LIDS: An Extended LSTM Based Web Intrusion Detection System With Active and Distributed Learning

Arul Thileeban Sagayam

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Godmar V. Back, Chair

Randolph C. Marchany

David R. Raymond

Kurt Luther

May 3, 2021

Blacksburg, Virginia

Keywords: Intrusion Detection System, Machine learning, Phishing detection, Active learning

LIDS: An Extended LSTM Based Web Intrusion Detection System With Active and Distributed Learning

Arul Thileeban Sagayam

(ABSTRACT)

Intrusion detection systems are an integral part of web application security. As Internet use continues to increase, the demand for fast, accurate intrusion detection systems has grown. Various IDSs like Snort, Zeek, Solarwinds SEM, and Sleuth9, detect malicious intent based on existing patterns of attack. While these systems are widely deployed, there are limitations with their approach, and anomaly-based IDSs that classify baseline behavior and trigger on deviations were developed to address their shortcomings. Existing anomaly-based IDSs have limitations that are typical of any machine learning system, including high false-positive rates, a lack of clear infrastructure for deployment, the requirement for data to be centralized, and an inability to add modules tailored to specific organizational threats. To address these shortcomings, our work proposes a system that is distributed in nature, can actively learn and uses experts to improve accuracy. Our results indicate that the integrated system can operate independently as a holistic system while maintaining an accuracy of 99.03%, a false positive rate of 0.5%, and speed of processing 160,000 packets per second for an average system.

LIDS: An Extended LSTM Based Web Intrusion Detection System With Active and Distributed Learning

Arul Thileeban Sagayam

(GENERAL AUDIENCE ABSTRACT)

Intrusion detection systems are an integral part of web application security. The task of an intrusion detection system is to identify attacks on web applications. As Internet use continues to increase, the demand for fast, accurate intrusion detection systems has grown. Various IDSs like Snort, Zeek, Solarwinds SEM, and Sleuth9, detect malicious intent based on existing attack patterns. While these systems are widely deployed, there are limitations with their approach, and anomaly-based IDSs that learn a system's baseline behavior and trigger on deviations were developed to address their shortcomings. Existing anomaly-based IDSs have limitations that are typical of any machine learning system, including high false-positive rates, a lack of clear infrastructure for deployment, the requirement for data to be centralized, and an inability to add modules tailored to specific organizational threats. To address these shortcomings, our work proposes a system that is distributed in nature, can actively learn and uses experts to improve accuracy. Our results indicate that the integrated system can operate independently as a holistic system while maintaining an accuracy of 99.03%, a false positive rate of 0.5%, and speed of processing 160,000 packets per second for an average system.

Acknowledgments

I would like to express my gratitude and appreciation to everyone that helped me during the course of my research work, guiding me to completion.

Dr. Godmar Back for his inputs and feedback to improve my work and providing me with resources that were necessary.

Dr. David Raymond for his constant guidance throughout the course of this work. The guidance and inputs he has provided throughout this work have really helped polish it into its final form.

Prof. Randy Marchany for his valuable inputs in the security domain and all the assistance I needed to drive this work to completion.

Dr. Kurt Luther for his guidance on the expert system, helping bring the module to its final form.

I would also like to thank members of the IT Security Lab that I have engaged with. Bouncing ideas off of each other and getting inputs on domains I'm not familiar with really helped put things into perspective.

Finally, I would like to thank my family for their unwavering support throughout my degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The growth in popularity of the Internet among the general public, and the proliferation of state-sponsored attacks on cyberinfrastructure, have increased the need for emphasis on network security in enterprises. Cyber-attacks have risen consistently over the last decade. In 2021, it was predicted [1] that a cyber attack would occur every 11 seconds, which is four times the rate it was 5 years ago. Particularly, web applications are a rich target for attackers. According to Rapid7, 90% of web applications are vulnerable to an already known CVE (Common Vulnerabilities and Exposures). Rapid7 is a cybersecurity consulting company that has captured web application data deployed on AWS and Azure in Q2 of 2018 for this analysis. A summary from the report in Table 1.1 showcases how the most common vulnerabilities from the OWASP Top Ten Web Application Security Risks, which includes common risks such as SQL Injection and Cross-Site Scripting, are responsible for the majority of the incidents.

Detection of vulnerabilities and intrusions plays a vital role in network defense. Ideally, we would like to proactively identify and correct all system vulnerabilities. Unfortunately, that is not feasible and hence solutions are created for both identifying vulnerabilities, and also detecting intrusions if an attacker tries to exploit an existing vulnerability. Intrusion detection systems are a vital component in securing systems

In this work, an intrusion detection model is introduced which has been constructed with the aid of machine learning techniques.

Top 5 Most Common Incidents	1. Cross-Site Scripting(XSS) 2. SQL Injection 3. Automated Threats 4. File Path Traversal 5. Command Injection
Average Time to Patch a Vulnerability	38 days
Oldest Unpatched CVE	340 days
Percentage of Apps with Known CVEs	90% of active applications
Routes/API Endpoints Exposed per App	2900 orphaned routes exposed

Table 1.1: Summary of attack findings

1.1 Motivation

Traditional intrusion detection systems like Snort[2] and Zeek[3] utilize a signature-based detection system where signatures of existing malicious activities are captured and new malicious activity is detected based on them. Although these systems are widely used, there is a need to look beyond these methods due to their inability to capture variants of the existing signatures. An alternate methodology is an anomaly-based detection, where a model is trained on normal traffic and any deviation from that is considered malicious, is offered as a better solution. The majority of these solutions are based on machine learning, but a lot of them have consistently used old datasets like KDD99[4], a dataset that is compromised of Transmission Control Protocol(TCP) packet features and that is more than two decades old. Also, there is a lack of emphasis on web application-based attacks and a lack of emphasis on resolving issues that prohibit real-time usage of these solutions. Work such as Siaterlis et al. [5] doesn't detect web application attack vectors. Other works by Javaid et al. [6] and Mohammadpour et al. [7] use the KDD dataset for training and only evaluate network attacks like Probing, User to Root(U2R), Denial of Service, and Remote to User(R2L) Attack. There is a need for a web-based intrusion detection system trained

with updated datasets that deal with the issues that block real-time usage as well.

1.2 Research Objective

The objective of this work is to develop an anomaly based intrusion detection system that does the following.

- Targets web application based attacks.
- Utilizes updated datasets to account for evolving cybersecurity attacks.
- Is a holistic system that can be deployed .
- Tackles problems like prediction speed and false positives.
- Is modularized for future updates .

Previous efforts by Pastrana et al. [8], Vartouni et al. [9], and Nguyen et al. [10] have incorporated modern, web application based datasets like CSIC 2010 [11] and some works on old datasets have included components like active learning for reduction of false positive rates [12]. No single work has combined all these objectives to get a holistic intrusion detection system for web application attacks.

1.3 Core Contribution

To achieve our research objective, a holistic intrusion detection system with extended features is designed and studied. As part of the whole system, there are two novel models introduced:

- LIDS - A novel Long Short Term Memory (LSTM) based intrusion detection system (IDS) which has been evaluated against current state-of-the-art machine learning-based intrusion detection systems.

- PhishCCA - A novel machine learning based phishing detection solution which utilizes TLS certificates as the data source for classification.

These models are combined to form a holistic, distributed, and modularized intrusion detection system. The integrated system utilizes a human in the loop system (HITL) where cybersecurity experts can weigh in on borderline decisions made by the LIDS. It is also deployed in a distributed model that places the system in local machines instead of utilizing a single, centralized machine.

1.4 Thesis Organization

The thesis is organized as follows:

- Chapter 2: Background - Chapter 2 introduces the background information necessary to understand the problem statement and the solution. The information includes an overview of Networks, Web Application Attacks, Phishing, Machine learning, and the dataset used in this work.

- Chapter 3: Design - Chapter 3 provides a deep explanation on the architecture and design of our solution.

- Chapter 4: Implementation - Chapter 4 showcases the actual implementation of the system designed.

- Chapter 5: Evaluation - Chapter 5 evaluates the results of the work and discusses the advantages and disadvantages of this solution.
- Chapter 6: Related Work - Chapter 6 discusses the research done in this domain targeting specific problems from this work.
- Chapter 7: Conclusion - Chapter 7 answers the challenge posed in Chapter 1 about how the solution tackles the problem statement and discusses future work.

Chapter 2

Background

This chapter provides the necessary background information to understand the constructs around this problem statement and the solution. Initially, there is a discussion on networks and network protocols to understand the context in which our solution is applied. This is followed by a discussion on web application attacks that is critical to understanding the attack vectors. A discussion on machine learning models and the dataset we used in our work follows.These are critical to our system design.

2.1 Networks

Computer networking architecture has been built with an emphasis on scalability and modularization. This has led to the rapid growth of the Internet. Most early network protocols weren't focused on security [13] resulting in vulnerabilities that are still impacting modern-day networks. In this section, we discuss one of the key abstract models of networks—the OSI model—and some discussion of widely-used protocols like HTTP and HTTPS and wraps up by discussing Packet Captures. Our intrusion detection system design examines network protocol and payload data to help detect attempted network intrusions.

2.1.1 OSI Model

Open Systems Interconnection (OSI) Model provides a conceptual framework for Internet communication. As described in Table 2.1, there are 7 abstract layers and there are different protocols and functions associated with each layer. Although the modern Internet system doesn't completely follow the OSI model, it is useful to understand Internet communications.

Number	Layer	Protocols	Function
7	Application	HTTP, TELNET	Human-Computer Interaction
6	Presentation	TLS, SMB	Data preparation
5	Session	Sockets,WinSock	Maintains connection
4	Transport	TCP, UDP	End-to-End Connection
3	Network	IP, ICMP	Packet transmission
2	Data Link	Ethernet, PPP	Format definition
1	Physical	-	Raw bit transmission

Table 2.1: Layers and protocols of OSI Model

Internet communication is done through the Network layer using the IP (Internet Protocol) that is further utilized by TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) for transporting content through sockets at the session layer. While these protocols and layers are key to the transportation of data across devices, the data link and physical layer help break this data down into appropriate frames and pass it through defined hardware. Our emphasis in this work is focused on the top two layers—Application and Presentation—over which many protocols like HTTPS, TLS that support web applications operate.

2.1.2 HTTP

Hypertext transfer protocol (HTTP) as described in RFC2616 [14] as "an application-level protocol for distributed, collaborative, hypermedia information systems". It is a client-server based protocol that is used at the application layer to exchange resources on the Web. Servers are used to host web applications through HTML documents and other static resources like images. The HTTP protocol is used by these servers to transmit their content to the clients which result in hosting their website on the Internet for open public access. Typically, the web browser acts as the client and constructs the request based on user input, and sends it to the server. Further, it also reconstructs the response from the server into an appropriate HTML document/web page. HTTP is also stateless, simple, and is extensible. As showcased in Fig. 2.1, all HTTP packets consist of a header section and a content section. The header of an HTTP request contains a variety of fields. Some of the most frequently used fields in either the HTTP request or HTTP response are:

- Host: Used to specify the host and port of the resource requested.
- User-Agent: Used to specify the user-agent through which the user is making the request.
- Accept: Used to specify the type of content that can be accepted.
- Accept-Charset: Used to specify the type of charset (Eg: Unicode) that can be accepted.
- Keep-Alive: Response to how long the connection can be kept alive.
- Allow: Used to specify allowed request methods.
- Connection: Used to specify what can be done with the current connection.

- Cookie: Used to transmit cookies
- Pragma: Used to specify implementation-specific directives (Eg: no-cache).
- Cache-Control: Used to specify instructions that should be used by the caching system.
- Content-Length: Used to mention the response content's length.

HTTP defines a set of request methods that help the server identify the operation to be performed. Important request methods include GET and POST. While POST is used to send data to the server, GET is used to request data from the server. The parameters in these requests help understand the communication sent to the server. To summarize, HTTP is the standard protocol that operates the majority of the Web and the data from the HTTP packets are typically very useful in determining any attack.

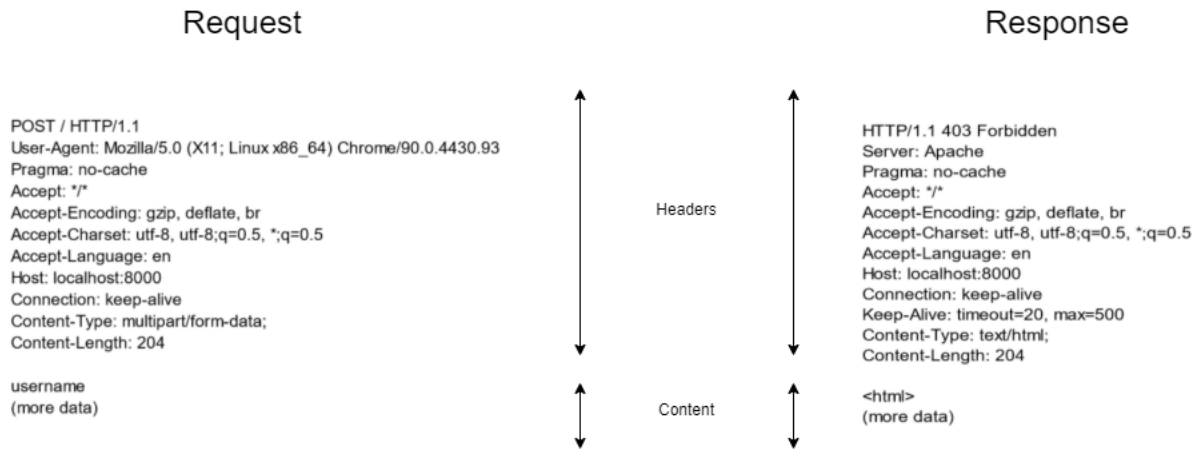


Figure 2.1: Example of a HTTP request/response packet

2.1.3 HTTPS

HTTP being extensible allowed for a variety of new protocols to be formed over it. One important one is hypertext transfer protocol secure (HTTPS), which adds security features

to the native is HTTP protocol. HTTPS guarantees confidentiality, integrity, and authenticity to the data transmitted over the HTTP protocol and has been widely adopted across the industry. HTTPS was built on Secure socket layer (SSL), a protocol that is based on public-key encryption. Over the years, a more secure version called Transport layer security (TLS) has been integrated instead of SSL. The security of the whole process relies on the TLS protocol that is extended over HTTP. Each website or server is issued a digital certificate by third-party vendors called Certificate authorities (CAs) who validate the authenticity of the website first. This certificate is signed with the private key of the certificate authority. Certificate authorities are entities that can issue digital certificates due to their credibility. They are very few, mostly composed of multi-national companies, have to follow a strict set of technical requirements, and undergo security audits frequently to maintain their credibility. As displayed in Fig. 2.2, whenever a client tries to establish a connection with the server, this certificate is shared with the client. The client has the public keys of all major certificate authorities and uses those to verify the certificate, and once verified, cipher specifications are set and shared between the client and the server. Based on these encryption specifications set, the data is appropriately encrypted and transmitted between the client and the server. Digital certificates are important to this process. They help web clients ensure the authenticity of the website. After validation of the certificate, the browsers typically displays a lock sign if a website uses HTTPS and has a valid digital certificate. This can be used by users to identify phishing/malicious sites.

