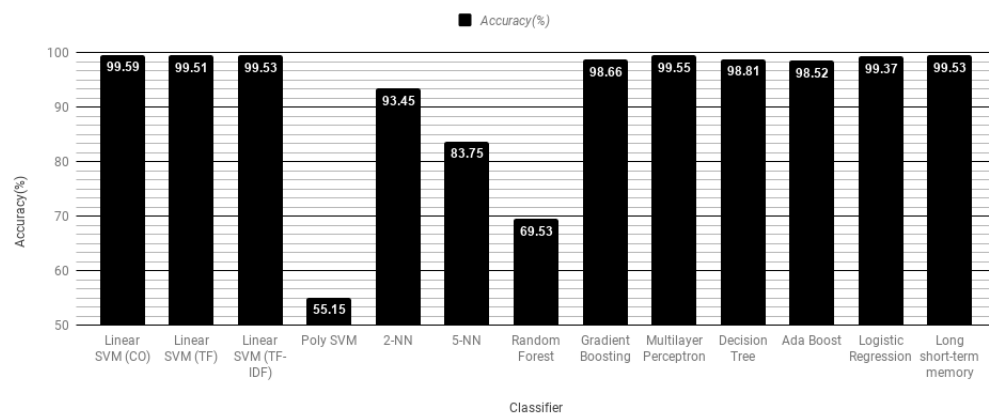


Figure 4.3: Classifier Comparison

	Decision Tree	Ada Boost	Logistic Regression
1st fold	0.9877810128	0.9851081094	0.9946064627
2nd fold	0.9870173261	0.9855369928	0.9935086631
3rd fold	0.9881622912	0.9855854136	0.9930310263
4th fold	0.9893561167	0.9852990311	0.9933174224
5th fold	0.9879242041	0.9844398835	0.9936995847
Average	0.9880481902	0.9851938861	0.9936326318

Table 4.5: Accuracy by different Classifiers(3)

Figure 4.3 is a graph of comparison from the results of Figure 4.2, Table 4.3, Table 4.4 and Table 4.5. From this we can say that the Linear SVM with count

occurrences is the best model in respect of accuracy with 99.59% accurate result prediction. Now we check the precision of our most accurate model.

	Precision
1st fold	0.9992988606
2nd fold	0.9994778522
3rd fold	0.9992108032
4th fold	0.9997375787
5th fold	0.9994776704
Average	0.999440553

Table 4.6: Precision of Linear SVM with count occurrences

Precision of LSTM is only 0.01% better than the precision of Linear SVM. Here is our best result.

Classifier	Accuracy	Precision
LSVM	99.59%	99.94%
LSTM	99.53%	99.95%

Table 4.7: Best result

4.3 Comparison with other

Here we compare our results with the recent research on XSS detection.

4.3.1 Detecting XSS Using ML

If we compare our results with Fawaz A. Mereani's[9] research.

Classifier	Fawaz A. Mereani	Our Result
Linear SVM	96.32	99.59
Polynomial SVM	99.60	55.15
k-NN	99.75	93.44
Random Forest	99.50	69.52

Table 4.8: Comparing with Fawaz A. Mereani’s research

Linear SVM is producing our best result where as 2-NN is producing Fawaz A. Mereani’s best result and we are only 0.16% less accurate.

4.3.2 DeepXSS

If we compare our results with Yong Fang’s[10] research.

LSTM	DeepXSS	Our Result
Precision	99.50	99.95

Table 4.9: Comparing with Yong Fang’s research

Our vectorization technique has produced better precision compared to Yong Fang’s research where both have used LSTM model for classification.

4.3.3 Multiclass Classification

If we compare our results with S.Krishnaveni’s[11] research with a much smaller data set.

Classifier	S.Krishnaveni’s Result	Our Result
Decision Tree	100	98.80
Multilayer Perceptron	96.20	99.54

Table 4.10: Comparing with S.Krishnaveni’s research

4.3.4 XSSClassifier

If we compare our results with Shailendra Rathore's[12] research.

