WEB APPLICATION SECURITY USING HYBRID TECHNIQUES OF ADAPTIVE CAPTCHA SYSTEM AND HONEYPOT

IGBEKELE EMMANUEL OLUFEMI
(18PGCD000018)
B.Sc (Computer Science), M.Sc (Computer Science)

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, COLLEGE OF PURE AND APPLIED SCIENCES,

LANDMARK UNIVERSITY, OMU-ARAN, NIGERIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY (Ph.D) IN COMPUTER SCIENCE.

SEPTEMBER, 2021

DECLARATION

I, **Emmanuel Olufemi IGBEKELE**, a Ph.D. student in the Department of Computer Science, Landmark University, Omu-Aran, hereby declare that this thesis titled "**Web Application Security using Hybrid Techniques of Adaptive CAPTCHA system and Honeypot**" submitted by me is based on my original work. Any material(s) obtained from other sources or work done by any other persons or institutions have been duly acknowledged.

IGBEKELE, EMMANUEL OLUFEMI (18PGCD000018)

………………………………….

Signature and Date

CERTIFICATION

This is to certify that this thesis has been read and approved as meeting the requirements of the Department of Computer Science, Landmark University, Omu-Aran, Nigeria, for the Award of Ph.D. degree.

Prof. A.A. Adebiyi Supervisor	Date
Dr. (Mrs.) M.O. Adebiyi (Co- Supervisor)	Date
Dr. (Mrs.) M.O. Adebiyi (Head of Department)	Date
(External Examiner)	Date

ABSTRACT

Web Services have become a trend in providing solutions to mundane and recurrent tasks. This development, however comes with the bottleneck of authenticity and intent of users of such services bringing about the advent of several Intrusion Detection Systems (IDS) as well as Intrusion Prevention Systems (IPS). Individually, these mechanisms have been found to be porous in its defense mechanism. Some of these IDS/IPS have at one time or the other been layered to further strengthen the grip of security against bot and spam attacks, yet, the problem lingers. Hence, this study seeks to harness the strength of two distinct IDS/IPS in a hybridized solution to reduce the menace of web application security. The objectives of this work are to identify the various compatible IDS/IPS that can be hybridized, design an improved hybridized web application security framework, implement the hybridized adaptive model in a web application by computer simulation and then evaluate the framework based on selected performance metrics of accuracy and usability.

The method engaged are SSH protocol, Diffie-Hellman key-exchange algorithm, Hidden Markov Models and Jess rules for integrating adaptive CAPTCHA and Honeypot to solidify the security of an internally developed web application after which Think-Aloud activity alongside Thematic Analysis were used for system evaluation.

Experimental results showed that both CAPTCHA and Honeypot can be layered over each other to produce a very high performance in terms of execution time, resulting in a robust and secure web application. The hybrid model was also found to scale linearly with increase in number of service alternatives. In performing the computer simulation experiment, Themes were formed and established which helped in determining the usability and accuracy component of the hybrid solution. Most prominent amongst the themes was found to be the issue of CAPTCHA puzzle solving where most experimental users would rather not have their details incorporated into CAPTCHAs. Also, the hybrid defense technique outperformed some of the other existing individual and hybrid techniques with a 93% accuracy thereby significantly improving web application security.

The improved hybridized model for web application security provided in this study is capable of enhancing security when deployed. It is therefore recommended for deployment in the industry.

DEDICATION

This thesis is dedicated to God Almighty, for the productive life given me, my very present help, who has successfully brought me to the climax of this phase of my sojourn in Landmark University as a Postgraduate student. To Him alone be all the glory and honour.

ACKNOWLEDGEMENT

Carrying out a research work and writing a thesis is an arduous task, and although the student's name is the one appearing on the cover, it would not have been a success without the efforts and contributions of numerous individuals.

I will like to profusely thank my Supervisor, Prof. Ayodele A. Adebiyi who also doubles as the Dean, College of Pure and Applied Sciences, and my Co-Supervisor, who also doubles as the Head of Department of Computer Science – Dr Marion O. Adebiyi for her motherly advice anytime I speak with her, their consistent and moral support is very much appreciated. Their vast knowledge, consistent reminders and check-up on the progress of the work, coupled with the constructive criticisms has refined this thesis and ensured it came out fine. It has been an honour and a pleasure to have experienced the privilege to be imparted by leading intellectuals, who have exposed me to the basics of computer science, critical analysis of scientific issues, and the art of neat technical writing. A special appreciation goes to Professor Okeyinka, whose fatherly advice and constant jolt has kept me on my toes to run with this research work to the end. Other members of staff of the Computer Science Department, Landmark University, specifically Dr. Gbadamosi Babatunde and Mr. Asani Emmanuel, I say a big thank you to them.

Special thanks to Landmark University for their financial sustenance and access to research resources.

My immense gratitude goes to my parents, Mr. and Mrs. Igbekele. My siblings, Ms. Isaiah Ayobami Igbekele, Mrs. Phoebe Titilope Odetola, Mrs. Lydia Opeoluwa Igbekele Olaleye, Miss. Temitayo Esther Igbekele. My in-law, Mr. Oladipo Olaleye for their moral, financial and spiritual support all through these years. May God's blessings continually hover on them all.

Special mention goes to my Boss, Mr. Matiluko Opeyemi, who has been of immense help and support anytime I seem to be stuck at a certain stage of the research work. May light continually shine upon your path and generations to come. Deep appreciation also to Kolawole Rachael Oluwatosin. Thank you for always spurring me and your absolute belief in me. I am deeply grateful and blessed to be connected to you.

My utmost gratitude goes to my Mentors, Dr. David Oyedepo, Dr. David Abioye and Pastor Abraham Ojeme, all of the Living Faith Church worldwide, their life-changing teachings, instructions and books have made tremendous impact in my life and I cannot but be grateful to God for divinely ordering my steps and guiding my path into destiny.

Very deep appreciation to my fiancée, Miss Okunlola Faith, thanks for always being supportive and an available shoulder to lean on. I am immensely blessed to have you.

I profoundly appreciate all researchers, authours, originators, theorists and all great individuals who have remarkably made profound and valuable input to research in the field of Web Application Security.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.0 Background to the Study

Crime has been an integral part of life right from creation and it has been passed along from generation to generation. A crime is an act committed against a set law which is deemed punishable by governing bodies of a constituted authority (Kaur, 2018). As the years rolled by, crime increased sporadically. With the advent of Cloud Computing, things have taken a dramatic turn. Cloud Computing is a ubiquitous and promptly available access to a configurable and limitless pool of resources for computation that are made available and used by a limited number of users and/or providers involvement (Ezenwoke and Igbekele, 2019). Thousands of years ago, there was nothing like Cybercrime. However, today, cybercrime goes side-by-side with physical crime. What is even more fascinating is that, in a few years, Cybercrime will overtake physical crime. Cybercrime is a social crime perpetrated by using technology, Internet and all its related constructs. Cybercrime keeps thriving today, due to the fact that crime and atrocities can be committed without anyone being physically present at the crime-scene (Ba, 2017).

The more technology keep advancing towards Internet of Everything, the more cybercrime will continue to thrive, knowing that investigation into such crimes is becoming more and more complicated to carry out. As efforts are being made to cover up a loose end, more and more gaps are being discovered. Availability of Internet (uncontrolled, unrestricted) has further spurred this demeaning act. One sure way to put an abrupt end to this debacle is to shut down the Internet, but that option seems like throwing the world back into the dark ages. The Internet and indeed the Web has gone beyond shutting down, hence the only option will be to resort to available control measures.