Digital certificates

A digital certificate contains a server public key that is used to encrypt data between a web client and server. The digital certificate also contains a digital signature that is used to validate authenticity. The certificate is signed using the host's private key and is validated

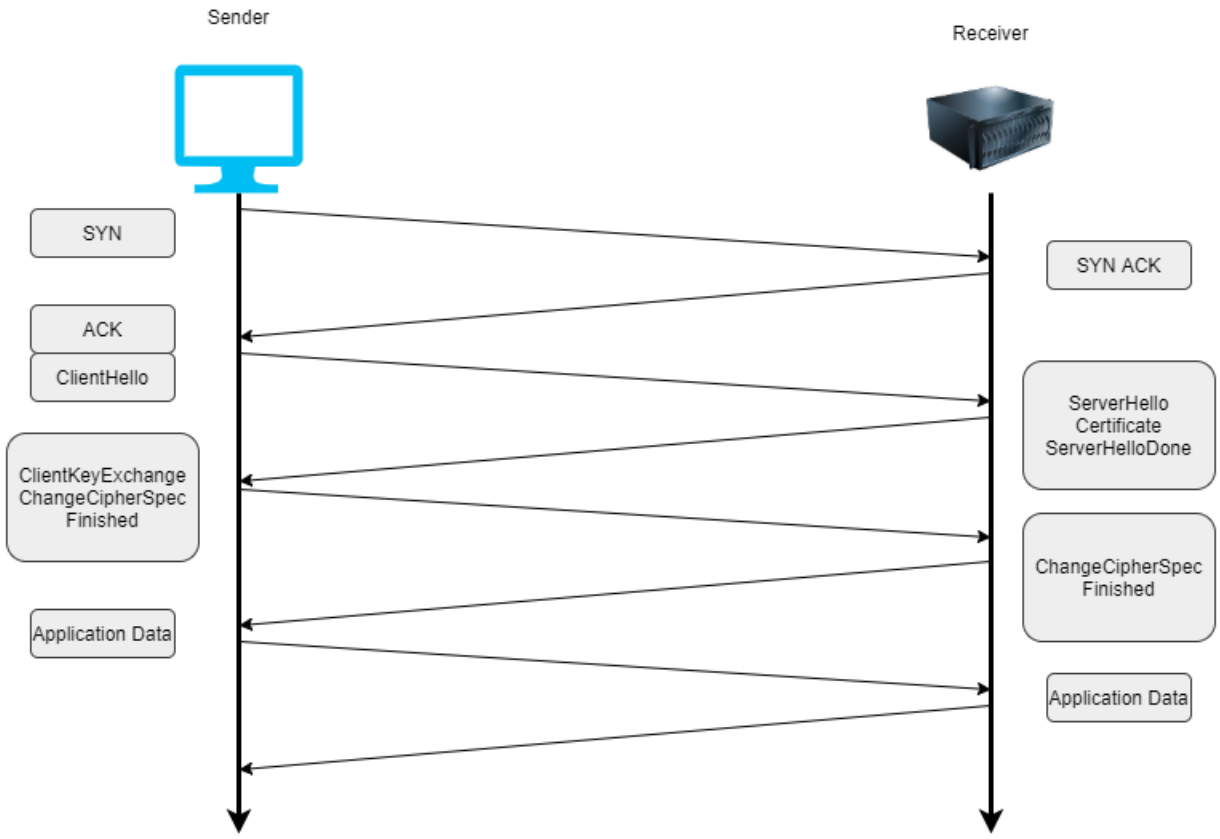


Figure 2.2: Example of a HTTPS request/response process

by the certificate authority.

The standard certificate format used for TLS is X.509 [15], shown in Fig. 2.3. There are three versions, with v3 being the latest. One important component of the digital certificate infrastructure is the ability to revoke certificates that become invalidated before they expire, perhaps because they have been compromised. X.509 version 3 allows for extensions that have made multiple Certificate revocation mechanisms feasible. TLS Certificates are sometimes integral to identify phishing websites directly from the browser.

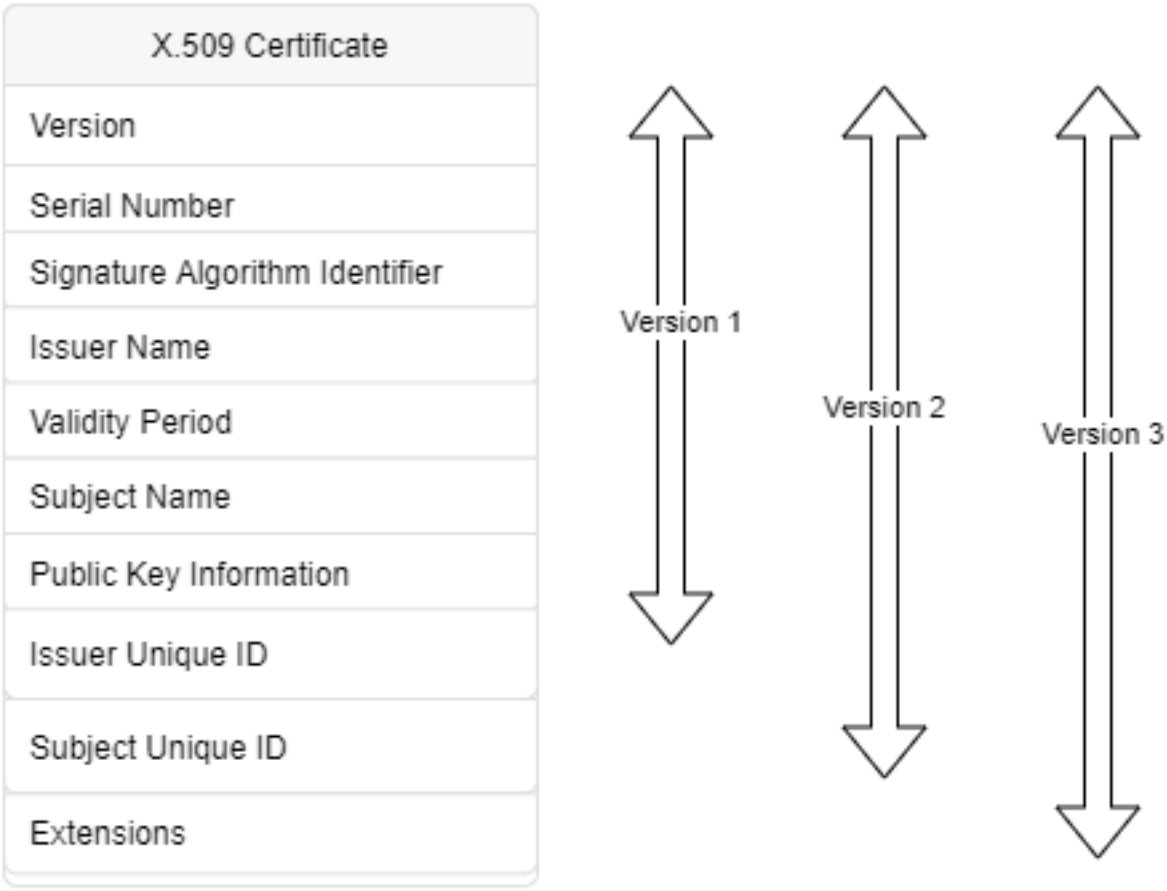


Figure 2.3: Format of the X.509 standard used for defining public key certificates, particularly TLS/SSL

2.1.4 Network Traffic Analysis

Packet traffic analysis is a process by which packets passing through a hardware interface are captured using a network tap or switched port analyzer (SPAN) port. Intrusion detection systems generally utilize promiscuous mode [16] on a network interface to monitor network traffic. Analyzing the traffic allows them to identify intrusions.

2.2 Web Application Attacks

Web applications are the most deployed [17] and used services in the software domain. Due to their widespread nature, they are highly targeted by attackers attempting to gain access to systems or data for malicious purposes. The number of web application attacks has increased drastically in recent years. In 2020, 4.2 billion web attacks were blocked, which is eight times the number in 2019 [18]. With the number of websites ever-increasing, this number is expected to continue to rise at a similar rate. Looking into these attacks, one can realize that a huge number of attacks are focused on denying important services, stealing credentials, or using the web service as a gateway to more secure systems. The reason behind these attacks showcases why security in this area is of utmost importance. A select set of web application attacks are discussed since they are related to this work and the work in this area. Zero-day exploits and DDoS are discussed particularly to support related work. The dataset utilized in this work focuses on web application attacks and the OWASP Top 10 list plays a major role in that. Understanding the below attacks would give an overview of how these attacks would act as anomalies in a normal system.

2.2.1 OWASP Top 10

The OWASP Top 10 Web Application Security Risks [19] is a widely recognized list of vulnerabilities in the domain of web application security. Updated once every few years, this list provides experts with information on the top 10 most widely observed attack vectors on web applications. The list has been quite consistent for several years, which shows that even though people know the attack methodologies that they should avoid, they are still very common and leave many web applications vulnerable. The latest version of the OWASP Top 10 was published in 2017 and consists of the following risks.

- Injection: All types of vulnerabilities that send untrusted data to an interpreter. For example: SQL Injection.
- Broken Authentication: Any flaw in the authentication mechanism/session management which potentially leads to user account compromise.
- Sensitive Data Exposure: Application not doing enough to protect sensitive data stored in the web applications.
- XML External entities: Improper parsing of XML input that contains external entities that could lead to unauthorized connection with them breaching confidentiality.
- Broken Access Control: Broken access control allows attackers to bypass authorization and perform operations as privileged users.
- Security Misconfiguration: Components being vulnerable due to usage of insecure or poorly created configurations.
- Cross-Site Scripting: Enables attackers to inject client-side scripts into the web page which is further used to compromise users.
- Insecure Deserialization: Injection of a payload by inserting a serialized object which is not sanitized properly when deserialization is done.
- Using components with known vulnerabilities: Inclusion of components with reported vulnerabilities.
- Insufficient logging and monitoring: Logging and monitoring is an essential security practice required to both prevent and audit any attack.

While some security risks listed above are direct vulnerabilities, some are an extension of a vulnerability and some are just security risks and don't pose a direct threat. Security

experts have always held the notion that fixing these security risks would protect most of the modern-day web.

2.2.2 Zero Day Vulnerability

Zero-day vulnerabilities are vulnerabilities for which the vendor has had only ”zero days” to fix or are newly discovered vulnerabilities for which patches and signatures don’t exist. Some vulnerabilities are even formed by poor parsing, leading to non-malign web requests disturbing the system integrity. Zero-day exploits are attack vectors that are used against these vulnerabilities to exploit the system.

2.2.3 Denial of Service Attack

Denial of Service is an attack by which a resource, typically a website is made inaccessible to the users. Although there are various ways to go about this, the most commonly used methodology [20] is to flood the service with thousands of HTTP requests, often requests with a maximum size payload and junk data. In the modern-day, a single computer cannot achieve a large enough request load to overload a system, and hence multiple systems are used for this purpose, which is called Distributed Denial of Service (DDoS).

2.3 Phishing

Phishing is a cyber-attack that is primarily delivered in form of an email. Although there are different variants to this process, the usual goal is to gain credentials or other sensitive information about the target. One common scenario uses a form of social engineering by sending a mail that entices the victim into entering credentials in a fake site set up by the

attackers that are made to look like a trusted website's login page. This form of attack is very common and multiple variants focus on researched individuals like spear-phishing, targeting low profile individuals in an organization, and whaling, targeting high profile executives in an organization.

2.4 Machine Learning

Machine learning is an interdisciplinary field in which we study various algorithms that automatically improve by themselves through a learning process. Since the 2000s, this field has grown due to the advent of high-performance computing, which is needed for many of these algorithms to work. Machine learning has been applied to various domains—healthcare, marketing, robotics, manufacturing, and other important fields to supplement and even replace human workers.

Likewise, machine learning is also widely used in cybersecurity. Due to the underlying nature of multiple machine learning algorithms to detect outliers or anomalies, it is highly useful in this domain which is focused on detecting anomalies. In our work, machine learning is used to detect anomalies that represent potential network intrusions.

2.4.1 Data Preprocessing

Preprocessing is critical before most machine learning models can be applied to datasets. Data has various forms of representation, such as numbers, text, and images. These representations should be converted into a generic form that is understandable by machine learning algorithms — vectors. The process of converting data to this form is called Encoding and the two types of encoding used in this work are one-hot encoding and TF-IDF

(Term Frequency-Inverse Document Frequency), both described in the following sections.

One-hot encoding

One-hot encoding is a categorical encoding technique where a column is constructed for each feature as shown in Fig. 2.4. In our work, a character-level one-hot encoding is used. A X x Y matrix is constructed where X is the number of characters used in the dataset. Y is the number of characters in a given sequence. In Fig. 2.4, the first column shows the characters in the dataset and the first row shows the sequence. The final matrix is depicted between.

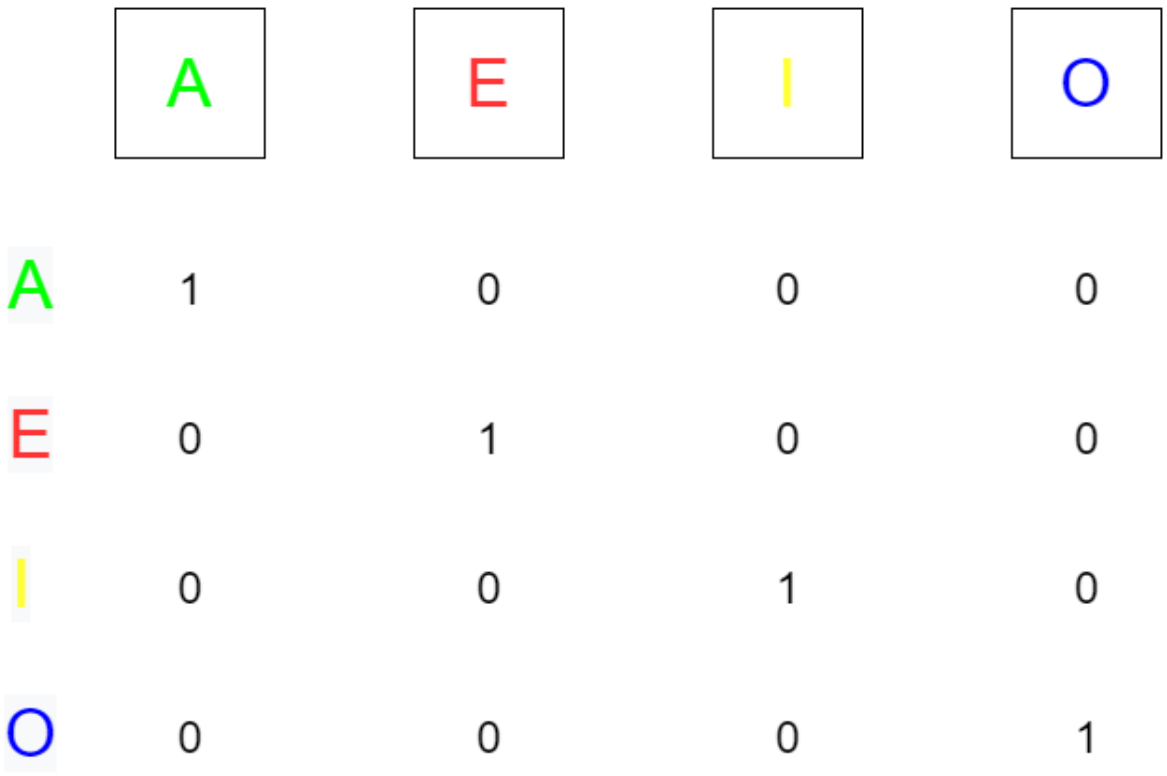
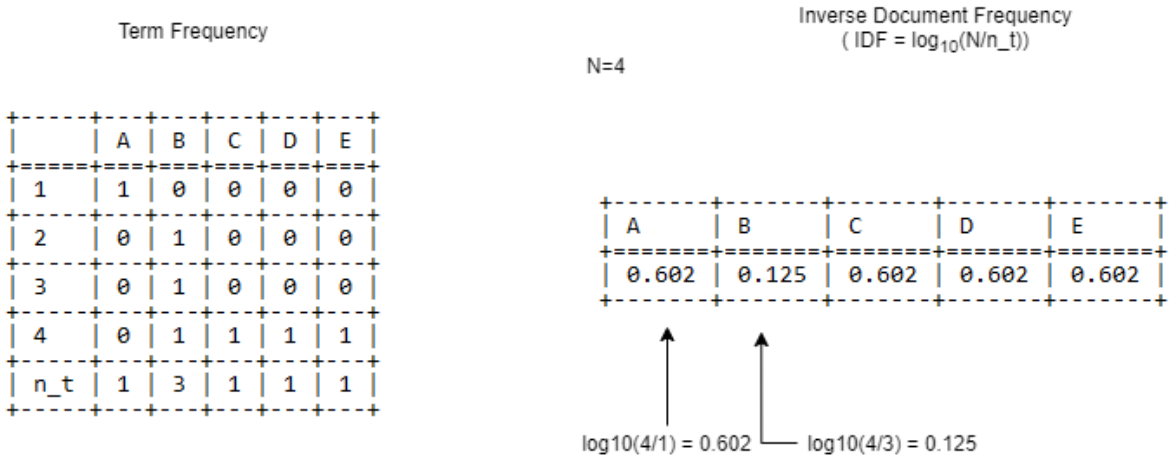


Figure 2.4: Character one-hot encoding - Example

TF-IDF

TF-IDF is a vectorization technique in which the vectors are created from the frequency of the term in a sequence instead of the individual parts of the sequence itself. Due to this mechanism, it is also referred to as an information retrieval technique. The TF-IDF algorithm takes the term's frequency (TF) and inverse document frequency (IDF) as showcased in Fig. 2.5 and multiplies it to obtain the weight for the term which is then used in the vector. Term frequency is the total number of occurrences of the term in the whole document. Inverse document frequency is obtained for each term by calculating $log_{10}(N/n_t)$, where N is the total number of terms in the document and n_t is the term frequency of the corresponding term. This vectorization technique focuses on the importance of the term in the document rather then the term itself.