Classifier	Shailendra Rathore's Result	Our Result
Random Forest	97.2	69.52
AdaBoost	96.6	98.51
Support Vector Machine	95.8	99.59
Logistic Regression	95.6	99.36
k-Nearest Neighbors	95.2	93.44

Table 4.11: Comparing with Shailendra Rathore's research

We are able to produce better overall accuracy of 99.59% against to 97.2% using linear SVM.

4.4 Context-Aware Honeypot

We have trained our Support Vector Machine model with the cheat sheet and tested on the previous known attacks. Since we want to extract good payloads so we start blocking easily predictable attacks and constantly keep increasing that threshold. These are our results Table 4.12

Similarity(%)	Bloacked Attacks
0	33425
80	23818
81	23808
82	23794
83	23778
84	23708
85	23642
86	23549
87	23453
88	22828
89	21469
90	21285
91	21188
92	21027
93	20724
94	20491
95	20248
96	20011
97	19839
98	19452
99	18823
1	0

Table 4.12: Attacks Blocked

In figure 4.4 we have plotted the graph from the data of table 4.12 which is depicting that at honeypot how gradually we can increase the complexity level. Figure 4.5 is an instance that at 80 percent similarity out of all the known XSS attacks 28.7 percent can still be executed because they have better complex payloads.