The Web started out as an idea to simplify communication and make life much more convenient through automated processes, however, the menace on web platform today has gone beyond what the initial conception of the idea was. Today, the Web hybridizes virtually all processes that has to do with our daily lives. We live, learn, shop and communicate over the web. Redundant processes which takes time to conclude can now be completed in milliseconds over the web (Choudhury, 2009).

Applications ranging from virtually every sector - Banking, Educational, Commerce, Sports – are now concluded through rapid processes over the Web. However, with these technological advances also comes the bottleneck of security. The more our world (day-to-day lives) is exposed to the web, the more vulnerable life become to different forms of attack. Virtually all web services engaged in this modern days requires a form of registration, login, purchases, payments amongst other services thereby making user and customer details available on thousands of servers all around the world.

The availability of these humongous data flying all over the web has created an attraction for different categories of unscrupulous identities over the Internet. These individuals have been identified under different names and categories (Khu-Smith, 2003). Unlike real life scenarios where an individual can be apprehended when a crime is being perpetrated, the Web is different in its entirety as one cannot even tell if the perpetrator of a crime is human or a robot (a computer bot).

A computer bot is an autonomous program that runs over a network thereby communicating with servers, users or systems in the similitude of humans. i.e such programs acts, behave and perform as humans.

Web applications are rife with bot cases where individuals and parties depends absolutely on computation metrics and data storage being processed on remote servers, by unknown systems with unknown interests (Smith, 2001).

Effective infrastructures that provides critical services are the target of these attacks, having an impact on several people and by implication, major consequences for organizational operation. There are trends of multiple cyber-attacks experienced by nations and states across the world: Recently, in February 2018, there was an accusation on Russian military by many nations claiming they were responsible for NotPetya, an attack recorded in June 2017 that brought the entire Ukranian economy to a halt (Higgins, 2018).

Over the past two decades, focus of development and research has shifted to Security of lives, identities and properties of web users. The quest of both the industry and academia is to find a lasting solution to the lingering problem of security as it relates to web applications.

In addressing this problem, a Completely Automated Public Turing Test to Tell Computers and Humans Apart (commonly known as CAPTCHA) has since been developed to succinctly separate computer bots from human users (Abdullah, 2016). CAPTCHA, on its own, is a class of Human Interaction Proofs (HIP) designed to prevent Web-bots from ever accessing Web services. CAPTCHA is a reverse Turing test based on audio, text or image based challenge systems. Several interactions over web applications assume services over the Internet use CAPTCHA techniques to effectively differentiate a human from a Web-bot (Banday and Shah, 2011).

Over the years, several CAPTCHAs have been developed and can be succinctly classified into two broad categories – Optical Character Recognition (OCR) and Non-Optical Character Recognition (Non-OCR). The OCR is based on Text while the focus of Non-OCR is on Multimedia. The problem however is that CAPTCHAs in their varying forms have been accessible by new bot programs. An instance, letter-segmentation mechanism has been found to break text CAPTCHAs (Abdullah, 2016).

Honeypot is a smart method proposed by Phil Haack in 2007 which harnesses the power of invisibility in web-forms to deceive bot-programs. Considering the fact that computer bots processes raw HTML rather than rendering the source code, it becomes imperative that additional form fields are introduced into web forms using HTML and hidden with Cascading Style Sheets thereby going undetected by bot programs (Moradi and Keyvanpour, 2015).

Independently, in a way, Intrusion Detection Systems (IDS) have shown vulnerability to manipulations and attacks and both Honeypots and it is no exception with CAPTCHAs. For CAPTCHAs recently, mechanisms such as Solving Attacks (Hernández-Castro, R-Moreno, Barrero, and Gibson, 2017), Bypass Attacks (Bursztein, Bethard, Fabry, Mitchell, and Jurafsky, 2010) and Human Exploitation Attacks (von Han, Blum, and Langford, 2013) have been successfully launched and these has paved a way for bot programs to access into sensitive data. Honeypots have not been totally secured, as oftentimes, not all fields of web forms are mandatory fields (Nkwetta, 2018). This therefore leaves a gap as to the security of Web applications in modern times.

Hence, this work looks to harness the power and attributes of two distinct IDS and IPS, hybridizing them into a single solution that further strengthens the security of Web Applications and indeed activities carried out over the Internet.

1. 2 Statement of the Problem

The world is rapidly evolving and technological advances keeps bringing out solutions to transient problems in web application security issues. Web-forms, transactions, accounts, profiles, payments and all other activities on the web continues to increase at a galloping rate and the quest for security measures and data privacy remains an issue on these platforms with research work gearing towards a safer clime for web applications..

Over the years, the sole aim of CAPTCHA (easy for a human but hard for a robot) has been outsourced by some organizations and establishments to unskilled labour pools. This act has given attackers even more insight into the workability of CAPTCHAs by divulging the solution mechanisms to human solvers. With this knowledge available to the general public, it has become increasingly more difficult to limit the activities of hackers when it comes to web security (Zhang, Hei, and Wang, 2019).

Similarly, in a bid to deepen the continuously widening difference between human and bots as regards the strength of the existing CAPTCHA security mechanisms, newly adapted theories are being presented, with a closer attention to the responses of users as partial credit. The underlining factor subsists and that is the fact that CAPTCHAs have a singular response: correct or incorrect, true or false. This has brought about the inclusion of the Partial Credit Algorithm (PCA) as a synchrony in interaction between computers and users is much more important than a true or false/correct or incorrect response while looking at the validity of the user information that was supplied. This has led to cracking of so many

CAPTCHAs and therefore significantly highlights the underlying development of hollow and 3D CAPTCHAs (Yu and Darling, 2019).

Psychology suggests that the volume of information available to the public makes it practically impossible to treat all information equally and with equal attention too. More often than not, selective information which aligns with our interests and beliefs tends to be our focus while information that seems to be irrelevant or appears irrelevant to the dominant cognitive needs is overlooked. This tendency enhances solutions tailored to a particular need and ignoring other loop holes (Chen, Luo, Liu, Wang, and Ma, 2019).

Just as there are so many independent options for web application security, so also, there are many possible solutions through hybridizations. Two profound bottlenecks comes into fore when trying to hybridize two explicitly different protocols:

- Selection issues: The choice of which security mechanism to choose amongst the several options.
- Compatibility issues: Implementation of the integration process.

Each of the mechanisms mentioned above have shown to be advantageous in certain environments while showing weaknesses in other environments. Hence, the choice of which one to layer (hybridize and apply) given the viciousness of the web environment and bots without prior knowledge about performance (Omid, Gary, Large, and David, 2017). The usefulness of an AI problem as regards security largely depends on an automated way of generating the problem instances together with their solution. *The security of CAPTCHAs should not be in the secrecy of a piece of code or that of a database*. (Yu and Darling, 2019). This, amongst others, is the problem being proposed to study.

In various frameworks usually known for recognition of different forms of CAPTCHAs, recognition network has been found to be a pivotal step towards the success rate of CAPTCHA solutions. However, these days, recognition networks have mostly focused on deep learning as applicable in object detection, image classification, image steganography among others (Chen et al., 2019).

This Thesis therefore focuses on creating an adaptive puzzle-based CAPTCHA and layer it with an adaptive Honeypot CAPTCHA and see how the hybridized security performs against existing similar individual mechanisms. The study sets probabilities for the layered mechanism and in the process increase or decrease the modalities being chosen during the next attempt and evaluation attempt based on its performance.

The ingenuity of this idea capitalizes on the information flow in public between humans. Without prior information, attackers will not be able to successfully complete their attack peradventure, by any chance, they are able to break the security layer of the CAPTCHA.