	A	B	C	D	E
1	1	0	0	0	0
2	0	1	0	0	0
3	0	1	0	0	0
4	0	1	1	1	1
n_t	1	3	1	1	1

A	B	C	D	E
0.602	0.125	0.602	0.602	0.602

Figure 2.5: TF-IDF vectorization technique - Example

2.4.2 Classifiers

Classification is a supervised learning methodology that is primarily used to classify data points into well-defined classes. This is done by primarily understanding the boundaries of

each class for the data points to be fit into each pre-defined class. The training process of the classifier helps understand the boundaries and the testing process can be used to detect if the classifier is working well. Classification can be categorized into three types based on the problem statement.

- Binary classification: A classification problem with only two output classes, such as *true* and *false*.
- Multi-class classification: A classification problem with multiple output classes.
- Multi-label classification: A classification problem where one input can be categorized into a combination of classes.

As discussed in [21], classifiers can also be categorized based on the algorithm used. A list of widely-used classifiers is provided below.

- Linear classifiers: Classification algorithms that classify data into labels based on a linear combination of input features.
- Support vector machines: Algorithm which finds a hyperplane (subspace whose dimension is one less than the actual dimension. For example: A N-1 hyperplane represents an N dimension) in an N-dimensional space to classify data points.
- Quadratic classifiers: Instead of using a linear decision boundary, quadratic classifiers construct a quadratic decision surface to classify data points.
- Kernel estimation: Statistical method which is used to estimate the probability density function of a random variable. It is utilized as a supporting property with other classifiers.

- Decision trees: A if-else tree-based structure that is trained over data to make classification decisions based on nodes of the decision tree. Each node has an if-else condition.
- Neural networks: Neural networks are network-based structures that are loosely based on human neurons. They utilize a deep learning process to learn from data points and classify them accordingly.
- Learning vector quantization: Artificial neural network that is based on distance-based classification.

There is no superior algorithm out of this list for wide usage. Each algorithm is useful based on the type of problem statement and the data used for solving it.

2.4.3 Outlier Detection

Outlier detection is a problem well-studied in the machine learning domain. Outliers are data points that don't conform to the normal distribution of the given dataset. A few of the automatic outlier detection algorithms are listed below.

- Isolation forest: Unsupervised tree-based anomaly detection algorithm that is based on the principles that anomalies are the minority and their attribute values are different from normal instances.
- Dbscan: Clustering-based anomaly detection method in which density is used as a factor to detect anomalies. Anomalies lie in less dense zones on the feature space and can thus be identified with ease.
- Minimum covariance determinant: Utilizing a statistical method to form a hypersphere (ellipsoid) that will engulf all normal data. Anything lying outside the structure is considered an anomaly. Can only be used if the inputs follow a gaussian distribution.

- Local outlier factor: Simple algorithm that detects data points far away from others based on the feature space to mark them as anomalies.

Although these automatic outlier detection algorithms exist, various algorithms like clustering and classification can also be used to detect outliers by tuning the model to accommodate for outliers as another class.

2.4.4 Neural networks

Neural networks are a group of algorithms that are studied under the domain "deep learning" that is modeled loosely based on human neurons. Like neurons, it's a bunch of nodes interconnected with other nodes to form a network. A simplistic view of a function of a neuron/perceptron is depicted in Fig. 2.7. For example, a vector of $[x_1\ x_2\ ...x_n]$ is input into the neuron that already has weights of $[w_1\ w_2\ ...w_n]$. A matrix multiplication occurs between the weight matrix and the input vector with which the bias value b is added and this is passed to the activation function which then outputs the result. Bias value is a constant that is used to shift the activation function. Activation functions like *sigmoid* and *relu* (described below) help make a final output decision based on the value outputted from the node. This is the functioning of a simple neuron and can be expanded into multiple layers and with multiple neurons in a single layer. Typically, weights are initialized randomly and altered to get a better result using backpropagation. Backpropagation is a process in which the output of the final layer is compared to the training class value (y) and based on the difference, an error value is calculated and propagated in reverse back into the network. This is showcased in Fig. 2.6. Each node in the neural network self-corrects its weight based on the error value and the error value diminishes as you get far away from the output layer. Like all supervised algorithms, most neural networks have the training and a testing phase. In a training phase,

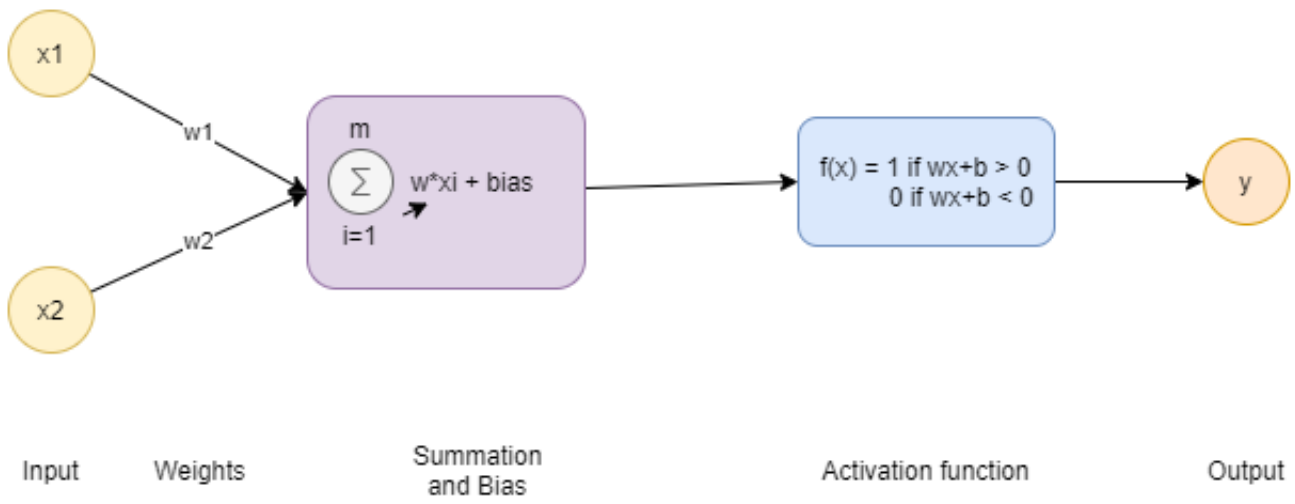


Figure 2.6: Workflow of a neuron in a neural network

the model has the weight matrix initialized randomly and predictions are made for each data point and based on the target value present, the error is calculated, backpropagated, and weights are re-aligned to fix the change and the focus of the neural network here is to minimize the loss function value and hence the errors. Once this is done, a good trained neural network would have a weight matrix that can make correct predictions and this is tested against a test dataset with similar distribution and through this, a neural network can be validated. There are various types of neural networks in the modern-day for processing different types of inputs and for different problems. Some examples include the following.

- Sequence processing: Gated Recurrent Unit (GRU), Long Short-term memory (LSTM), Recurrent Neural Network (RNN)

- Image processing: Convolutional Neural Network (CNN), Deep Neural Network (DNN)

- Representational learning: Autoencoders, Generative Adversarial Network (GAN)

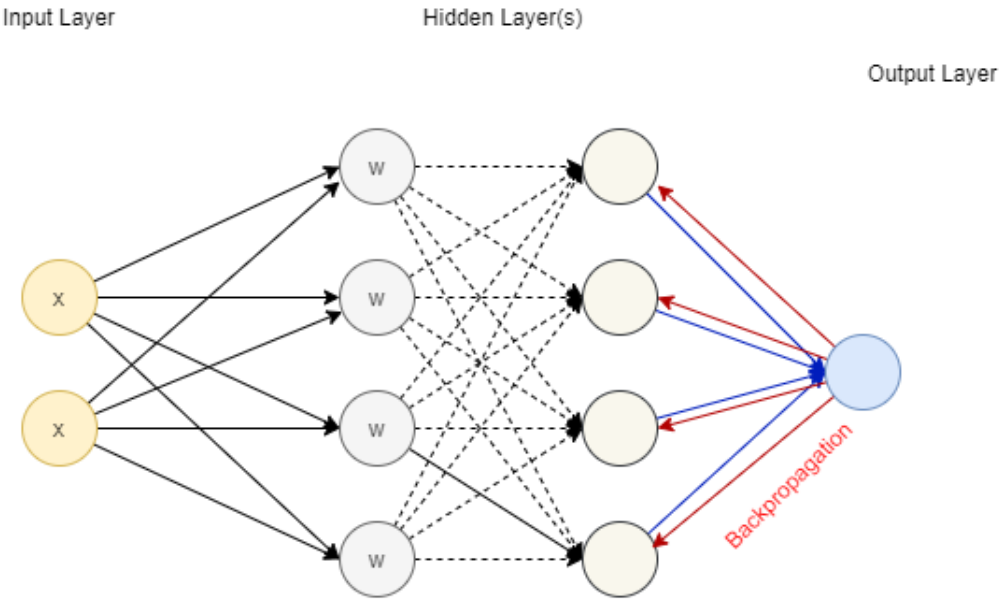


Figure 2.7: Working of a neural network with backpropagation

2.4.5 Parameters and layers

There are certain machine learning parameters and neural network layers used in this work and related work that are described below:

- Leaky ReLU: ReLU(Rectified Linear) is a piecewise linear activation function in which the output is the input if it is positive and zero if negative. This causes neurons to die in some cases in which case Leaky ReLU is used which changes the slope in the negative axis, causing a leak.

- Sigmoid: Sigmoid is an activation function described mathematically as $\frac{1}{1+e^{-x}}$ and is widely used due to its simplicity, but has a drawback of having a short range.

- Binary crossentropy: Binary crossentropy is a loss function that is used in binary classification tasks that uses log loss between the target value and the actual value to calculate the loss value.

- Adam optimizer: Adam optimizer is an extension to the gradient descent algorithm used to optimally update network weights based on training data.
- Dense layer: Dense layer is the most commonly used layer. It has no special properties and is a regular connected layer.
- Time Distributed layer: Time distributed layer is a dense layer with time as a feature. it is used while building sequential models to keep track of steps.

2.4.6 Select models

While there are an abundant number of models in machine learning, there are only a few pertaining to our problem statement which are studied here:

Recurrent Neural Network (RNN)

Traditional neural networks, also known as feed-forward neural networks pass only the output from the current layer to the next layer. With these networks, the historical data does not influence the current data since historical data don't influence the current weights. Recurrent neural networks(RNNs) weigh in here by creating a looping mechanism as depicted in Fig. 2.8. RNNs have a hidden state associated with them which is a representation of prior information and which is passed through loops from one step to the next. This helps process sequences since sequences need to have historical context associated with them to be understood right. Due to backpropagation in neural networks, RNNs have a vanishing gradient problem - gradient shrinks exponentially as it back propagates and this causes parts of sequences that aren't close to the current part to have very minimal contribution to the hidden state i.e. lack of long-term dependencies over multiple steps. To resolve this issue, new neural networks like LSTMs and GRUs have been proposed.

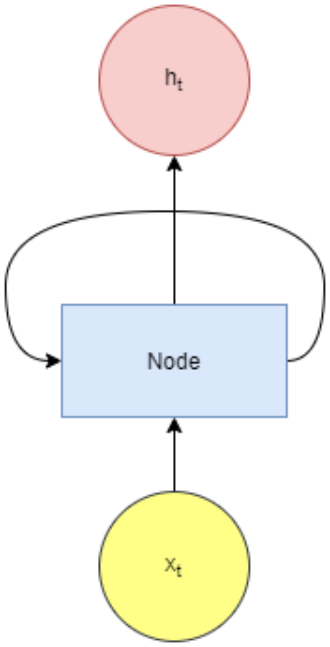


Figure 2.8: Single node of a Recurrent Neural Network

Long Short Term Memory (LSTM)

As discussed, LSTMs were created to solve long-term dependencies and thus have taken the Natural Language Processing (NLP) field by storm due to their ability to process long sentences as sequences. As displayed in Fig. 2.9, as opposed to only the 'tanh' function gate (used to train through gradient descent) in RNNs, LSTMs also have multiple gates — forget gate, input gate, and an output gate. In an LSTM network, the cell state is important since it is the information that travels from node to node and the gates interact with the cell state to decide what information to be passed along. Using the gates, the input, and the previous output, a single cell block decides how much of the previous cell state to retain in the current cell state and pass along and also to recalculate the hidden state. This is the underlying mechanism by which LSTM can process long sequences. In real time applications, based on the underlying design with the ability to maximize the probability of the sequence, this model is tuned to be used in various scenarios. One such scenario that is going to be of emphasis

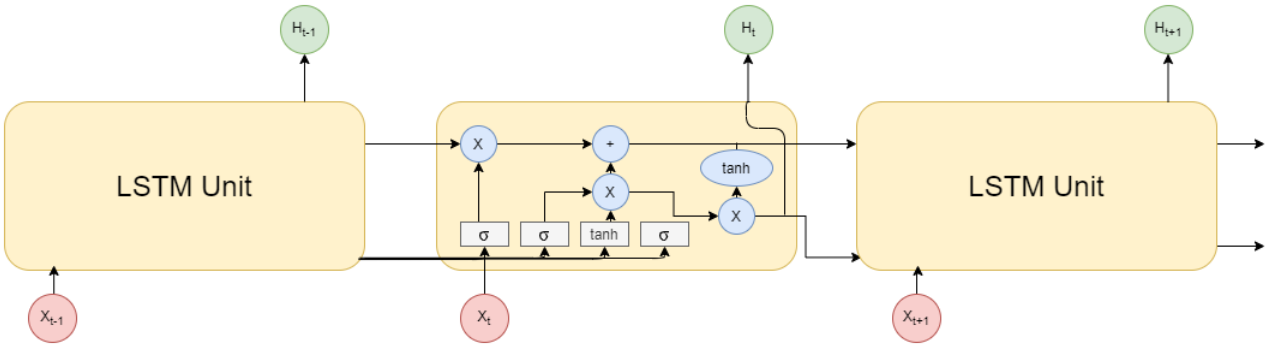


Figure 2.9: Cells of an unrolled LSTM network

is its ability to calculate the next part of the sequence. After a model is trained with the training dataset, given an initial sequence $s{=}w_1w_2...w_{n-1}$ for testing, the model should be able to predict a list of words that could be part of the sequence which is represented as

$$P(w_n|s) = \{x_1 : p_1, x2 : p_2, ..x_m : p_m\}.$$

For example, a model is trained with inputs [1,2,3],[1,2,4]. If a vector of [1,2] is inputted into the model during the testing phase, the output would be 3:0.5,4:0.5 in which 3,4 are outputs and 0.5 is the probability of occurrence. As opposed to Hidden Markov models which also function along with a similar basis, LSTMs can have deeper networks and don't get trapped into a local optimum.