Context-Aware Honeypot

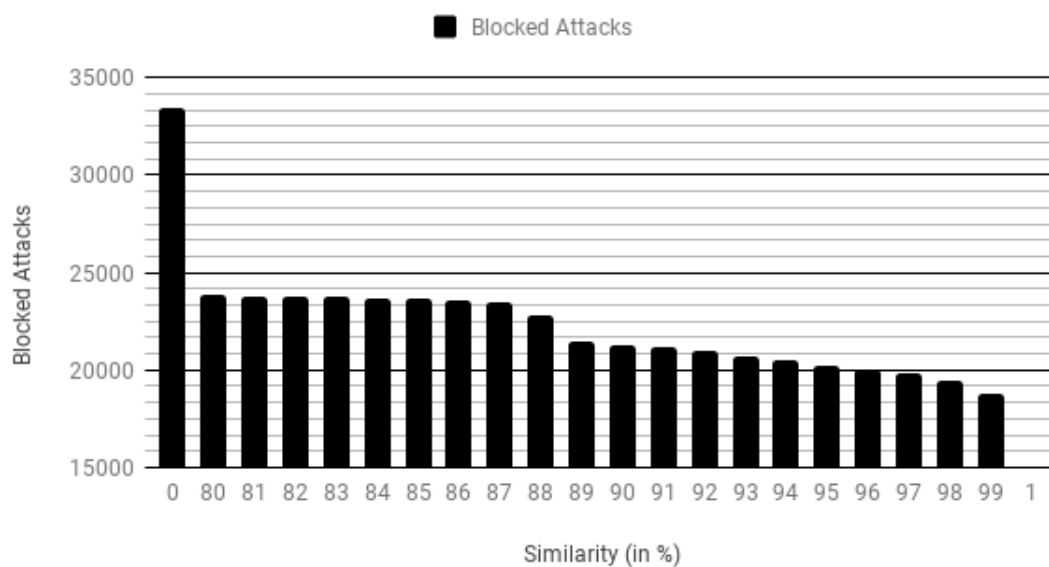


Figure 4.4: Attacks Blocked

At 80% similarity in ContextAware Honeypot

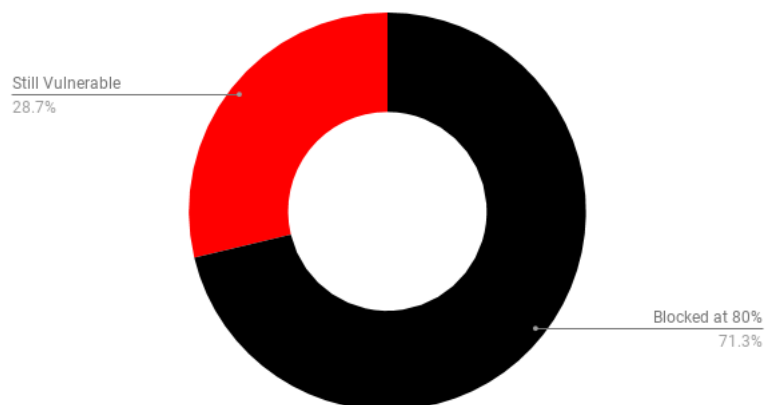


Figure 4.5: At 80% similarity in ContextAware Honeypot

4.5 Summary

We can summarize this section by saying that after all these experiments we can finally say that Linear SVM along with Count Occurrence is producing the best accuracy and LSTM is producing best precision. So HIDPS is using LSTM classifier which has a precision of 99.95% and accuracy of 99.53% and on the other hand context switcher will use Linear SVM classifier because of the accuracy of 99.59%.

Chapter 5

Related work

Over the course of time, security researchers have proposed the theory of context generation based on the attacks and several of them have linked Intrusion detection system with Artificial Intelligence. But it was all started by I. Kuwatly's research on a dynamic honeypot design for intrusion detection where he discussed a honeypot capable enough to adapt in a dynamic network environment which is changing constantly. He used a combination of virtual honeypot and passive/active probing for deployment and configuration of honeypot [13]. Later came Xianjin Fangln's research on integrating IDS/IPS with AI, so that it can detect anomalies by adapting according to networks. Here he used the artificial neural network (ANN) in the IDS to learn about the network and reduce false positives. Then it would adapt to the new environment and hence can identify new attacks [14]. Then these techniques were integrated into honeypot by Wira Zanoramy Ansiry Zakaria's research by reviewing techniques like Expert System (ES) and Case-Based Reasoning (CBR) so that intelligent honeypots can be developed [15].

Daniel Fraunholz's research on a dynamic honeypot which can configure, deploy and maintain itself by using machine learning techniques like k-means algorithm. This brought the idea of identifying machines and devices in a network which are then analyzed and clustered. According to these clusters, honeypots are placed in the network. And here only he discussed the idea of context-aware features in these honeypots [16].

There was also a good amount of research that took place in detecting XSS Attack by using ML. It all started with the detection of SQL attacks back in 2005 by Fredrik Valeur [17]. Over time the research has picked up the pace. Recent research done by Fawaz A. Mereani brought an accuracy of 99.75% using k-NN for classification [9] and Yong Fang's research has a precision of 99.5% using LSTM for classification [10].

Chapter 6

Conclusions

There are still 50 false positive which are still undetected, our aim is to reduce this number. You will find our research project and tests at

<https://github.com/Shubham295/Honeypot-HIDPS.git>

6.1 What can be done next?

We hope that the network security researchers will find our current research project useful and would like to contribute to it for further learning and development. Currently we are dependent on logs for detecting, use of softwares like wireshark will not even remove that dependency but also makes it fast and real-time. It is only developed for Cross-site scripting (XSS) attack, there are plenty more attacks where we can develop it further like

1. SQLI attack
2. MitM attack
3. CSRF attack
4. Directory Traversal attack
5. DDoS & DoS attack

6. Birthday attack
7. Spear Phishing & phishing attack
8. Drive-by attack
9. Malware attack
10. Eavesdropping attack
11. Password attack