1. 3 Motivation of the Study

As each of the individual CAPTCHA and Honeypot solutions becomes more and more difficult for bots, attackers will easily shift focus towards the use of Human Exploitation Attacks (human CAPTCHA solvers). It is believed that, amongst others, the most notable challenge of the future remains stopping humans from doing the dirty work on behalf of bots. Mohammed (2014) postulates that addressing this menace will require developing CAPTCHAs with the ability to ascertain motives and intent of users to determine the genuineness of interaction rather than just differentiating if an interaction is being initiated by a human user of a computer program (Al-Fannah, 2018).

Mohammed (2014) suggested that the concept of layering is better than sole dependence on a single layer of protection (depth of defense principle). For this reason, dependence and sole reliance on a CAPTCHA puzzle as the basic means of preventing bot interactions is not ideally an effective approach. Bots are generally programmed to successfully attack a specific type of CAPTCHA and, for this reason, a web application that uses such a CAPTCHA or any other single technique as its only defense will immediately be susceptible to attacks by such bot. Past experiences suggests that the development of supplementary safeguards against web application intrusions has become a vitally important weapon in the fight (Al-Fannah, 2018).

1. 4 Aim and Objectives of the Study

The aim of this thesis is to develop an improved web application security framework using a hybrid technique of adaptive CAPTCHA and Honeypot. The objectives are to;

1. identify from literature various Intrusion Detection Systems (IDS) solutions and hybrid models.
2. design an improved web application security model using SSH protocol, Diffie-Hellman key-exchange algorithm, Hidden Markov Models and Jess rules for integrating adaptive CAPTCHA and Honeypot.
3. implement the hybridized adaptive model in a web application by computer simulation.
4. evaluate the framework based on performance metrics: Accuracy and Usability.

1. 5 Research Questions

There have been numerous independent options for web application security, however, there are relatively few and "untested" solutions through hybridizations, which brings to fore the following research questions:

i. ***Question 1:*** *Is it possible to improve Web Application security by hybridizing protocols to fend off intrusions from bots?*

ii. ***Question 2:*** *Would the proposed Web Application security solution outperform the existing web security solutions highlighted in the literature?*

1. 6 Scope of the Study

This research work addresses the problem of internet theft, fraud, security and privacy experienced in web application technologies and services using a hybridized layered security framework. Basically, this implementation of the work is focused on an existing web application currently in use in Landmark University. This work leverages on the shortfall of the CAPTCHA and Honeypot protocols independently to implement a hybridized layered security framework.

1. 7 Significance of the Study

Cybercrime is now a global epidemic. Suggestions of several kinds of research in preventing and detecting intrusion detections are eminent. There is a need for proper and better secure techniques for web application usage, to help in reducing the criminal activities over web applications. This study provides an enhanced hybridized adaptive framework leveraging on existing independent protocols. It is believed that this hybridized protocol will help increase user privacy protection level and also address some security shortfall of individual existing protocols, thereby attempting to increase consumer trust level in web applications. This study will further avail e-commerce sites, online banking applications, e-payments and so on. The privilege of carrying out their operations with the assurance of the safety of every transaction conducted over their web applications.

1. 8 Organization of the Chapters

The chapters of this thesis are organized as follows:

Chapter One contains the background of the study, statement of the problem, justification of the study, aim and objectives, research questions, the scope of the study, the significance of the study, and contribution to knowledge. The remaining sections of this study are prepared as follows:

Chapter Two: The literature review, assesses studies that have been carried out in the aspect of CAPTCHA, Honeypot, Intrusion Detection Systems, Intrusion Prevention Systems, Hybrids and Layered techniques, and related works.

Chapter Three: The methodology gives a detailed description of the existing approach, the proposed model, dataset, research design, the layout, and the performance evaluation metrics including some implementation screenshots.

Chapter Four. The results and findings of the study are discussed. First, the proposed hybrid technique is evaluated, and the result compared with existing methods. The findings are then discussed.

Chapter Five: The Conclusion and Recommendation, captures the completion of the study by giving a transitory summary of the work done in this thesis with its discoveries and supplementary research tips.

1. 9 Definition of Operational Terms

Crime: An act committed against a well-defined law and having a punishable repercussion to violators

Web Application: An automated software that enhances processes for speed of execution.

Security Mechanism: A recognized defense protocol that safeguards a web application.

CAPTCHA: Completely Automated Public Turing to Test Computers and Human Apart, a software that acts as a security mechanism to tighten the security of web applications against intrusion.

OCR: Optical Character Recognition, a term used to indicate acceptance and comprehension based on visual contact.

Puzzle CAPTCH: This is a CAPTCHA variance that involves Image segmentation and arrangement to solve.

Honeypot: This is a CAPTCHA variance that tends to deceive computer programs using decoy HTML fields.

Layered: Having two or more items placed on top of each other.

Hybridize: Combination of two (2) or more independent entities.

Secure Shell: A protocol that harnesses different web application algorithms like RSA, AES, Diffie-Hellman, amongst others in enhancing security.

Cyber Attack: A cyberattack is any maneuvering offensively done with the aim to target computer networks, information systems or personal computer devices.

Cloud Computing: Cloud computing are on-demand computer system resource facilities available to users without directly managing the infrastructure.

Cybercrime: Cybercrime is any form of crime that involves a computer and a network.

Intrusion Detection Systems: Devices or software applications that allows for monitoring a system or network for policy violations or malicious activity.

Optical Character Recognition: Optical character recognition (OCR) is a technology that facilitates business solutions used in automating extraction of data from printed or written text before converting the text into a format that is machine-readable for data processing such as editing or searching.

Encryption: A process involving the conversion of information or data to codes with the sole aim of preventing unauthorized access.

CHAPTER TWO

LITERATURE REVIEW

This section reviews the literature and related works on Human Interaction Proofs, Intrusion Detection and Prevention Systems, CAPTCHA, Honeypot and some hybrid techniques.

2.1 Theoretical Background

This section presents reviews of the literature focusing on discussion on cybercrime, the types, a few reasons behind cybercrime and why it thrives. Also, a critical look is done in the area of web application security as regards CAPTCHA – Optical Character Recognition (OCR) and Non-Optical Character Recognition (Non-OCR), defense approaches, Honeypot and a few other web application defense mechanisms.

2.1.1 Cybercrime

Cybercrime is a social crime that engages the technological tools in its perpetration. It is a social crime that is thriving and ravaging the world as we see it today. Although still new (and gaining more ground daily), the impact is as gruesome (if not more) than conventional crime. As early as 1820, there was a record of a group of employees of an Organization (Joseph-Marie Jacquard) who tried to sabotage an invention made by the Organization in order to protect their job, however, this is a bit different from Cybercrime as we know it today (Ba, 2017).

Cyber criminals keep evolving with sophisticated ways of targeting their individuals as well as public and private parastatals. The rate at which Cybercrime is thriving compared to the way the Cyber security is thriving are not in any way commensurate. Cybercrime, in all its forms today consists of both the computer and humans (Ibekwe, 2015). Nations and Government are now having laws on cyber securities in order to protect data and prevent its loss. Sometime ago, it is understood that the Liberian Government had to shut down the Internet amidst anti-Government protests.

Kaur (2018) defined cybercrime as an unlawful act in which the computer is both a tool and a target. He further defined the kinds of cybercriminals in the following categories:

1. **Crackers:** Causing damage for fun or to satisfy some anti-social motives e.g Virus creators.
2. **Hackers:** Explore another's computer system as competition, education or curiosity.
3. **Pranksters:** They perpetrate tricks for temporary harm.
4. **Career-Criminals:** Earn their income from damaging other people's contents.
5. **Cyber-Terrorists:** Group of like-minded individuals that crash websites through traffic.
6. **Cyber-Bulls:** Harassment over the Web, vicious posts amongst others.