Logistic Regression

Logistic regression is a statistical model designed for calculating probability based on data points (regression) but has been extended to solve classification problems due to its predictability nature. It has gained huge traction in recent years due to its ability to define a better decision boundary than other linear classifiers by fitting values onto the sigmoid curve. The objective function of logistic regression can be defined as

$$arg_{\beta}max : log\{\prod_{i=1}^{n} P(y_i|x_i)^{y_i}(1 - P(y_i|x_i))^{1-y_i}\}$$

This approach is called the maximum likelihood function where the probability function is a sigmoid function and each class's probability is calculated and put through this equation for all classes (from 1 to n) to estimate the final output.

Random forest

Farnaaz et al. discusses the usage of Random forest in the field of intrusion detection [22] and how random forest classifiers are one of the best classifiers to be used in the domain of cybersecurity. In theory, the random forest is a very simple algorithm. A random forest is an ensemble of decision trees where the decision trees make individual decisions on the classification problem and a majority voting is done to determine the final class. While it looks very simplistic, a combination of models making a decision is always better than a single model making a decision in most cases since a single model is more likely to overfit.

Other models

Beyond these models described above, there are a few other models used in the related work that prompts discussion and those are listed here.

- Convolutional Neural Network: Convolutional Neural Network (CNN) is a specific type of deep neural network that is used to process visual imagery by multiplying pixels with the weights of hidden layers.

- Autoencoder: Autoencoder is a neural network that is used to compress and automatically encode the data into a more efficient form through unsupervised learning.

- Radial Basis Function Network: Radial Basis Function Networks (RBF Networks) are neural networks that use a radial basis function as an activation function. This network's output is formed by applying the radial basis function over the input and neuron parameters.
- Support Vector Machine: Support Vector Machines (SVMs) are algorithms that help construct hyperplanes to create a decision boundary between classes and hence used widely for classification.
- SGDClassifier: SGDClassifier (Stochastic Gradient Descent Classifier) is a simple classifier that is optimized through Stochastic Gradient Descent. This optimization algorithm can be coupled with simple machine learning classifiers like Logistic Regression and NaiveBayes Classifier.
- NaiveBayes: NaiveBayes is a classification algorithm based on the Bayes theorem and is used only when there is a strong correlation between features of the same class.
- Adaboost: Adaboost is an ensemble learning method used over multiple binary classifiers that uses an iterative process to fix errors of weak classifiers and makes the weak classifiers reduce those errors in the future.
- DecisionStump: DecisionStump is a simple one-level decision tree that makes a decision based on a single feature.

2.5 Dataset

There are two widely used solutions for intrusion detection systems, behavioral analysis-based (anomaly-based) solutions, and signature-based solutions. Behavioral analysis is learning patterns based on data available while a signature-based solution is focused on pre-trained

signatures to be compared. While signature detection focuses on capturing known attacks with strong accuracy, behavioral analysis is touted as a better way to go about tackling the problem since it can learn new attacks on its own and detect them. To tackle the problem in this fashion, an abundance of data is needed for the model to learn the difference between malicious input and benign input. For our solution, abundant packet data, particularly HTTP requests and phishing certificates, are needed to generalize which requests or certificates are normal and which are malign. For this purpose, a dataset containing HTTP requests—CSIC 2010—is used and a phishing certificate dataset is manufactured for this work.

2.5.1 CSIC 2010

The HTTP CSIC 2010 dataset [11] contains generated anomalous and malicious web requests to an E-commerce malicious web application. It has been the standard dataset used in academia and the industry for testing web application intrusion detection or prevention systems since 2010. The anomalous requests in the dataset are of three types:

- Static attacks: Attacks which request hidden or non-existent resources. Eg: Session ID in URL rewrite, request hidden configuration files.
- Dynamic attacks: Modify valid request arguments. Examples include SQL injection (SQLi), cross-site request forgery (CSRF), and cross-site scripting (XSS).
- Unintentional illegal requests: Web requests which are modified from normal behaviour unintentionally.

This dataset contains 36,000 normal requests and 25,000 anomalous requests. The normal requests are split into test and training datasets while the anomalous requests are intended

for only testing. This dataset supersedes the DARPA KDD99 dataset and it contains actual HTTP web requests. The KDD dataset, created in 1999 contains, a variety of network features extracted from Transmission Control Protocol (TCP) packets. It is used in models to help them learn to identify network attacks. All of the data used for the generation of these web requests and attacks were used from a real-time database of the E-commerce web application.

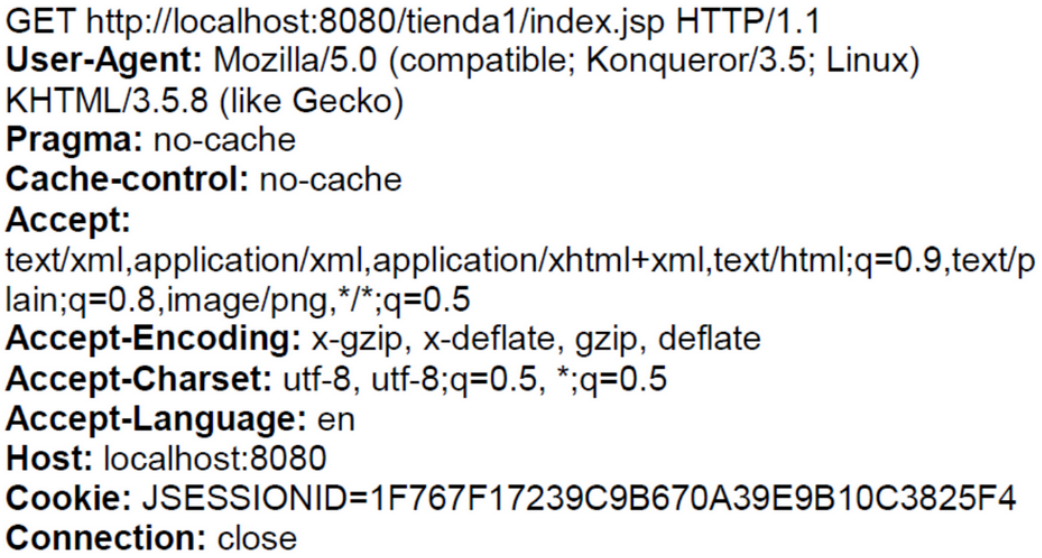


Figure 2.10: Sample datapoint from the CSIC dataset

2.5.2 Phishing Certificates

Phishtank [23] is a web service that uses crowdsourcing to get new lists of phishing sites and detects if they are phishing sites or not. Using the API they provide and a python script that constantly extracts data through the API once every 2 days, a list of phishing websites was formed. This process was done for 5 weeks and another script was used to ping the website and get its TLS certificate. During this process, 15,200 phishing site certificates

were extracted. For benign site certificates, TLS certificates were extracted from Alexa top one million [24] websites through random selection to gather 15,200 benign site certificates, to be proportional to the malign dataset.

Chapter 3

Design

3.1 Design Goals

As outlined in the motivation section, we would like a network intrusion detection system that is robust, quick and accurate, while satisfying these goals:

- The system should have high accuracy with an emphasis on reducing false positives.
- The system should incorporate a human-in-the-loop (HITL) mechanism to detect anomalies that don't conform to the dataset's distribution.
- The system should be able to learn in a distributed way to facilitate use in enterprise networks with high rates of network traffic.
- The system should learn from errors and improve over time, using active learning.
- The system should be modularizable and accept other add-ons for detection of other malicious activities beyond web application intrusions.

3.2 Design Overview

A holistic system has been designed as showcased in Fig. 3.1, with the aforementioned design goals in mind.

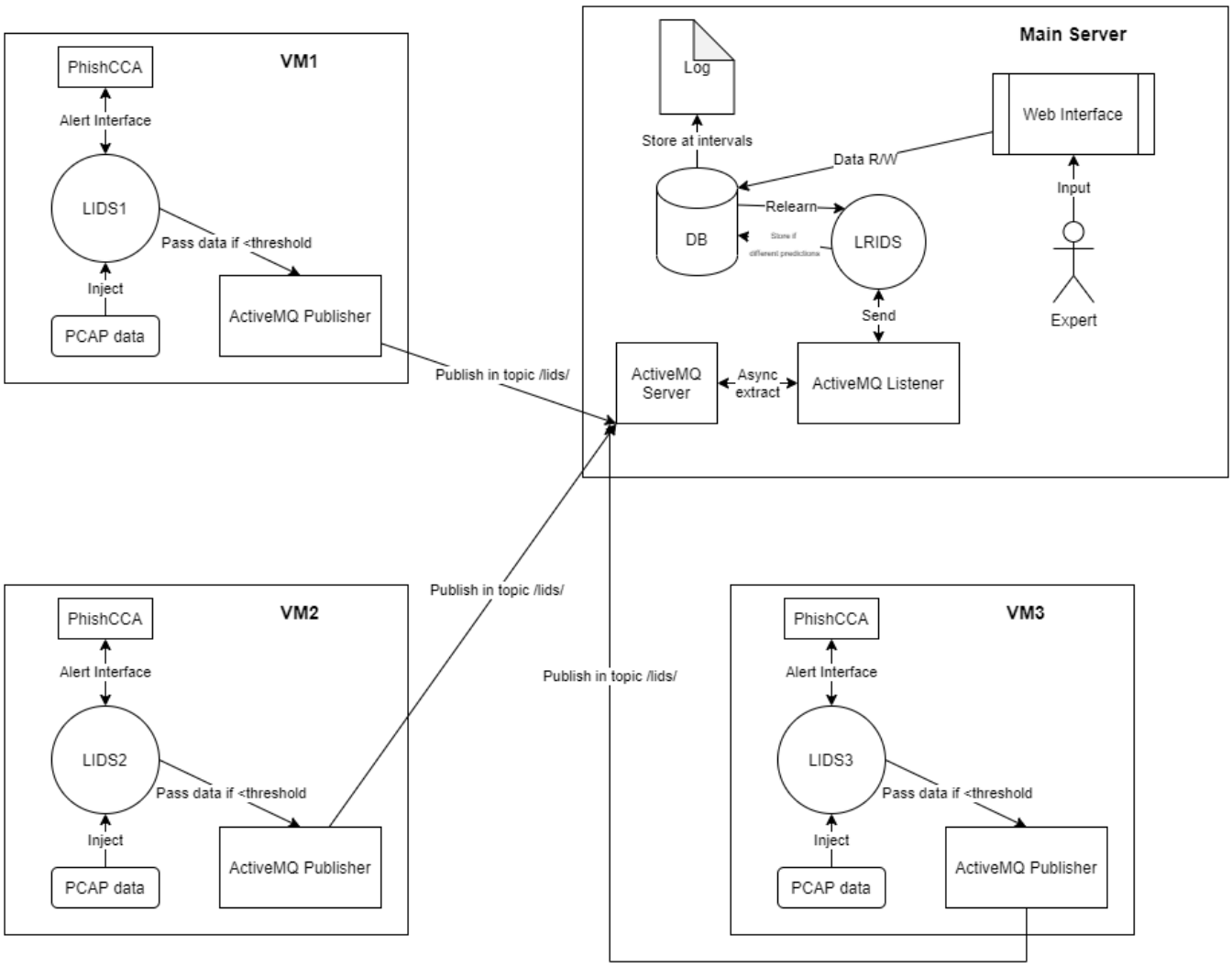


Figure 3.1: Overview of the complete system design

In the example system design, three Virtual machines are used as an example of a local datastore (distributed packet collection sites). The main server is a central server where most major components lie. The major components of this system are:

- LIDS - Novel **L**STM based **I**ntrusion **D**etection **S**ystem which has good accuracy with a very low false-positive rate.
- LRIDS - **L**ogistic **R**egression based **I**ntrusion **D**etection **S**ystem derived from another

work to be used for a second opinion in some situations (described later).

- PCAP data - Datapoints from packet captures that are available at local sites.
- PhishCCA - Phishing detection system based on TLS certificates (Module add-on).
- ActiveMQ [25] - Java-based message-oriented-middleware from Apache that is used as a message broker service.
- Active learning system - Consists of a web interface, database, and a logging service to help the intrusion detection system actively learn from expert opinions.

Individual components, including LIDS, LRIDS, and PhishCCA, are first trained individually using their respective datasets and tested to evaluate their performance. The trained models are then integrated into the system described in Fig. 3.1. All these components work together to form a functioning system that satisfies our goals. Fig. 3.3 showcases the overall working of the system using a swimlane diagram. In a local datastore (VM), the steps in Algorithm 1 are used to evaluate network data and identify anomalies as it is examined by LIDS.

Algorithm 1: System working - Local datastore

```
Result: Final prediction
datapoint=preprocess(PCAP);
(prediction,probability)=LIDS.predict(datapoint);
if probability<threshold then
    AMQsender.publish(datapoint);
    delay();
    AMQlistener.get(newPrediction);
    return newPrediction;
else
    return prediction;
end
```

Intrusion predictions are sent to the main server if the probability is below a pre-determined threshold, and Algorithm 2 is followed to make a final determination. During this process, the

Algorithm 2: System working - Main server

```
AMQlistener.get(lids_dp);
prediction=LRIDS.predict(dp);
if lids_dp.prediction!=prediction then
    dbms.insert(dp);
    expertSystem();
    newPrediction=dbms.select(dp.id);
    logs.insert(lids_dp,newPrediction);
    lrids.fit(lids_dp,newPrediction);
    AMQsender.publish(newPrediction);
else
    AMQsender.publish(prediction);
end
```

expert system uses a consortium of experts to manually input their prediction and majority votes are taken into consideration for the final prediction.

To summarize the prediction process, LIDS makes a local prediction for an input and if that probability is beyond a set threshold, the data point is passed to the main server. In the main server, the LRIDS makes a prediction and if it doesn't match the LIDS' prediction, the data point is sent to the expert system where experts decide on whether it is malicious or benign. The new prediction is stored in logs, used to update the LRIDS model, and passed along to the local datastore at user defined time intervals. This time interval is dependent on how often the user would like the LIDS model to be updated.

3.3 LIDS

LSTM based Intrusion detection system (LIDS) is the most integral component of this system, around which the whole system is designed. This model is a multilayered neural network with an Input, LSTM, Hidden, and Output layer in order. The LSTM layer is used to learn the probabilistic distribution of the sequences as discussed in Chapter 2 and the hidden layer

applies the activation function on it to get a good prediction. The system is designed based on Bayesian probability. Two methods were used—Single threshold and Dual-threshold—on the probability outputs. In the single threshold method, a fixed threshold is set below which a character's probability of occurrence in that sequence would lead to the datapoint being classified as malicious. In the dual-threshold method, two thresholds one for the current character and another for the next and previous characters around it was set (if the predicted class for the whole sequence is normal). Using these methodologies, frequently used characters in a web request that have high probabilities of occurrence would be classified as part of a benign request. The abnormal characters or an abnormal combination of characters would lead to the request being classified as malicious.