References

- [1] Fahmida Y. Rashid. *Sony Networks Lacked Firewall, Ran Obsolete Software: Testimony*. URL: <https://www.eweek.com/security/sony-networks-lacked-firewall-ran-obsolete-software-testimony>.
- [2] Daan Pepijn. *5 of the worst security missteps by major tech companies in 2018*. URL: <https://thenextweb.com/contributors/2019/01/16/5-of-the-worst-security-missteps-by-major-tech-companies-in-2018/>.
- [3] Tech2 staff. *Aadhaar security breaches: Here are the major untoward incidents that have happened with Aadhaar and what was actually affected*. URL: <https://www.firstpost.com/tech/news-analysis/aadhaar-security-breaches-here-are-the-major-untoward-incidents-that-have-happened-with-aadhaar-and-what-was-actually-affected-4300349.html>.
- [4] Gopal Sathe Rachna Khaira Aman Sethi. *UIDAI's Aadhaar Software Hacked, ID Database Compromised, Experts Confirm*. URL: https://www.huffingtonpost.in/2018/09/11/uidai-s-aadhaar-software-hacked-id-database-compromised-experts-confirm_a_23522472/.
- [5] Paul Mutton. *Fraudsters modify eBay listings with JavaScript redirects and proxies*. URL: <https://news.netcraft.com/archives/2014/04/28/fraudsters-modify-ebay-listings-with-javascript-redirects-and-proxies.html>.
- [6] Paul Mutton. *Hackers still exploiting eBay's stored XSS vulnerabilities in 2017*. URL: <https://news.netcraft.com/archives/2017/02/17/hackers-still-exploiting-ebays-stored-xss-vulnerabilities-in-2017.html>.
- [7] L. SPITZNER. "Honeypots: Tracking Hackers. 1 edition". In: Addison-Wesley Professional, 2002. Chap. 1.
- [8] Edward Ma. *3 basic approaches in Bag of Words which are better than Word Embeddings*. URL: <https://towardsdatascience.com/3-basic-approaches-in-bag-of-words-which-are-better-than-word-embeddings-c2cbc7398016>.
- [9] Fawaz A. Mereani; Jacob M. Howe. "Detecting Cross-Site Scripting Attacks Using Machine Learning". In: *The International Conference on Advanced Machine Learning Technologies and Applications* 723 (2018). DOI: https://doi.org/10.1007/978-3-319-74690-6_20.
- [10] Yong Fang; Yang Li; Liang Liu; Cheng Huang. "DeepXSS: Cross Site Scripting Detection Based on Deep Learning". In: *International Conference on Computing and Artificial Intelligence* (2018), pp. 47–51. DOI: <https://dx.doi.org/10.1145/3194452.3194469>.

- [11] K.Sathiyakumari S.Krishnaveni. “Multiclass Classification of XSS Web Page Attack using Machine Learning Techniques”. In: *International Journal of Computer Applications* 74.12 (2013), pp. 36–40. DOI: [dx.doi.org/10.5120/12940-0033](https://doi.org/10.5120/12940-0033).
- [12] Jong Hyuk Park Shailendra Rathore Pradip Kumar Sharma. “XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs”. In: *Journal of Information Processing Systems* 13.4 (2017), pp. 1014–1028. DOI: <https://doi.org/10.3745/JIPS.03.0079>.
- [13] I. Kuwatly ; M. Sraj ; Z. Al Masri ; H. Artail. “A dynamic honeypot design for intrusion detection”. In: *The IEEE/ACS International Conference on Pervasive Services* (2004), pp. 95–104. DOI: <https://doi.org/10.1109/PERSER.2004.1356776>.
- [14] Xianjin Fang ; Lingbing Liu. “Integrating Artificial Intelligence into Snort IDS”. In: *3rd International Workshop on Intelligent Systems and Applications* (2011), pp. 1–4. DOI: <https://doi.org/10.1109/ISA.2011.5873435>.
- [15] Wira Zanolamy Ansiry Zakaria ; Miss Laiha Mat Kiah. “A review on artificial intelligence techniques for developing intelligent honeypot”. In: *International Conference on Computing Technology and Information Management* 2 (2012), pp. 696–701.
- [16] Daniel Fraunholz ; Marc Zimmermann ; Hans D. Schotten. “An adaptive honeypot configuration, deployment and maintenance strategy”. In: *19th International Conference on Advanced Communication Technology* (2017), pp. 53–57. DOI: <https://doi.org/10.23919/ICACT.2017.7890056>.
- [17] Fredrik Valeur; Darren Mutz; Giovanni Vigna. “A Learning-Based Approach to the Detection of SQL Attacks”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment* 3548 (2005). DOI: https://doi.org/10.1007/11506881_8.
- [18] Michele Adams. Mokube Iyatiti. “Honeypots: concepts, approaches, and challenges”. In: *Proceedings of the 45th Annual Southeast Regional Conference*, (2007).