The anonymous nature of the Internet makes it the perfect platform for committing various atrocities. This has led to categorization of Cybercrime into three (3) broad categories:

Cybercrime against Organization: This includes Hacking, Denial attack, Virus attack, Mail bomb and so on.

Cybercrime against Individual: This includes Email spoofing, Phishing, Spamming, Cyber-defamation, Cyber-stalking, Computer-sabotage, Malware and so on.

Cybercrime against Property: This includes Intellectual property crime, Cyber-squatting, Cyber-vandalism, Hacking system, Trojan horse, Logic bomb and so on (Kaur, 2018).

2.1.2 CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Human Apart)

Completely Automated Public Turing Test to Tell Computers and Humans Apart (commonly known as CAPTCHA) has been a web application security mechanism for close to two decades now. The tool has been a well-known effective mechanism in tackling the problem of security as regards web application. But as with all technological advancement made in the 21st century, the more technologically developed we are as a universe, the more vulnerable we become to security threats and intrusions. So many modifications have been done ever since the introduction of CAPTCHAs and some of them will be reviewed in this section.

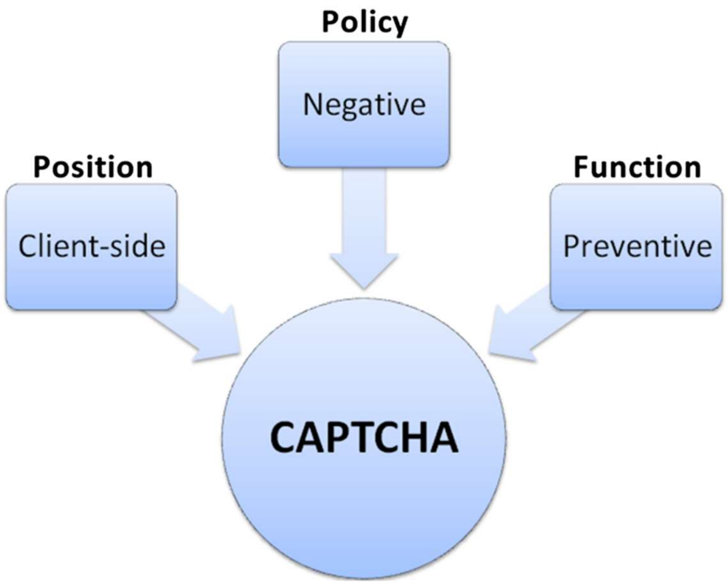


Figure 2.1: Features of CAPTCHA as a security mechanism (Source: Mohammad, 2014)

Figure 2.1 shows the essential working briefs of a CAPTCHA system. This particular modules checks for the enablement of Javascript and cookies on the browser. Cookies, or as they are more often called, HTTP cookies, are bits of text-file data stored on a browser. Most websites use those tiny bits of data to customize users and enable user-specific features. Core website functionalities are enabled on those platforms such as e-commerce, shopping carts, and they are also used for more controversial purposes, such as tracking user activity. Cookies are an integral part of the way the web works as well as a privacy concern and security risks. In view of this, casual users of the web and developers have good and valid reason to better understand how these tiny bits of data work. In addition, a duration for filling out the form can be explicitly set. When this is done, submissions that take longer period or appear too fast can be refused.

The idea behind visual CAPTCHAs is to pose a challenge to users of a system based on what they are able to see, recognize and identify. It is believed that there is a form of communication through visuals between users of a system and the system itself, hence the reason for this form of security mechanism. A few of the Visual CAPTCHAs are discussed as follows.

2.1.2.1 Optical Character Recognition (OCR) CAPTCHAs

OCR-based CAPTCHAs are the most accepted widely-used form of CAPTCHAs. Due to the simplicity of the variant, most web applications have adopted this CAPTCHA. Basically, OCR-based CAPTCHA employs the use of texts in its implementation. These CAPTCHA implementations largely challenges automatic character recognition programs that are readily accessed by human users. These CAPTCHAs are presented in various forms to the users. They could come in distorted, rotated, deformed and so on all in a bid to make it difficult for computer bots to decode (Nguyen, 2014). Examples of Text-based CAPTCHAs includes BaffleText, Gimpy, Ez-Gimpy, ScatterType, Pessimal Print, amongst others. A recent development in CAPTCHAs have them displayed in ASCII art format. In this format, texts are rendered in ASCII art characters. An example of this is shown in Figure 2.3 showing the ASCII presentation format for CAPTCHA.

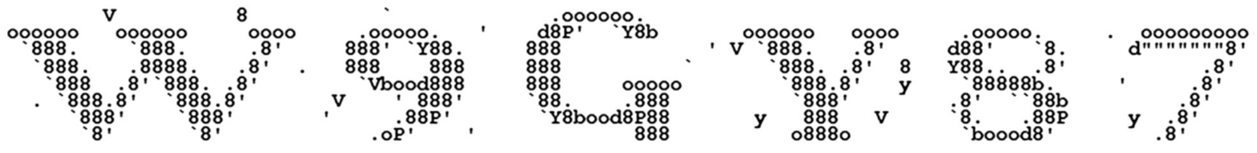


Figure 2.2: ASCII art Character format for CAPTCHAs (Source: Mohammed, 2014)

2.1.2.1.1 Image-depicted CAPTCHAs

As a build up from the OCR-based CAPTCHAs, images were also introduced to increase the difficulty of machines and bot programs to decode and break. As a matter of necessity, this particular CAPTCHA capitalizes on the weakness of Computer's vision in dealing with images. For this CAPTCHA, the essence of its use is the ability of humans to extract and describe more from a picture than can be automatically generated by an external program. Several designs for Image-depicted CAPTCHAs have surfaced over the years ranging from Asirra, ARTiFACIAL, Imagination, Bongo, ESP-Pix, amongst others (Anil, Naveli, and Bhukya, 2018). Majority of these Image-based CAPTCHAs are rendered in 2D but since new bot programs were able to break through these 2D Image-based CAPTCHAs, 3D Images were introduced. It is widely agreed that Humans can decode and interpret 3D images better than computers can even attempt, and upon this premise 3D images subsist (Woo et al., 2014). An example of a 3D image-recognition CAPTCHA is depicted in Figure 2.3 showing a 3D interactive CAPTCHA.