3.4 LRIDS

Pham et al. [26] show that a logistic regression-based IDS (LRIDS) is the best machine learning algorithm available with 96% accuracy amidst the common machine learning algorithms. It is also used in our system as an alternate IDS to the main LIDS as a second opinion before the prediction is passed to the expert system. Other systems like Transformers, Naive-Bayes based solutions were also tested to be used as the secondary IDS, but since LR has the highest accuracy after neural networks and is also one of the fastest models, this was chosen.

3.5 Phishing detection system

Phishing detection is an integral part of many intrusion detection systems in a corporate environment to avoid focused phishing attacks like spear phishing. Here, a phishing detection

system is developed and added as an add-on to showcase the modularizability of the overall system. Existing methods use URLs, HTML content, and other data sources to detect a phishing site. A key component of any site is its TLS certificate if it uses HTTPS protocol. According to the Anti-Phishing working group's analysis [27], 80% of phishing sites utilized HTTPS in 2020 and hence it is a viable source for features that might be used to differentiate phishing sites from benign ones. In this work, we create a phishing site classifier using the below-listed features from the TLS certificate.

- Textual features
  - Certificate Revocation List (CRL) URL
  - Domain Common Name (CN)
  - Certificate Authority (CA) Common Name
- Numerical features
  - Count of empty features
  - Country of Issuer and Owner (Mapped)
  - Organization of Issuer and Owner (Mapped)
  - Signature Algorithm (Mapped)
  - Difference between current and begin time
  - Difference between current and end time
  - Difference between end time and begin time

The selection of features was based on the individual influence of each certificate parameter in the classification process. For example, in Fig. 3.2, one of the features is the time difference between the current time and certificate begin time. This can help differentiate malign and

benign certificates. Likewise, other features and their combinations play an important role in this classification. Utilizing this, both an LSTM model and a random forest model are constructed. We select the best performing model to be used in the system based on their evaluation later.

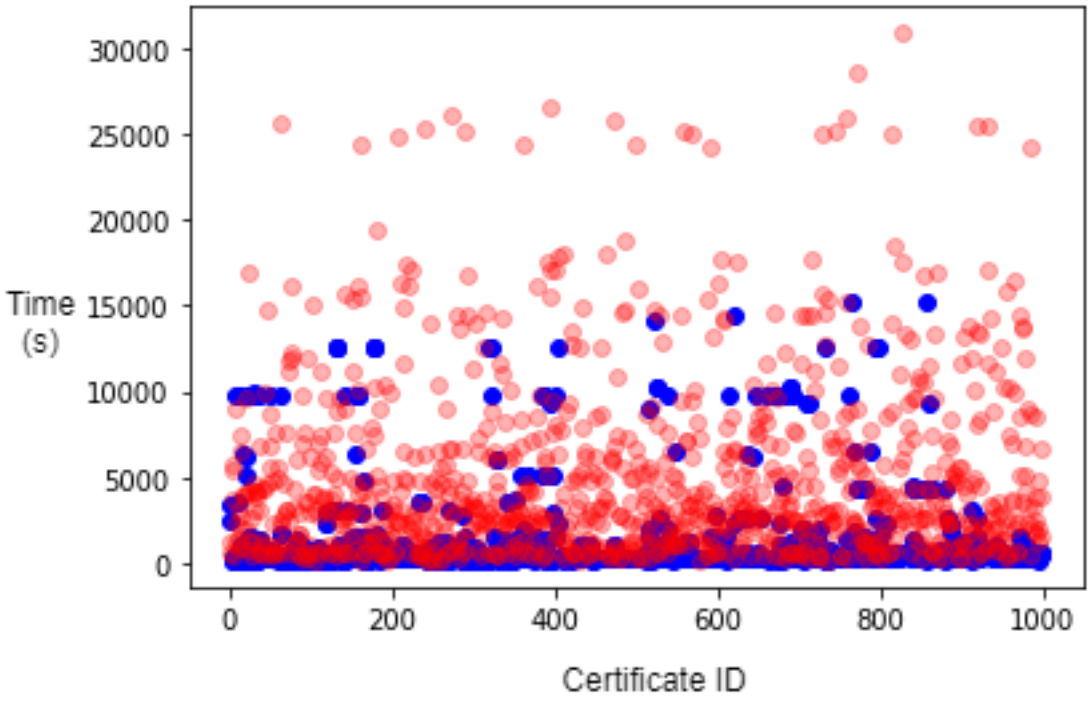


Figure 3.2: Plot of Current time - Issue time(Begin) of the certificate over all data points

3.6 Distributed active learning system

To meet two of our goals, a system is needed to make predictions in a distributed way while using an active learning mechanism. For the distributed component, Apache ActiveMQ (AMQ) is used as a message broker service. Predictions are made at the local level within a VM and passed to the main server using AMQ if it is within the pre-defined threshold.

This setup is constructed using a virtual machine that is also copied into multiple instances of the same snapshot. The main server has an Asynchronous ActiveMQ listener that listens for new messages pushed into the AMQ service from LIDS nodes. The listener then pulls them and makes a prediction using LRIDS and passes it along. LIDS cannot actively learn due to limitations imposed by one-class training (Keras cannot fit a new class different than the pre-trained classes into a pre-trained model). Hence, every false positive/false negative based on the expert's opinion is stored as logs in the main server and is passed on to the local datastore at user-defined intervals. This is used for retraining the model. LRIDS can actively learn on its own based on deviations from experts opinions.

3.6.1 Expert system

For the active learning component, an expert system is used if the prediction of the LRIDS and the LIDS for the same datapoint vary, for example, if LRIDS classifies a network packet as benign while LIDS classifies that same packet as malicious. All these data points are added to a database. The web interface extracts and displays these to a group of experts. Each expert enters a manual prediction and the final prediction is made based on the majority consensus. Each expert has a confidence value assigned to them, set at 100% initially that changes after tallying their prediction with the final prediction. If an expert's confidence value goes below 50%, the user's prediction is not taken into account.Their prediction will be taken into account only to change their confidence value and not for making a final prediction.

Scalability

To ensure the scalability of the active learning system, non-expert users can be added. The number of needed expert inputs can also be restricted to a particular value. From the work of Ion et al. [28], it can be learnt that security non-experts cause a reduction in quality of security related decisions. Hence, using non-experts in security is also considered a risk factor. To accommodate for all these issues, in this work, non-expert users are allowed (not considered during evaluation) during the sign-up process who are then shown a resources page in which the external resources are listed. These external resources help with the evaluation of the packets and are needed to be read (opened) to be allowed access to the evaluation page. This process is in line with the work of See et al. [29] through which we understand that non-experts can learn and even outperform experts with a good learning process. A suggestion that is not used in this work is weighing the input of expert and non-expert users to give the expert's input more value. Also, the attack vectors are expected to be learned over time to reduce attack querying.

3.7 Conclusion

In this chapter, we discuss five design goals and how they are achieved in the proposed implementation. Each of these five elements is listed below:

- The system should have high accuracy with an emphasis on reducing false positives. The foundations of LIDS are developed with a focus on reducing false positives and delivering high accuracy which would help achieve this goal.
- The system should incorporate a human-in-the-loop (HITL) mechanism to detect anomalies that don't conform to the dataset's distribution. The active learning mech-

anism with the consortium of experts based on the difference in predictions between LIDS and LRIDS resolves this goal.

- The system should be able to learn in a distributed way to facilitate use in enterprise networks with high rates of network traffic. The development of a prototype using a distributed model of learning achieves this goal.

- The system should learn from errors and improve over time, using active learning. An active learning system with a group of experts helps achieve this design goal.

- The system should be modularizable and accept other add-ons for detection of other malicious activities beyond web application intrusions. The integration of a phishing detection component serves as the add-on for testing. All components in this holistic system are modularized and can be replaced with ease.

By incorporating the above listed components into our overall system, we meet the design requirements outlined at the beginning of this chapter. In the next chapter, we discuss the details involving the implementation of this system.

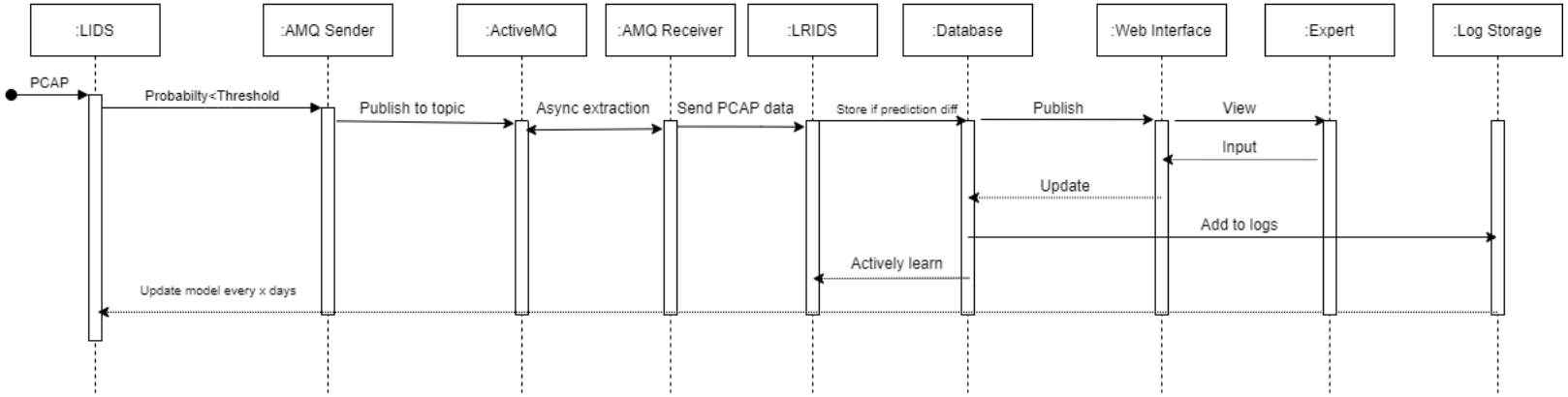


Figure 3.3: Workflow of the system represented through a swimlane diagram

Chapter 4

Implementation

4.1 Intrusion Detection System

The intrusion detection system includes two components, our novel LSTM-based IDS (LIDS), and the logistic-regression-based IDS (LRIDS) developed by Pham et al. [26]. LIDS is a multi-layered LSTM model that was implemented using Keras with Tensorflow as the backend. The CSIC2010 dataset was cleaned to remove redundant data points. In the anomalous dataset, even human errors (the third type in the dataset composed primarily of mistyped URLs) were considered anomalous requests whereas, in real-world scenarios, they are not considered as intrusions. Since this is not within the scope of our work, a data cleaning process is necessary to remove these data points. This data cleaning process is consistent with the machine learning domain and a few other works [30] [31]. In the data preprocessing step, requests were processed through Regular Expressions and then one-hot encoded, a vectorization technique discussed in Chapter 2. Using these one-hot encoded vectors, sequences were constructed, padded, and then fed into the LSTM model for training. The LSTM model has an input of 30 units (length of padded input vector) that are then passed into the LSTM layer, TimeDistributed layer, and finally, the activation layer which contained a softmax activation function to output the probability. The dataset was already provided in a split manner for training and testing. Hence, the model was trained only with normal requests (train dataset) to establish a baseline and any deviation from that

baseline would be considered an anomalous request. As mentioned in Chapter 3, pre-defined thresholds are used for classification.

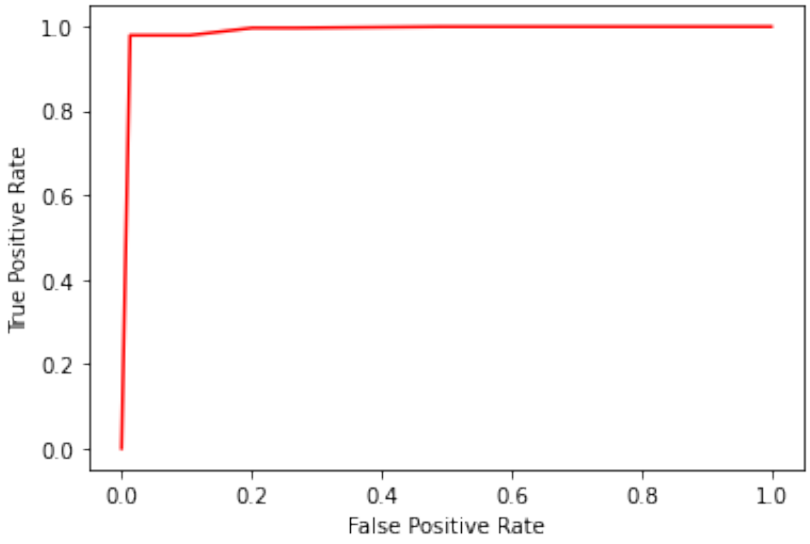


Figure 4.1: Receiver Operating Characteristic (ROC) curve for the threshold between 0 and 0.014

The Receiver Operating Characteristic (ROC) curve in Fig. 4.1 depicts the classifier's operation with varying thresholds. This helps identify the best solution considering the tradeoffs between True Positive Rate (TPR) and False Positive Rate (FPR). In a typical classifier, the threshold varies from 0 to 1 since the threshold is internally calculated. In this work, the threshold is empirically defined and we varied it between 0 and 0.014 with a single threshold LIDS system in increments of 0.0002. This allowed us to determine thresholds that resulted in the best performance for implementation. In the single threshold method, a threshold of 0.0005 is set. In the dual-threshold method, a threshold of 0.005 for the current character and 0.001 for neighboring characters is set.

LRIDS is a simple logistic regression model which was developed from the Scikit-learn library derived from another work [26]. The same cleaned dataset from LIDS is used in this model as well. This model is trained on both normal and malicious data, which is different than

Technology	Description
Python	Programming Language used
Scikit-learn	Python library that primarily contains vectorization techniques and machine learning algorithms
Keras	Python library that helps build deep learning models
Numpy	Python library that is useful for mathematical functions on large, multi-dimensional arrays
Matplotlib	Python library used for visualization, particularly plotting data
Tensorflow	Python library that serves as the backend for the deep learning algorithms to function
Jupyter Notebook	Interactive web tool used for running code inline in segments with extended visualization techniques

Table 4.1: Technologies used for IDS and Phishing module development

the one-class training method used for LIDS. LRIDS is trained with both classes of data to be in line with its original work [26]. For the data preprocessing step, Scikit-learn's TF-IDF Vectorizer is used to convert the textual sequences into vectors. These vectors are fed directly into the Logistic regression model for training and testing both.

The implementation of both these models was done primarily on Jupyter notebook [32] and was trained on the high-performance computing resources in Virginia Tech's Advanced Research Computing division [33] .

4.2 Phishing detection system

As listed in Chapter 3, certain features were extracted from the certificates using Python scripts. For the phishing detection system, two models—LSTM and a Random forest—were developed with the Tech stack listed in 4.1. Data preprocessing on textual features was done by directly feeding the data into a Scikit-learn's TF-IDF Tokenizer. This process helped tokenize text into vectors suitable for input into any machine learning algorithm.