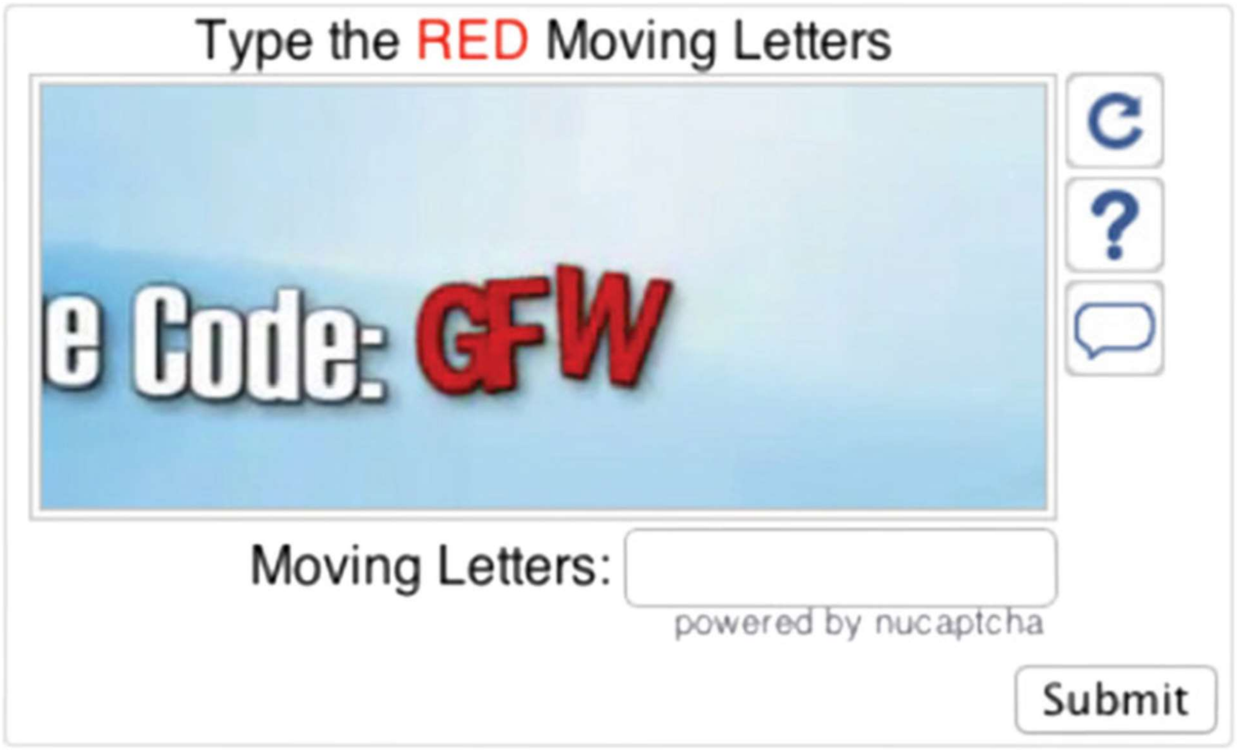


Figure 2.3: An example of a 3D interactive CAPTCHA (Source: Winter-Hjelm, 2009)

2.1.2.1.2 Moving-Objects CAPTCHAs

The emergence of the Image-based CAPTCHA and the incorporation of the 3D version birthed the moving-object variety of CAPTCHA. This variety of CAPTCHA focuses on Videos and Animations and the ability of humans to watch a short clip and make deductions based on that. Naturally, this security mechanism seems appropriate and secure, however the cost of implementation and its usability makes it less attractive (Kluever, 2008).

2.1.2.1.3 Interactive CAPTCHAs

Interactive CAPTCHAs came about because the system intends to engage the user more as opposed to the norm. This higher level of user interaction provides a more secure measure in web applications. However, contrary to what the name depicts, users do not interact extensively with the protocol except to answer a question by giving a brief description of an item or word as perceived from a text, image or video. This form of CAPTCHA is very cumbersome for bots to break, humans on the other hand have no trouble proffering a solution (except for human users with disabilities). The success of this form of CAPTCHA is largely dependent on the inability of computer programs to successfully imitate human behaviours especially in the areas of motion (Abdullah, 2016).

2.1.2.2 Non-Optical Character Recognition (Non-OCR) CAPTCHAs

As with every form of technological advancement, identifiable setbacks are bound to occur in one or two cases, most prominent being humans with disabilities such as Vision Impairment (for visual CAPTCHAs) and a few others, bringing about the advent of non-visual CAPTCHAs. These CAPTCHAs thrive maximally on semantic tests as well as speech-recognition to determine who exactly the users of a system are. A few of the Non-visual CAPTCHAs are discussed as follows:

2.1.2.2.1 Semantic CAPTCHAs

Semantic CAPTCHAs boasts the most secured form of CAPTCHAs amongst the numerous currently in use. It achieves this feat based on the fact that it has gone far beyond the capabilities human-oriented questions being answered by bot programs or machines. These kind of CAPTCHAs provides the user with questions that requires a little nudge and juggle of the brain to answer and in record time too. Questions such as “A pupil learns in a building referred to as?”, “The Father is the head of the what?” and so on are some of the ways Semantic CAPTCHAs are structured. However, there is the possibility of questions having numerous answers being asked thereby making these kind of CAPTCHAs have multi-answer solutions, often times,. An example of such question can be “An example of a top management staff in a University is?”. Although for now, this CAPTCHA remains hard to break down for most computer bots, its high cost of implementation is one of the reason it has not gained much ground as regards web application security implementation.

2.1.2.2.2 Audio CAPTCHAs

The advent of OCR CAPTCHAs was a big relief to Web Application security until the issue of users with visual impairments and disabilities surfaced. Efforts were made to ensure a parallel was drawn and that brought about the advent of Audio-based CAPTCHAs. These CAPTCHAs mechanism allows a short voice recording to be played and such will be typed by a user as a word or sentence spoken in the audio file. Recently, there has been a new development to this kind of CAPTCHA. Users are now expected to repeat a word or sentence as it is spoken in the audio file. When users do respond, the response checks to confirm if that was submitted by a human or a speech-synthesis system. The drawback to this has a lot to do with language barrier and intonation of the user. Oftentimes, a human-user tries to pronounce a word or sentence and the system returns a failed attempt (Premanand, Meiappane and Arulalan, 2015).

2.1.2.3 CAPTCHA Hybrid Variances

The sole aim of implementing and incorporating CAPTCHAs is to secure online services from abuse by bots and other fictitious programs under the disguise as human users. In our world today, online services such as e-marketing, e-banking, public networking, e-Transportation, mobile interactions and general web surfing, fictitious programs attack incessantly at the expense of legitimate and honest interactions.

CAPTCHA, as used in many applications seeks to prevent comment spam in blogs. Synonymously, it also seeks to create and ensure that only actual human users can create free email accounts. This definitely is one sure way to prevent those unwanted spammers in online registration from creating a deluge of email addresses. Also, the provision of proper authentication test to ascertain that the interactions being carried out are actually by human users in any online polling system. It thrives on actual human interactions; through which dictionary attacks are disabled from exhaustive password search and intentional locks (Subramanyam and Priya, 2018).

Since its introduction into web forms, a singular check (one CAPTCHA module) has always done the work, however, bots are also getting more intelligent by the day and within a short time, it was discovered that even bots now solves some CAPTCHAs. The solution percentage kept increasing and this has been the singular motivation behind CAPTCHA hybridization. A few of those hybridization projects are looked at in the next segment.

2.1.2.3.1 Multi CAPTCHA

(Manzoor and Soumya, 2014) in their work, argued that Multi-CAPTCHA requires a human user to solve a CAPTCHA test via multiple user interactions. The back-and-forth interactions between client and server further amplifies the statistical time-difference between a valid user and a human solver, which improves performance in attack detection. Multi-CAPTCHA uses a sequence of mouse-clicks to grant a user interactive access to solve a CAPTCHA challenge. The way the process works is that a normal CAPTCHA image is systematically generated and displayed first. Afterwards, the user clicks on the CAPTCHA image to begin the input sequence expected by the multi-CAPTCHA. When there is a click on the CAPTCHA image, multiple buttons with garbled characters appear below the CAPTCHA image. With these character-buttons displayed, a user must then click on the button corresponding to the first character or pattern in the CAPTCHA image. With each click, new sets of buttons are rendered. This goes on and on continuously until there has been a click for each character of the CAPTCHA image. A major limitation of Multi-CAPTCHA is the time-taken to solve it. Allowing actual users to take as much time as needed to decode the image first before starting the multi-step challenge/response sequence can be demanding.

2.1.2.3.2 Two-Step CAPTCHA

Manzoor and Soumya, (2014) in their work, presented the idea of blocking automated spambot attacks, which allows users to be engaged in a two-step process of authentication. The first step has the user given a set of images that is a solution to a question associated with this step to which the user must recognize. The next step which is a follow up of the previous step exhibits entering of some values which are closely linked with the selected image in the first step in order to further prevent the probability of a bot attack. This test is used to differentiate a machine‘s ability to show intelligence which directly equates to, or inseparable from what a human will do. Three distinct participants are involved in the test – two entities (a human and a computer, both are not visible to the judge) and judge. The two entities largely communicate with the judge solely through text channels. The judge acts as a verifier of which text channel differentiating both correspondence – which is human and which is computer. In today’s world, virtually all websites hold sensitive data, often, they require user-signup at times, while some websites are solely for conducting polls etc.

In the first step, a question pops up alongside a 3x3 grid of images. Users are then required to simply click on the image which depicts the answer to the question. At any click on an image by the user, alphanumeric values corresponding to the image chosen is displayed to the user. The user is then saddled with the responsibility of selecting the four values (top, bottom, and left, right) in the given drop down menus. After a successful entry, the user submits the selections. A successful authentication occurs only if the image chosen is correct as well the all the four corresponding values entered are correct. A major limitation of this CAPTCHA is the possibility of a breakdown in the authenticating machine into websites is considerably low. A very simple and less time consuming approach.

2.1.2.3.3 Super CAPTCHA

This CAPTCHA works based on a series of mouse clicks that allows a user interactively solve captcha questions. On the server side, session data is stored about the indices of both correct responses and user clicks. At the completion of the input sequence, there is a comparison of correct input sequence and the user-clicked index sequence. If a match exists, CAPTCHA has been properly solved by the user. The proposed approach is an embodiment of three distinct elements. A set of objects, distortions are added to confuse the spambot, and this increases the false positive detection rate. For each objects, there are questions that are assigned and this invariably makes the detection easy for the users. This system thrives on combing image-based object recognition techniques with knowledge-based component. Users of the system are asked to identify selected objects within a scene and produce an evaluated response that hinges on certain level of human tolerance to ascertain the authenticity of the user. This system is considerably difficult for spambots. The developed system was found to be user-friendly as it affords users the possibility of solving the posed challenge with just a few clicks. SUPER CAPTCHA can be used in various forms for security purpose and it adds additional complexity to the system and provides ease of access to human and improves human accuracy rate and also lower spambot attacks rate.

2.1.3 Honeypots

Honeypots on the other hand are a class of intrusion detection mechanism that helps detect unscrupulous activities within a system. In modern times, Organizations are much more interested in threat detection and control rather than its prevention.

The founder of the Honeynet Project who also is a topmost expert in security, Lance Spitzner defines honeypots as “a high-value resource in security whose value is succinctly determined by an attack, a probe or a compromise”. Honeypots capitalizes on deception in order to combat attackers (Higgins, 2018).

Honeypots are sacrificial computer system that is intended to serve as a decoy for cyberattacks. It mimics a target for hackers using their intrusion attempts to gain appropriate information about cybercriminals and their mode of operation. Also, it serves to distract them from other targets. The concept allows addition of a visually hidden form element, and subtly refuses submission if form field is not empty. Consequently, a time-frame can be set for filling out the form. Submissions that come in in microseconds are usually condemned to spam attacks and are blatantly refused.

In an understandable way, honeypots are solutions which camouflages as authentic systems in order to attract, detect, track and analyze behavioural patterns when the system is accessed by a user in an illegal manner. Honeypots are categorized as an active web application defense mechanism, deployed with the sole aim that the system will be attacked. One of the major attributes of a successful honeypot is attractiveness to a hacker. “Attractive” here simply portrays the honeypot appears to be the sole target device for which the attacker has approached the system: A setup with a distinct ability to be exploited, and in the process, proffer the maximum value to the hacker (Higgins, 2018).

Essentially, the major advantage of Honeypots lies in their ability to capture sensitive information as well as defending the system within which they are being deployed. Honeypots are deployed under two (2) major categories, as decoys and as sensors.

2.1.3.1 Honeypots as Sensors

Considering the fact that valuable data about the attacks can be collected by these honeypots, they are therefore seen as a tool to receive, and are therefore commonly used as sensors. In particular, they are useful for detecting vulnerabilities and weaknesses in web application designs, since the captured data packets are available for scrutiny in understanding the behavioural strategies of the attackers, and their motivations as well.

2.1.3.2 Honeypots as Decoys

Honeypots are used also in diverting the attention of an attacker away from major valuable system components in which they are being deployed. The effectiveness of this strategy lies solely in the conviction to the attacker that what is being expected is what he is providing the system. That is, the input being supplied by the attacker is what the system requires (Nassar and Miller, 2013). Hence, when such field are triggered, the system notifies the Administrators of the presence of an intruder thereby providing profound secure mechanism against such intended attack (Divyashree, 2018).

Honeypots have become an essential part of frameworks for network security, whilst also experiencing a few setbacks. Amongst the many setbacks tied to the deployment of honeypots for any establishment is the effort and time its maintenance, monitoring and deployment consumes. This has made researchers come to the conclusion that the efforts that goes into designing and implementation of honeypots are best diverted into guarding a protection system. However, the main aim of solutions such as HoneyBOT, Modern

Honeypot Network and Honeydrive is to simplify the process of developing honeypots, either on a small or an enterprise network.

Threatstream, a cybersecurity company has since developed a software known as Modern Honey Network, which is an open-source honeypot software. The development of MHN seeks to ensure a simplified process of deploying and managing honeypots propelling organizations to see it as a security utilization tool for easy adoption. The setting up and monitoring of the MHN honeypots is automatically processed. It does this using an API to integrate application layer firewalls, IDS, SIEM, IPS, and other security tools in helping to create a strong defense against detected attacks.

Greg Martin, CEO of Threatstream, made a claim for the inexhaustible the value that honeypot as a security research tool brings, but the workload that goes into its deployment, running and high cost of maintenance, most organizations have considered it very unrealistic and therefore ignore. The Modern Honeypot Network was then created to simplify honeypot problems in deployment and ensure it becomes an essential security tool for companies in various industries. Modern Honeypot Network can be deployed as different open-source honeypots which includes Dionea, Conpot, Kippo, Amun, Glastopf and most importantly, the Snort and so on.

2.1.3.3 Snort Honeypot

Intrusion Detection System attacks tends to be more cultured and cynical as years roll by. It is becoming more and more challenging to perform automatic real-time simulation attack and monitoring. Humongous data is always generated from which analysts must be observant enough to form themes or patterns. However, the voluminous event flow produced by the IDS sensors makes it relatively difficult in uncovering attack plans, a feat security administrators usually look towards achieving.

Intrusion Detection System seeks to identify and provide on-the-spot intrusion observations. Also, reactive IDSs (and IPs) like Snort allows for the interception, response, and/or prevent the intrusions. IDSs now have in-built sensors that allows for detection of attack signatures in data packets, and a few advanced ones have detection mechanisms that focuses on behavioural activity in determining malicious traffic themes and patterns. Supposing the packet signatures are not a perfect match with the stored signature packets in the IDS signature knowledge-base (database), the detection system activities alerts administrators about the possibility of a fresh attack.

2.1.4 Intrusion Detection and Prevention Systems (IDS/IPS)

Modern day Network Security is thriving spontaneously, this is due in part to the fact that every user takes special caution with his systems to avoid unexpected and unintended intrusions and attacks by malicious users or a hacker. Cloud computing has grown sporadically providing services of different functionalities, some of which poses some great security challenges to users of the systems. Security issues are vast and of different impact on systems. Figure 2.4 shows the general classification of Intrusion Detection Systems.