Data processing on numerical features was done differently for the Random Forest model and the LSTM model. For the Random Forest model, the features were directly supplied after a label encoding technique was applied to map particular labels like organization, country, and signature algorithm to numerical values. For the LSTM model, the numerical features were supplied as one-hot encoded vectors. The dataset was then split in the ratio of 80:20 for testing and training respectively. The Random Forest model is a traditional ensemble of decision trees that was used due to the abundance of available features. With a maximum depth of 30 and the criterion being 'gini', the random forest classifier was trained on the selected features. 'gini' criterion is used to calculate the best split when training the individual decision trees in the random forest. This is established by calculating the probability of incorrectly classifying a randomly chosen element in the dataset. The LSTM model as showcased in Fig. 4.2, has 4 input layers. The 3rd Input layer takes in the encoded numerical features. The other 3 input layers take in Domain Common Name (CN), Certificate Authority (CA) Organization's CN, and the certificate revocation list (CRL) URL respectively. LSTM layers are used for the processing of sequences and then passed through Dense layers for activation towards output through the "Leaky ReLu" activation function. "Leaky ReLu" fixes the vanishing gradient problem which is persistent in LSTM models. These outputs are finally concatenated to be passed through a final Dense layer which outputs the final result through a "sigmoid" activation function in form of a binary class. Since this is a binary classification problem, the standard loss function of "binary crossentropy" is used along with the widely adopted "adam" optimizer function. The number of epochs was 25 and the batch size was 256. Batch size is the number of samples processed before the model is updated and epoch is the number of iterations a model is run through during the training process. By standard practice, both of these are decided by varying them, typically by increasing batch sizes in increments of 32 while reducing epochs from a high value looking for better results.

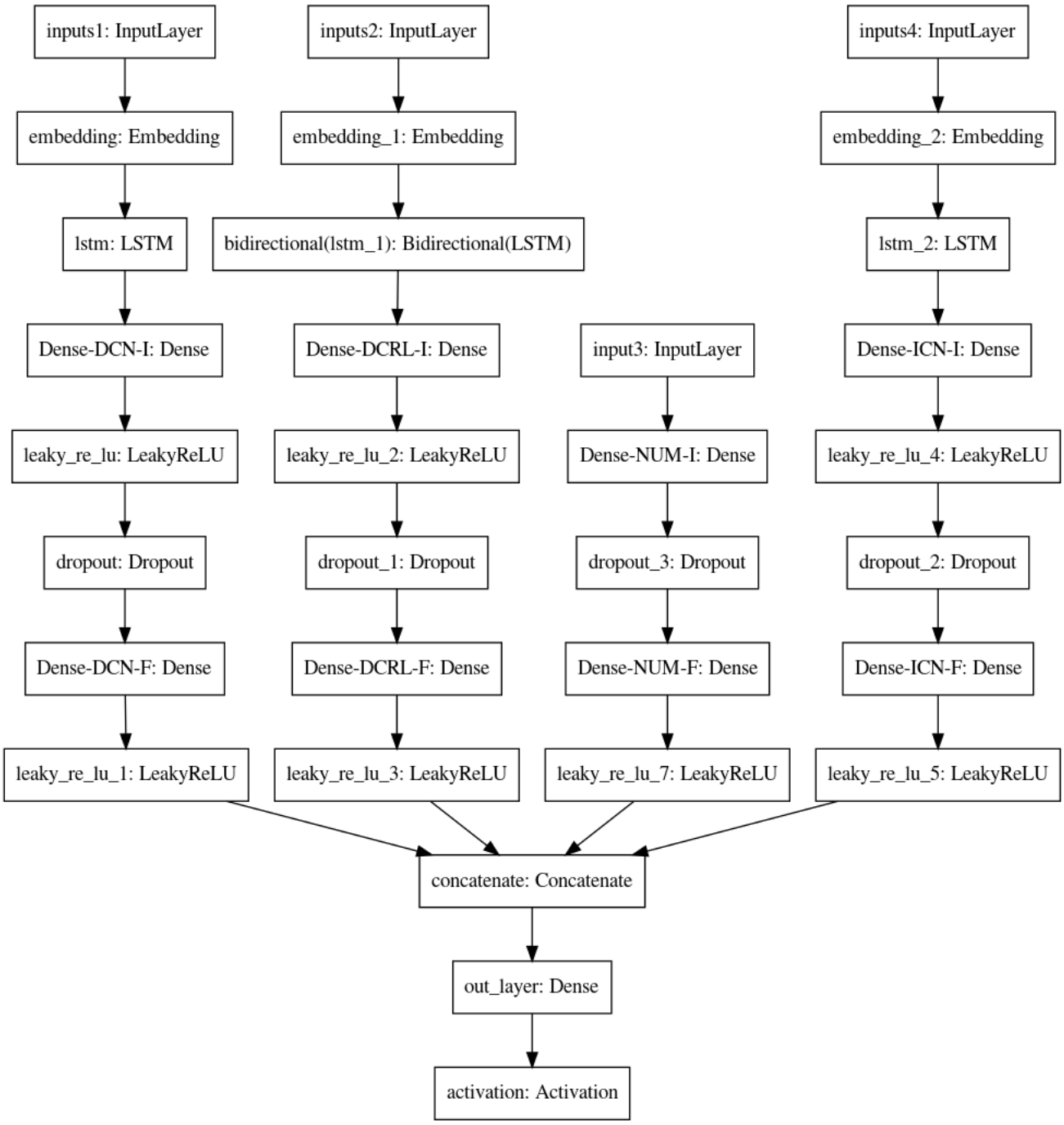


Figure 4.2: LSTM model created for phishing detection

4.3 Integrated system

To represent a distributed model, three Virtual machines (VMs) were created with the host machine representing the main server and the VMs representing distributed nodes. VirtualBox [34] was used for device virtualization, with virtual servers running on Ubuntu16.04. The host machine was running Windows 10 with Windows Subsystem for Linux. A message broker system, Apache ActiveMQ was used for communication between the distributed nodes and the host machine. To send data, a message is pushed into a queue in the AMQ service. To receive data, an asynchronous listener script checks the AMQ service and retrieves messages as they arrive. Using this design, LIDS that is situated at each distributed node would push the packet data if the probability of prediction is within the LIDS threshold. At the host machine, the data is pulled through the listener script and passed onto the LRIDS. LRIDS then makes a prediction and injects the data into a MySQL database if the LRIDS prediction doesn't match the LIDS prediction. These mismatched predictions are then presented to our experts through a web interface as shown in Fig. 4.3. This website was constructed using the Python-Flask framework. After the experts' opinion is entered, the prediction is sent back to the nodes using AMQ and updated in the database accordingly. Gathering the expert opinion is an asynchronous process and other packets are processed while the LIDS waits for results from the experts. Scalability needs to be ensured to make sure this is not a bottleneck. Currently, the expert system is set up in such a way that all experts need to input to make a final prediction. When more experts are integrated into the system, the current system can easily be modified to predict once a certain number of experts input their prediction.

When a URL arrives at the phishing prediction system, the system grabs the TLS certificate of the URL and makes a prediction at the local node. Using a python script to read emails from an email server, all URLs inside it are grabbed and run through the system to detect

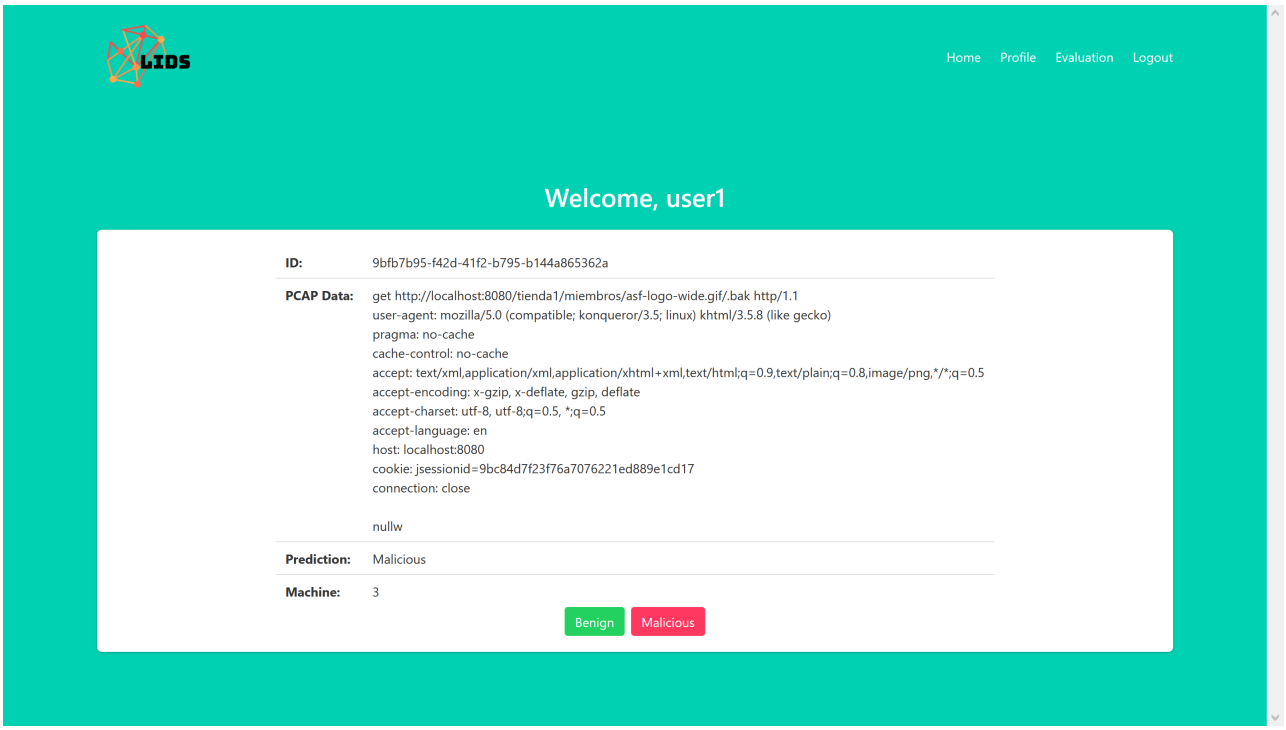


Figure 4.3: Snapshot of the web interface with the perspective of the expert

phishing sites and generate IDS alerts.

Technology	**Description**
Python	Programming Language used
VirtualBox	Virtualization software
Apache ActiveMQ	Message broker system used to communicate between distributed nodes and the main server
Ubuntu 16.04	Operating system used in the virtual machine
Windows 10	Operating system used in the host machine
Stompest	Python library that is used to connect to the AMQ Service
MySQL	SQL based Database management system
Flask	Python based micro web framework

Table 4.2: Technologies used for integration

Chapter 5

Evaluation

In this chapter, we evaluate our LSTM based intrusion detection system (LIDS) and our phishing detection system with the test datasets. After that, the complete system, with pre-trained models embedded, is tested with select data points. Finally, case studies are presented to showcase the functionality of the individual components in real-time scenarios.

5.1 IDS

LIDS and LRIDS are evaluated using the test subset of the CSIC 2010 dataset. Select metrics like accuracy, precision, recall, and false-positive rate are used to evaluate the model. Our IDS systems are measured for their accuracy and false-positive rates (FPR) and we report our results below. The above-mentioned metrics are calculated as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

$$FalsePositiveRate = \frac{FalsePositive}{FalsePositive + FalseNegative}$$

True positives are the anomalous requests classified accurately as anomalous. True negatives are the benign requests classified accurately as benign. False positives are benign requests classified inaccurately as anomalous. False negatives are anomalous requests classified inaccurately as benign.

5.1.1 LIDS

As discussed in Chapter 4, LIDS is trained with a dataset that contains only routine, non-malicious web traffic to learn the baseline. In the testing phase, it is tested against a dataset containing malicious and non-malicious data, which is available as a test subset in the CSIC dataset. Like a typical evaluation, loss function calculation and multi-fold testing would not be useful in this model since the model is trained only on one class of data. Calculating loss function value is not helpful since the neural network is not learning from multiple classes and inaccurately classifying the data points. In other words, there is no decision-making involved, hence no errors, and hence no loss function value attributed to it. Multi-fold testing is typically used to understand if a model has been overfit. Here the model is trained on only one class of data, hence it can be inferred that there is no room for overfitting since two classes will be used for testing. The metrics considered for evaluating this model are Accuracy, Precision, and Recall. *Precision* measures the amount of anomalous request predictions that belong to the actual anomalous class, *recall* measures the amount of correct anomalous request predictions made out of all the anomalous requests available in the test dataset, and *accuracy* measures the percentage of correct predictions across both classes. Results of the model are:

- Accuracy - 0.990
- Precision - 0.992

- Recall - 0.983
- False Positive Rate - 0.0005

To evaluate the speed of prediction, 1000 data points were picked and predictions were made. An average of the time taken to make those predictions was calculated which is 0.00061708 seconds. The impact of the speed of prediction is further discussed in Section 6.3.

5.1.2 LRIDS

As discussed in Chapter 4, LRIDS is trained with both normal and malicious data as a replication of the work by Pham et al. [26]. The testing phase involves the same process as LIDS and the same metrics are used. Results of the model are:

- Accuracy - 0.974
- Precision - 0.976
- Recall - 0.979
- False Positive Rate - 0.0328

5.2 Phishing detection system

The Phishing detection system is composed of two machine learning models, Random Forest and LSTM. Each model is trained and evaluated with the manufactured dataset as discussed in 2.5.2 with a ratio of 80:20 for training and testing respectively. In this case, three major metrics are considered for evaluation of the certificate classification, accuracy, precision and recall. Results of these evaluations are shown in Table 5.1. As you can see from the results,

Model	Accuracy	Precision	Recall
Random Forest	0.98	0.98	0.998
LSTM	0.76	0.78	0.72

Table 5.1: Classification results of certificates for phishing detection

Random Forest performs much better than the LSTM model and hence the primary model used in the complete system.

The Random Forest seems to outperform the LSTM model, even though the LSTM model is deeper and there is a need for the processing of sequences. We hypothesize two possible reasons for this:

- As shown in Fig. 5.1, the loss function saturates after a particular epoch. This means that the model doesn't learn beyond this epoch. This could further mean that the currently available data doesn't allow for more learning.
- This solution is consistent with the notion [35] that random forest is one of the best models for classifying multi-variate data points.

5.3 Complete system

For evaluation of the complete system, 2000 requests were selected at random from the CSIC 2010 test dataset. No phishing messages were added as part of this evaluation. This was passed into LIDS in which the requests within ±10 % of the actual LIDS threshold are passed onto the LRIDS. In our experiment, 36 requests met the threshold and were forwarded to the LRIDS. In the LRIDS, predictions are made and 16 predictions varied from the LIDS prediction values. These requests were passed into the expert system for analysis. We assume for our analysis that the experts always make the right prediction.

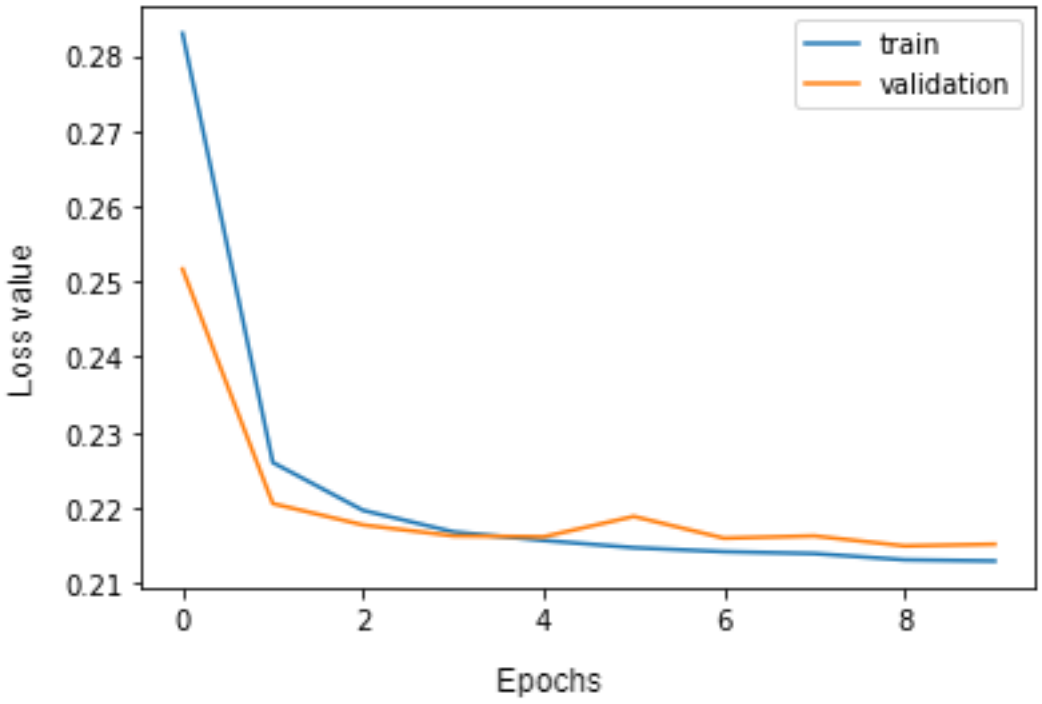


Figure 5.1: Graph of loss function while training the LSTM model for phishing detection

Among the 16 predictions made, 12 predictions were different from the predictions made by LIDS. According to our assumption, experts don't make mistakes and hence the predictions from the LIDS are erroneous. The expert system, therefore, helped identify 12 erroneous predictions from LIDS.