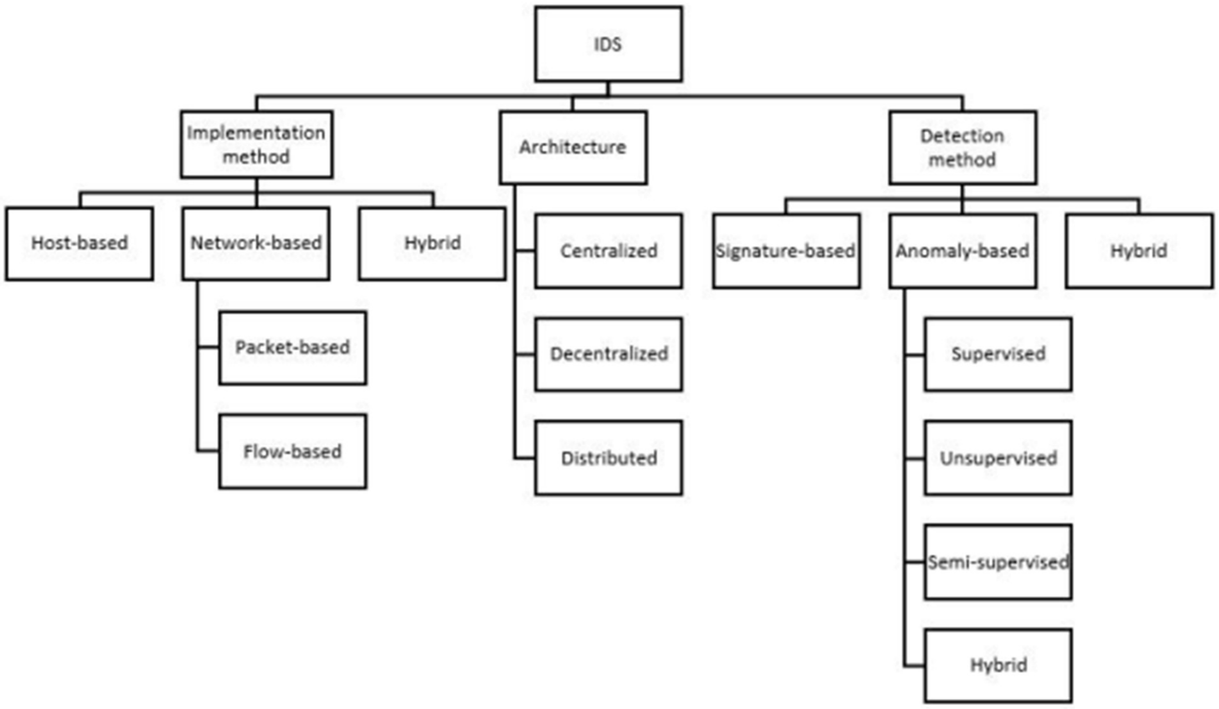


Figure 2.4: General Classification of Intrusion Detection Systems (Source: Nisioti et al., 2019)

Issues such as viruses, Denial of Service (DoS), hacking intrusions worms, and so on, among others have become a norm with Web Services. Cloud Computing affords users the privilege of having all required resources encapsulated and centrally monitored from a main controller in cloud computing area creating a loop hole which is a simple way for intruders to attack. Furthermore, experienced or knowledgeable attackers need not do much to discern the weaknesses of systems and retrieve the valuable information or resource and therefore, making it essential to guard against attack or intrusion. Additionally, in handling low latency or poor performance for clients, it becomes imperative for an administrator to focus on filtering malicious accesses. A few of the customary Intrusion Detection and

Prevention Systems do not succinctly abrogate the issues discussed above (Selvaraj et al., 2016).

Intrusion Detection and Prevention Systems appeared as one of the best network security solution especially when compared with several other techniques. The fact that IDS and IPS are a regular background system, computer systems and monitoring network traffic makes their analysis of network traffic for potential intrusions over the network originating outside the firm in addition for system attacks or misuse created within the firm.

The System Administrator hinges on multiple devices in protecting and monitoring their systems and network. Identification of the system or network activities lies solely on the system administrator and he does so by monitoring activities and pushing alerts when certain activities are noticed, activities like network traffic scanning to resolve linked computer systems. So many application software exists for this purpose, an example is ID software or device that monitors network or system activities for the purpose of detecting malicious activities or policy violations and processes the reports for the view of an administrator. Intrusion Detection and Prevention Systems exists as one of the several techniques created to stop unauthourized accesses; however, it mainly focuses on detection of unexpected events, generating log files and reports about system intruder.

Intuitively, intrusions within a system are usually the activities that violate established system security policies while IDS is the programmed process used in identifying these intrusions. Studied for a couple of years, Intrusion detection hinges on the beliefs that the behaviour exhibited by an intruder will be succinctly different from a legitimate user which makes many unauthourized actions easily detectable. This allows us to classify Intrusion

Detection systems into three broad categories, viz: Signature-based, Anomaly-based and Specification-based detection systems.

2.1.4.1 Signature-Based Detection Systems

This detection system also commonly referred to as misuse based, is a type of detection which is very efficient against known attacks. It is largely dependent on receipt of patterns and regular updates from the attackers which makes it unable to detect unknown previous threats and also new releases. A major issue with signature-based IDS is that every signature must be stored in the database, hence, a complete database will contain multiple patterned entries. Each received packet is then to be compared with all the database entries. By doing this, throughput is slowed drastically and can also be very resource-consuming making the IDS susceptible to DoS attacks. Majority of the IDS evasion tools capitalize on this vulnerability using it to flood the signature signature-based IDS systems with several redundant packets thereby causing traffic overflow, hence, making the IDS time out and drop packets and as a result, thereby possibly missing attacks. Furthermore, signature-based detection systems still exhibit vulnerability against new attacks because it relies on the signatures currently in the databank in detecting its attacks.

2.1.4.2 Anomaly Based Detection Systems

This type of detection is determined by the network classification: normal and anomalous, which is based on analysis or rules as against the patterns or signatures of the signature-based IDS. Before absolute implementation of this system, adequate knowledge of the network behaviour must be made known. Unlike the misuse based detection system, Anomaly based detection system can detect previous unknown threats, but the false positive to rise more probably.

2.1.5 IDS and IPS Hybrid Issues

Over the years, there has been so many significant improvements on Intrusion Detection Systems with diverse forms of incorporations and marginalization. Most prominent amongst these HIPs have been in the area of CAPTCHA development. A CAPTCHA's work has been to decipher the user of a system (note: not the intent of the user but the kind of user – human or bot) and this is largely carried out through a challenge-response task more often than not, inserted into web forms. Its purpose is solely to restrain automated submission of forms by bots (automated scripts posting spam messages at the slightest available means). This module in itself provides this feature to most users of web forms on any interactive site.

This mechanism has been a huge success down the years and has definitely prevented attacks on web forms in diverse measures. However, with the success of this development comes the tenacity of bot developers and black hackers in breaking down this major obstacle to them perpetrating their actions. Hence, in recent years, a couple of concepts have been conceived and a few of them introduced and hybridized into CAPTCHA. This concept of layering two different Intrusion Detection Systems and hybridizing them to detect and counter significant bot attacks on web forms has not really gotten much acclaim amongst web developers for various reasons, majorly usability. Other issues arising from hybridization of two or more IDS includes Content Access Control, Security, Spam Prevention, User Access and Authentication, User Management, amongst many others. These issues are explained as follows:

2.1.5.1 Content Access Control

Content Access Control is the part of IDS hybridization that allows for permissions management in determining the type of content by role and user. It allows for specific customized view, editing and delete permissions for whatever selected content type. Furthermore, it has the option of enabling settings for each content access, thereby making it possible for customization for each content node. There is a level of access that every IDS is allowed, hence, the concept of hybridization does either of two things: grants more access or restrict more access. Either ways, the level of access is not expected to be compromised when hybridization occurs. Both access conditions must be satisfied before any form of data access is allowed. Controlling access to content involves how much of information or data is embedded into the CAPTCHA module and being presented to the user to solve. Essentially, these controls revolves around three distinct modalities:

i. The role-based access control settings which is a default content node.

ii. Every type of content can be streamlined to its default content access role settings.

iii. User access control which leverages on the type of user trying to gain access.