5.4 Case Studies

To demonstrate the usability of the individual components in real-time, we describe a short list of case studies below:

5.4.1 Select attacks

SQL Injection

SQL Injection (SQLi) is placed at the top of the OWASP Top 10 list, as part of the Injection attacks. SQL Injections are viable when the input from the end-user is not properly sanitized and the internal SQL queries are not parametrized. This could let the attackers manipulate the query internally to compromise the database. RedTiger's Hackit [36] is a challenge platform where the website has SQLi vulnerabilities as part of the challenge and is widely used for practice. The goal of the first challenge in this website is to use a SQL Injection vulnerability to gather credentials. In this challenge's page, the input for the cat parameter (category) in the GET request is not sanitized and is directly used to query the database. Using a SQL query to select the credentials in the database is quite straightforward with the above information. The select query is inputted through the request and the result (credentials) is displayed on the website. This request is replicated in Postman [37] and the packet data is exported in a JSON format. It is then reconstructed as shown in Fig. 5.2 and fed into LIDS which predicts this as a malicious request.

Cross Site Scripting

Cross-site Scripting (XSS) is a client-side code injection attack that is the most common cyber attack making up for 40% of web attacks [38]. It enables the attacker to execute malicious scripts on the client-side. Using this type of attack, a malicious script can be delivered to the end-user that will be executed in their browser because the script is being delivered from a trusted source, the website. Pwnfunction [39] is a challenge platform that has XSS challenges. The first challenge is a simple challenge in which the XSS vulnerability should be exploited to pop a JavaScript alert in the attacker's browser. The parameter

"1","GET https://redtiger.labs.overthewire.org/level1.php?
cat=1%20UNION%20SELECT%201,2,username,password%20from%20level1_users
HTTP/1.1
User-Agent: PostmanRuntime/7.26.5
Pragma: no-cache
Cache-control: no-cache
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: overthewire.org
Connection: keep-alive

null"

Figure 5.2: Packet sent that contained a malicious request (SQLi) to the RedTiger website

"somebody" in the GET request is not properly sanitized and a simple Javascript code can be inserted through the parameter which is reflected on the client-side. This request is replicated in Postman followed by the packet data being exported in a JSON format. It is then reconstructed as shown in Fig. 5.3 and fed into LIDS which predicts this as a malicious request.

5.4.2 Phishing detection

OpenPhish [40] is a website that provides live lists of phishing websites as they are added to blacklists. Ten random phishing websites from OpenPhish that utilize HTTPS were selected from the list which are listed below:

- https://saisoncardkey.com/mobile saisoncard-signin.php

"1","GET https://sandbox.pwnfunction.com/warmups/ma-spaghet.html?
somebody=%3Csvg%20onload=alert(1337)%3E
HTTP/1.1
User-Agent: PostmanRuntime/7.26.5
Pragma: no-cache
Cache-control: no-cache
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: pwnfunction.com
Connection: keep-alive

null"

Figure 5.3: Packet sent that contained a malicious request(XSS) to the Pwnfunction website

- https://customeropudol.com/
- https://easytdauth.com/home/en.html
- https://natwest.remove-unauthorised-payee.co.uk/Login.php
- https://www.visadpsgiftcard.com/navyfederal/
- https://hsbc.payees-manage.live
- https://myjcb.cn.solutionstouproof buzz
- https://eventgarenafreefirebgid.my.id/
- https://authenticate-device.me/admin
- https://mail.profilo-app.com/info/index.html

All these websites were run through the phishing detection module in which 8 sites were detected as phishing sites, 1 was detected as a benign site and the TLS certificate was not extractable from 1 other site.

5.5 Limitations

While we have shown that our system is highly successful in detecting intrusions with a low false-positive rate, there are some limitations in our testing and evaluation that could be explored in future work.

- Lack of real-time evaluation: Real-time network packets from a data stream could not be evaluated because we do not have ground-truth data identifying the packets as truly malicious or non-malicious.

- Need for a better dataset: Even though a modern dataset is used, this dataset is created from a simulated environment and does not completely reflect real-world attack scenarios.

- Lack of evaluation on scalability: Scalability of the distributed process could not be evaluated due to lack of infrastructure.

- Inability to utilize all evaluation metrics: Due to the one class training methodology coupled with empirical threshold setting, multiple machine learning metrics could not be calculated or are not useful which can be used in a typical classifier setting.

In this chapter, we evaluate our LIDS and phishing detection models with appropriate machine learning metrics and further study the working of these models in real time through

case studies. Through this, we can understand our complete system can work in a real time environment with accurate predictions.

Chapter 6

Related work

In this chapter we describe existing intrusion detection systems (IDS) that our work builds upon. We also describe anti-phishing systems, which are also relevant to our IDS system design

6.1 Intrusion detection systems

Intrusion detection systems are devices or software that are used to detect malicious network behavior. They are primarily of two types, signature-based and anomaly-based. These are discussed below.

6.1.1 Signature based IDS

Signature-based Intrusion detection systems (IDS) rely on the knowledge of existing attack signatures to identify malicious network traffic. In widely used signature-based intrusion detection systems like Snort [2], Zeek [3], and OSSEC [41] pre-existing patterns, called rules, are used for detection. IDS rules are based on existing indicators of compromise, and are created such that different attacks conform to different rules. If a network packet or system log entry matches a particular pattern, it is detected as an intrusion and reported. Indicators of compromise could constitute a variety of things, such as email data, malicious Internet

protocol addresses, or malicious byte codes in a transmitted file. Typically, in modern enterprises, Signature-based IDSs are widely adopted because of various advantages. They are:

- Developed and marketed by commercial companies, who sell complete solutions and provide customer support for these products.
- Regularly updated software and rulesets.
- Less error-prone than current Anomaly-based IDS in detecting existing attacks, which account for the majority of the attacks against systems.

Snort is the most widely used Network Intrusion detection system [2]. Currently owned and developed by Cisco, Snort is an open-source tool that has three major functions:

1. Packet sniffing
2. Packet logging
3. Network intrusion detection (or prevention) system

Snort can operate either in intrusion detection (IDS) mode, or in intrusion prevention (IPS) mode. In IDS mode, Snort observes network traffic using a network tap or switched port analyzer (SPAN) port and signals network alerts if a rule is triggered. In IPS mode, Snort sits inline with network traffic and can block packets that trigger alerts. Snort is widely adopted due to its ability to operate alone and if needed, inline without depending on any external software or hardware. Snort's IPS operates primarily based on the set of rules contributed by Snort's community of users. These rules define different kinds of malicious activity and are used to trap them to generate alerts for the end-users.

Zeek was primarily developed as a tool to generate compact network event data extracted from available network data. Zeek is customizable to allow the end-user to collect all available network metadata and then process it based on their functional requirement (for example, process the packet data for intrusion detection). The development of the Signature framework, which is very similar to Snort's low-level pattern matching techniques, paved the way for Zeek to be used as an IDS [3].

OSSEC is a host-based intrusion detection system that operates differently from Snort and Zeek. As opposed to dissecting network packets and analyzing them with rules, OSSEC scans the logs of a system on which it is installed and detects malicious behavior based on its own set of rules. Furthermore, OSSEC also runs a file integrity check by checking for file hashes at particular time intervals. It also detects rootkits, malware, and intrusions based on log files.

Other widely use IDSs include Suricata [42], Cisco Stealthwatch [43], TippingPoint [44], Samhain [45] all of which detects intrusions based on existing signatures.

6.1.2 Anomaly based IDS

Anomaly-based Intrusion detection systems are trained with a baseline of a normally functioning system and any deviation from this normal baseline is considered an anomaly. Although they are not as widely used as signature-based IDSs, anomaly-based intrusion detection systems are growing in usage with the advent of machine learning. Machine learning-based Intrusion detection systems are primarily anomaly-based since they are trained on existing data that serve as the baseline for the system to understand what is normal data and what is anomalous. There are two types of such systems: One class training, where normal system data is fed for the algorithm to learn the baseline of the normal system, and

all class training where both normal and anomalous data is fed to learn both baselines.

As discussed in Chapter 2, two major datasets, CSIC 2010 and KDD99, are used to train these systems. In one of the earliest works in this domain, Siaterlis [5] uses a Bayesian model, based on Dempster-Shafer's theory of evidence that is trained and evaluated over the KDD99 dataset for the detection of Distributed Denial of Service attacks. Although this was focused on the detection of one particular attack vector, this work introduced the use of statistics and probability theory to detect anomalies from network data. The Boltzmann machine [46], a stochastic energy-based network, is also trained and evaluated using the KDD99 dataset to detect anomalies. Boltzmann machines are widely used to self-adapt to new forms of data (network attacks in this scenario) that don't conform to the training dataset. In this work, the authors deploy the one-class training model in which the normal data from KDD is used to train the model, and testing is done using both classes. Javaid et al. take this work further by utilizing neural networks [6] to tackle this problem. In their work, an autoencoder is used to learn features from unlabeled data, and a classifier is then used to classify them based on the learned features. Mohammad et al. [7] tackle this problem with a different approach using Convolution neural networks (CNNs). In this work, the author encodes the data using a 1-n encoding technique with max-min normalization to get it in the range of [0,1]. Using this method, a [11x11] matrix is constructed using the 121 features from the KDD99 dataset to be trained and tested using the CNN model.

While a majority of the work on anomaly-based IDSs has been trained and evaluated on the KDD99 dataset due to its long-lasting presence, since the introduction of the CSIC dataset in 2010, various solutions have used it for their works. Pham et al. [26] utilize various machine learning algorithms like Random forest, Adaboost, logistic regression, and SGDClassifier (Stochastic Gradient Descent Classifier) to be trained and evaluated over the CSIC dataset to understand which machine learning algorithm performs best as a network

intrusion detection system. The authors conclude that logistic regression performs best while Adaboost performs the worst. Similar to Javaid et al.'s work [6] utilizing the KDD dataset, this solution [9] uses a stacked Autoencoder to extract features from the CSIC dataset and passes it along to an isolation forest for classification between normal and anomalous requests. The selection of features is key to a good model. In that regard, Nguyen et al. [47] utilize a generic feature selection method which is used to identify redundant features to remove unnecessary features from the CSIC dataset. Then pre-existing classification algorithms are used to showcase good results. Pastrana et al. [8] discuss how the dataset and its representation through features is key. They utilize 1-gram methodology and various classification algorithms like Support Vector Machines (SVM), Multilayer Perceptron (MLP), etc. to showcase how 1-gram vectorization is ideal for this problem through their results. Choraś et al. [31] take a different approach and use a graph-based segmentation technique to generate patterns of normal and abnormal behavior and based on these patterns, intrusions are detected which is similar to signature-based IDS. Bochem et al. [48] utilize a Long Short Term memory network to be trained and evaluated over the normal unprocessed CSIC dataset to be used as an IDS.

Active learning systems

Active learning systems are machine learning-based systems where humans are involved in the loop (HITL) to actively improve the model if it makes erroneous predictions. In cybersecurity, this has been an active area of interest recently due to the high cost of error associated with any false negatives. In one of the earliest works of active learning, Abe et al. [49] focuses on improving the detection of outliers by involving an expert in the classification process. The work is not focused on intrusion detection, but rather outlier detection and uses the KDD99 dataset for the process since intrusions are considered as outliers. This

work portrays the potential of active learning to drastically reduce false positive and false negative rates. An interesting work released as a Microsoft technical report - ALADIN [50], uses an active learning system to improve the classification process and not just accuracy when evaluated over the KDD dataset. In this work, traffic that does not fit into pre-existing classes, as well as traffic that does not fit into a class with assurance, are sent to the expert for classification. The work showcases that classification accuracy is improved while rarely occurring classes are identified with about half the number of labels from the original classification model. Ziai et al. [12] develop an active learning model by training and testing over the KDD dataset in which the expert classifies unlabelled data that is then compared to the result of the classifier and the classifier is then updated. The approach used in each component is highly flexible. Different classifiers and different sampling strategies are used for unlabeled data selection, which are used to study the active learning methodology use for intrusion detection systems.

Distributed learning systems

Distributed learning systems are machine learning-based systems where multiple inputs from individual systems are taken to make one final decision. These can be of two types — the same machine learning model used in multiple local machines, or different machine learning models used in a single machine. Nguyen et al. [51] has adopted the latter strategy to create multiple intrusion detection models using NaiveBayes, BayesNetwork, Decision Stump, and RBFNetwork to be evaluated over the CSIC dataset. The decisions from these models are combined to form a final decision. Idhammad et al. [52] takes on a different model for cloud-based systems where a distributed intrusion detection model would be beneficial. In this work, the author utilizes a single model spread across local nodes for classification, and the network data is synchronized into the central server.

6.2 Phishing detection systems

Phishing detection systems with the application of machine learning have used various features from websites to detect their malicious nature. These features have included URLs embedded in phishing messages, HTML content from messages, and screenshots. Some works [53], [54], [55] have utilized features from the URL to determine if a site is a phishing site or a benign site through different methods. While Feroz et al. [53] use a clustering mechanism to combine similar URLs based on host and lexical features that are then classified to determine phishing sites, James et al. [54] take a different approach by first doing feature selection and then running various machine learning algorithms for the classification problem. Bahnsen et al. [55] use a Recurrent Neural Network to classify websites as phishing or benign by directly feeding the URLs through the neural network.

Content of the phishing page is also considered useful as a feature for classification. Opara et al. [56] utilize only the HTML content of a website as the data over a CNN to learn the semantics and the structure of the document that further classifies it as a phishing or a benign site. To improve on this work, Tian et al. [57] utilize both the HTML content and visual analysis data using optical character recognition (OCR) on screenshots of the webpage to identify phishing websites. While tackling the issue of classifying malicious sites in general, Torroledo et al. [58] also classifies phishing sites as part of this work utilizing features from TLS certificates over an LSTM model.

6.3 Evaluation - Comparison

We now compare our results with the results of existing works which are tabulated in Table 6.1. Results of existing works are either gathered directly from their publication or the

Authors	Model	Year	Accuracy (%)	FPR (%)
Nguyen et al.	Combination	2012	90.98	-
Nguyen et al.	C4.5	2013	94.49	5.9
Pastrana et al.	C4.5	2015	97.00	4
Chora et al.	RegEx generation	2015	94.46	4.34
Bochem et al.	LSTM	2017	97.00	4
LRIDS (Pham et al.)	Logistic Regression	2016	97.46	3.28
LIDS (Single Threshold)	LSTM	2020	86.58	0.2
LIDS (Dual Threshold)	LSTM	2020	99.03	0.5

Table 6.1: Comparison of results with IDS using CSIC 2010

artifact they have published.

We see in Fig. 6.1 that LIDS outperforms existing works in accuracy. LIDS also outperforms existing works drastically in the false positive rate which is also a key measurement for IDSs.

We cannot compare the speed of prediction of LIDS with existing works since they haven't evaluated the speed of prediction of their model in their works. Based on previously stated calculation, 1667 HTTP packets can be processed in one second by LIDS. Depending on the scalability of the distributed system, huge volumes of packets can be processed in a second. For example, with 100 distributed nodes, approximately 160,000 HTTP packets can be processed in one second. Typically a HTTP request header varies in sizes between 200 bytes and 2 KB [59]. Assuming a whole packet with both header and body is twice the size, the range of size of a packet comes out to be between 400 bytes and 4KB. If 160,000 HTTP packets can be processed per second and assuming an average size of the packet to be 2.2KB (mean value of 400 bytes and 4KB), that equates to a link speed of 3.52GB that can be processed in line.

In this chapter, we explain in detail about the related works in this domain and compare them against our work. From the evaluation, it can be learnt that our work outperforms existing works in both speed and false positive rate.

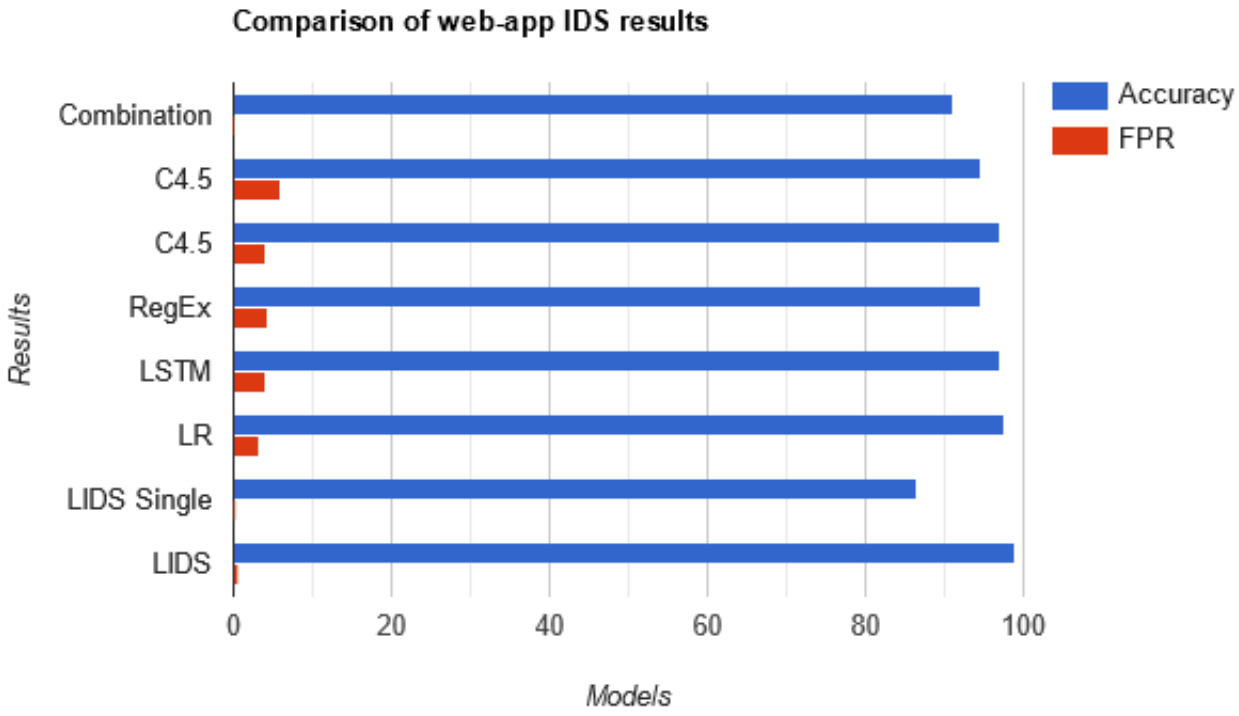


Figure 6.1: Comparison of web-app IDS results

Chapter 7

Conclusion

Intrusion detection systems are a formidable part of the cybersecurity domain. With web applications and other attacks on the rise, there is a need to explore new, novel techniques for identifying never-before-seen 0-day attacks. We help address this challenge by introducing a machine learning-based intrusion detection model. After defining requirements, we created a holistic machine learning-based Intrusion detection system called LIDS, or LSTM-based Intrusion Detection System, which uses a distributed model for scalable network-based intrusion detection. We also incorporate an existing model, which is a logistic-regression based IDS called LRIDS, and we integrate an active learning process by integrating human experts in the process. This combined system satisfies the requirements defined at the start of this work and demonstrates a functional, network-based intrusion detection system.

Our evaluation demonstrates that this model performs better than existing systems with slightly better accuracy and a drastic false-positive improvement. This model also is deployable in a systematic fashion as opposed to existing works, where only a machine learning model exists. There is still room for improvement, and possible future work will be discussed in the next section, but this work demonstrates the capabilities of machine learning in the detection of intrusions and also the possibility for potential deployment.

7.1 Future Work

The evaluation showcased that this work has potential in real-time usage, but there are potential future enhancements that could improve performance. These possible enhancements include:

1. Generate and train on a real-time dataset: Even though this work has utilized a modern dataset (CSIC-2010) compared to most of the existing models, this dataset is focused on a limited set of attacks and is generated from a simulated environment. While this is good enough to capture a variety of attacks, this can be improved by capturing real-time traffic and labeling that manually. This generated dataset would be ideal for training the system before deployment.

2. Stress testing the scalability of the distributed system: In this work, the distributed system was tested with a simple prototype of only 3 nodes. The scalability of this system needs to be studied to understand how the whole integrated system works in a real-time environment.

3. Evaluate in a production environment: This work is deployable in a production environment where multiple packet sniffers can be set in place to capture packets as local datastores and engage in the whole process. This deployment can be done and real-time evaluation can also be done to enhance the credibility of this work.

4. Extending web application attacks to other network attacks: This work focused on web application attacks, but can be extended to detect intrusions at a network level like Snort or Zeek does.

5. Better solutions to integrating non-expert users for scalability of expert system: Non-expert users need to be appropriately trained and their inputs need to be weighted to

ensure the integrity of the system.

7.2 Concluding Remarks

Existing anomaly-based Intrusion detection systems have had various issues that prevented them from real-time usability. We address these shortcomings through our system design and we further evaluate it against the CSIC dataset. Our results indicate that our intrusion detection system can operate independently as a holistic system while maintaining an accuracy of 99.03%, a false positive rate of 0.5%, speed of 160,000 packets per second for an average system and thus outperforming existing anomaly-based Intrusion detection systems.

Bibliography

[1] TheConversation, "Cyberattacks are on the rise amid work from home – how to protect your business," 2020. [Online]. Available: https://theconversation.com/cyberattacks-are-on-the-rise-amid-work-from-home-how-to-protect-your-business-151268

[2] "Snort - Network Intrusion Detection & Prevention System," https://www.snort.org, Accessed: 2020-03-07.

[3] "The Zeek Network Security Monitor," https://zeek.org/, Accessed: 2020-03-07.

[4] UCI, "KDD Cup 1999 data," 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[5] C. Siaterlis, V. Maglaris, and P. Roris, "A novel approach for a distributed denial of service detection engine," 01 2003.

[6] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS)*, ser. BICT'15. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, p. 21–26. [Online]. Available: https://doi.org/10.4108/eai.3-12-2015.2262516

[7] L. Mohammadpour, T. C. Ling, C. S. Liew, and C. Y. Chong, "A convolutional neural network for network intrusion detection system," *Proceedings of the Asia-Pacific Advanced Network*, vol. 46, pp. 50–55, 2018.

[8] S. Pastrana, C. Torrano-Gimenez, H. T. Nguyen, and A. Orfila, "Anomalous web payload detection: Evaluating the resilience of 1-grams based classifiers," in *Intelligent Distributed Computing VIII*, D. Camacho, L. Braubach, S. Venticinque, and C. Badica, Eds. Cham: Springer International Publishing, 2015, pp. 195–200.

[9] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect web attacks using stacked auto-encoder," in *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. IEEE, 2018, pp. 131–134.

[10] H. T. Nguyen and K. Franke, "Adaptive intrusion detection system via online machine learning," in *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, 2012, pp. 271–277.

[11] ISI, "HTTP DATASET CSIC 2010," 2010. [Online]. Available: https://www.isi.csic.es/dataset/

[12] A. Ziai, "Active learning for network intrusion detection," *arXiv preprint arXiv:1904.01555*, 2019.

[13] C. Timberg, "The real story of how the internet became so vulnerable," The Washington Post, 2015. [Online]. Available: https://www.washingtonpost.com/sf/business/2015/05/30/net-of-insecurity-part-1/

[14] "RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1," Network Working Group, 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616

[15] "RFC5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Network Working Group, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280

[16] "Promiscuous Mode," Wikipedia, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Promiscuous_mode

[17] "Web Server Survey," Netcraft News, 2021. [Online]. Available: https://news.netcraft.com/archives/category/web-server-survey/

[18] J. Wilson, "Cyber-attacks on web applications up 800 per cent in H1 2020: Report," TheSafetyMag, 2020. [Online]. Available: https://www.thesafetymag.com/ca/topics/technology/cyber-attacks-on-web-applications-up-800-per-cent-in-h1-2020-report/240124

[19] OWASP, "OWASP Top 10 Web Application Security Risks," 2017. [Online]. Available: https://owasp.org/www-project-top-ten/

[20] Cloudflare, "Famous DDOS Attacks," 2020. [Online]. Available: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/

[21] J. Wu, Z. Gao, and C. Hu, "An empirical study on several classification algorithms and their improvements," in *Advances in Computation and Intelligence*, Z. Cai, Z. Li, Z. Kang, and Y. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 276–286.

[22] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016, twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916311127

[23] "Phishtank," http://phishtank.net/index.php, Accessed: 2020-03-07.

[24] "Alexa - Top sites," https://www.alexa.com/topsites, Accessed: 2020-05-12.

[25] "ActiveMQ," accessed: 2020-03-07. [Online]. Available: https://activemq.apache.org/

[26] T. S. Pham, T. H. Hoang, and V. Van Canh, "Machine learning techniques for web intrusion detection — a comparison," in *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, 2016, pp. 291–297.

[27] "Phishing Activity Trends Report - 3rd Quarter 2020," Anti-Phishing Working Group(APWG), 2020. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q3_2020.pdf

[28] I. Ion, R. Reeder, and S. Consolvo, ""...no one can hack my mind": Comparing expert and non-expert security practices," in *Proceedings of the Eleventh USENIX Conference on Usable Privacy and Security*, ser. SOUPS '15. USA: USENIX Association, 2015, p. 327–346.

[29] L. See, A. Comber, C. Salk, S. Fritz, M. van der Velde, C. Perger, C. Schill, I. McCallum, F. Kraxner, and M. Obersteiner, "Comparing the quality of crowdsourced data contributed by expert and non-experts," *PLOS ONE*, vol. 8, no. 7, pp. 1–11, 07 2013. [Online]. Available: https://doi.org/10.1371/journal.pone.0069958

[30] Ke Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," *Lecture Notes in Computer Science*, vol. 3224, 2004.

[31] M. Choraś and R. Kozik, "Machine learning techniques applied to detect cyber attacks on web applications," *Logic Journal of the IGPL*, vol. 23, no. 1, pp. 45–56, 12 2014. [Online]. Available: https://doi.org/10.1093/jigpal/jzu038

[32] "Project Jupyter," https://jupyter.org/, Accessed: 2020-03-07.

[33] "Advanced Research Computing at Virginia Tech," https://arc.vt.edu/, Accessed: 2020-03-07.

[34] "Oracle VM VirtualBox," https://www.virtualbox.org/, Accessed: 2020-05-04.

[35] A. Polyakov, "Machine Learning for CyberSecurity 101," 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-for-cybersecurity-101-7822b802790b

[36] "RedTiger's Hackit - SQLi Challenge," http://redtiger.labs.overthewire.org/, Accessed: 2020-03-07.

[37] "Postman," https://www.postman.com/, Accessed: 2020-03-07.

[38] Jastra, "Cross-Site Scripting (XSS) Makes Nearly 40% of All Cyber Attacks in 2019," PreciseSecurity, 2020. [Online]. Available: https://www.precisesecurity.com/articles/cross-site-scripting-xss-makes-nearly-40-of-all-cyber-attacks-in-2019/

[39] "XSS Game - Ma Spaghet! | PwnFunction," https://xss.pwnfunction.com/warmups/ma-spaghet/, Accessed: 2020-03-07.

[40] "OpenPhish - Phishing Intelligence," https://openphish.com/, Accessed: 2020-03-07.

[41] "OSSEC," https://www.ossec.net/, Accessed: 2020-03-07.

[42] "Suricata," https://suricata-ids.org/, Accessed: 2020-03-07.

[43] "Cisco Stealthwatch," https://www.cisco.com/c/en/us/products/security/stealthwatch/index.html, Accessed: 2020-03-07.

[44] "TippingPoint," https://tmc.tippingpoint.com/TMC/, Accessed: 2020-03-07.

[45] "Samhain," https://www.la-samhna.de/samhain/, Accessed: 2020-03-07.

[46] U. Fiore, F. Palmieri, A. Castiglione, and A. Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 12 2013.

[47] H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, K. Franke, and S. Petrović, "Enhancing the effectiveness of web application firewalls by generic feature selection," *Logic Journal of the IGPL*, vol. 21, no. 4, pp. 560–570, 2013.

[48] A. Bochem, H. Zhang, and D. Hogrefe, "Poster abstract: Streamlined anomaly detection in web requests using recurrent neural networks," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 1016–1017.

[49] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 504–509. [Online]. Available: https://doi.org/10.1145/1150402.1150459

[50] Jack W. Stokes, John C. Platt, Joseph Kravis and Michael Shilman, "ALADIN: Active Learning of Anomalies to Detect Intrusion," https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2008-24.pdf, accessed: 2020-03-14.

[51] H. T. Nguyen and K. Franke, "Adaptive intrusion detection system via online machine learning," in *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, 2012, pp. 271–277.

[52] M. Idhammad, K. Afdel, and M. Belouch, "Distributed intrusion detection system for cloud environments based on data mining techniques," *Procedia Computer Science*, vol. 127, pp. 35–41, 2018, proceedings of the First Tnternational

conference on Intelligent computing in Data Sciences, ICDS2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918301066

[53] M. N. Feroz and S. Mengel, "Phishing url detection using url ranking," in *2015 IEEE International Congress on Big Data*, 2015, pp. 635–638.

[54] J. James, Sandhya L., and C. Thomas, "Detection of phishing urls using machine learning techniques," in *2013 International Conference on Control Communication and Computing (ICCC)*, 2013, pp. 304–309.

[55] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing urls using recurrent neural networks," in *2017 APWG Symposium on Electronic Crime Research (eCrime)*, 2017, pp. 1–8.

[56] C. Opara, B. Wei, and Y. Chen, "Htmlphish: Enabling accurate phishing web page detection by applying deep learning techniques on HTML analysis," *CoRR*, vol. abs/1909.01135, 2019. [Online]. Available: http://arxiv.org/abs/1909.01135

[57] K. Tian, S. T. K. Jan, H. Hu, D. Yao, and G. Wang, "Needle in a haystack: Tracking down elite phishing domains in the wild," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 429–442. [Online]. Available: https://doi.org/10.1145/3278532.3278569

[58] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, "Hunting malicious tls certificates with deep neural networks," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 64–73. [Online]. Available: https://doi.org/10.1145/3270101.3270105

[59] "SPDY: An experimental protocol for a faster web," Chromium. [Online]. Available: http://dev.chromium.org/spdy/spdy-whitepaper