2.1.5.2 Security

Security is that system functionality that emphasizes on the absolute protection of Data and information within the system. Within the concept of hybridization, security issues focuses on the fact that the two different IDS being hybridized have their distinct security mechanisms and layering them is not indirectly exposing the vulnerability of each of them. It inadvertently confirms the source of the request to be sure it is not from a spambot or a crawler. It does this by carrying out an inspection on the HTTP header of the UserAgent before limiting the number of crawler requests allowed to be executed within the given time frame. An error code is generated by the server as a response message when the limit is exceeded. Every IDS is built differently but with one sole aim, determining if the user is a human or a bot, however, they are all configured differently and how secure one is might be undermined by the vulnerability of the other and vice versa. It is on this premise that security of the hybrid IDS is being questioned.

2.1.5.3 Spam Prevention

Spam protection has been identified as a vital part of managing interactive web forms. As spambot experience a surge and increase amongst webforms, it becomes expedient that spam detection tools which helps alleviate unwanted messages and in the process increase user productivity and improve system performance through the evasion of unnecessary traffic from web servers should be developed. For example, Antibot is an extremely lightweight incorporated Drupal module specifically designed to combat spambot submissions on your webforms in an innovative-fashion. This module works below the surface and neither requires any for of user-interaction from end-users. The singular expectation of the end user is the enablement of JavaScript on their browser. Without this, the supposedly protected forms will not be displayed at all and a notification appears to the user informing of the need for the JavaScript to be enabled as it is a requirement for the form. The absoluteness of the fail-proof of this technique is very much unlikely to begin with. In theory, there is the possibility of a persistent spambot to retrieve data from your web server your decoy system notwithstanding. However, much research into the subject-matter suggests that the bulk of spambots do not interfere with data already hidden using JavaScript. It already is a known fact that javascripts cannot be read by most spambots at all.

2.1.5.4 User Access and Authentication

Of the many-sided aspects of website authentication, much of the focus has always been on the user and the human-to-computer interactions that follows suite. With this in mind, user authentication becomes crucial in understanding creation or improving a website's login procedure. Either amplifying the internal security, providing a more friendly and interactive customer-experience for users of your site or simply increasing customer acquisition, one thing is of utmost significance, knowing how the user authentication fits into the solution. It is an essential module providing the generic service of granting user-access and also registering users with prior authentication against an external site or service and thereby storing the authentication parameters. When an attack is initiated simply through the knowledge of the username, it only takes a brute force attack to impersonate that user. In preventing this, it is usernames also needs to be as protected as much as passwords are. Username Enumeration allows attackers trigger the "forgot password" mechanism of web forms to detect usernames. The attacker uses the trial-by-error mechanism by entering a non-existent username thereby receiving a "Username does not exist" response. Continuous trial by the attacker on usernames on the form will eventually produce a valid user. If the user does not exist, no password reset email will be sent, but the attacker will not know this is the case.

2.1.5.5 User Management

There are several modules that comes into fore when user management is being considered. Everything that has to do with the users of a platform are all encompassed in this segment. Modules such as role administration, password reset, node access, simple authentication, block content permissions, protected webpages, secure login, and so on. Some webpages provisioned via HTTP and HTTPS, Secure Login masterminds the secure submission of user data and other forms via HTTPS, thereby preventing the transmission of sensitive user data such as passwords. Secure Login does not only lock down the user/login page but goes a step further to lock any page containing any form of user interaction login, as well as other forms it is configured to secure.

2.1.6 Improving IDS and IPS with Feature Selection

During the last decade, researches have focused on improving Intrusion Detection Systems detection rate as well as its performance. This is achieved by shifting attention on the algorithm for detection and considering different techniques or a combination of both. However, attention may have shifted away from the process which appears to be at the center of it all, that is, feature selection (FS). FS is a process that helps with identification of an optimal subset of important features representing each class over the original set. More often than not, feature selection is of much more importance compared to detection algorithm preference.

From an optimal subset describing the input data efficiently rather than the whole feature space, FS does not only enhance system accuracy, but also decreases the false positive and computational time. Feature Selection on its own is not used in creating new features, however, it does its selection based on relevance and non-redundancy. This leads to overfitting inclusion and poor generalization in its classification or clustering process (Miao and Niu, 2016). The feature selection process revolves around two major components: an evaluation criterion and a search strategy. The latter choice is the reason for choosing the considered features as part of the optimal subset, while the criteria for evaluation scores each feature. Peradventure this score exceeds a threshold, it becomes relevant and subsequently included in the subset.

Feature Selection method is divided in two broad categories: filters and wrappers. Filters are not specifically concerned about the choice classification technique, however, they assign score to the proposed features with information theory and statistical methods, revolving around correlation coefficient, information entropy, information gain, and mutual information. Hence, a filter is a more favorable method because it is fast and simple. However, in contrast, a wrapper, as depicted in Figure 2.5, carries out a candidate evaluation subset factored by the detection algorithm used via a predicted model. In each of these iteration, a feature subset is either accepted or rejected considering its use by the classifier on the training set and however the results are interpreted. In as much as wrappers considers detection algorithm before producing a subset adjusted to the specific algorithm and Intrusion Detection System, they also can lead to overfitting and can be intensive in its computation, with a special focus on the network data, usually highly dimensional.

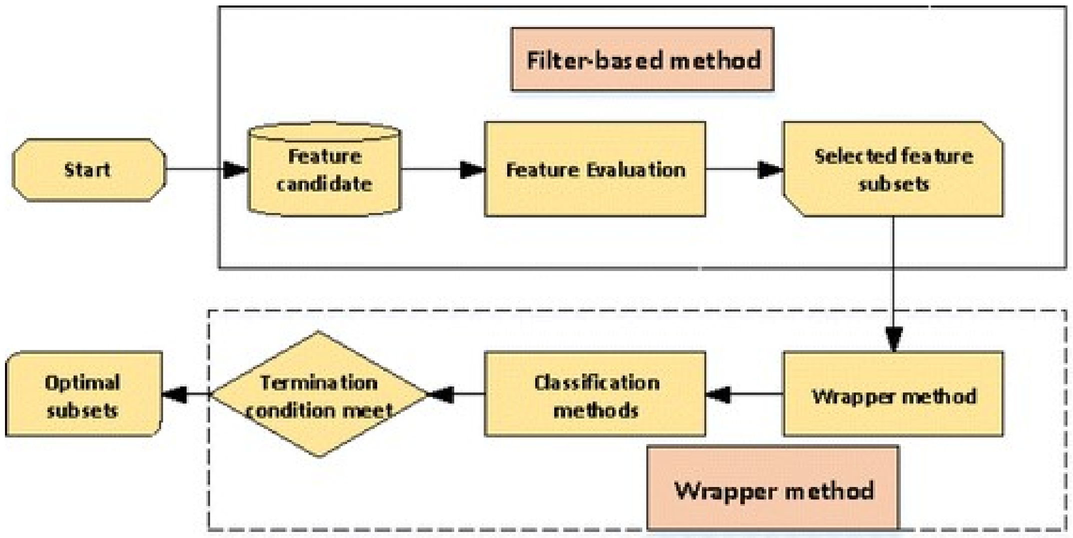


Figure 2.5: Feature Selection Methods (Shukla et al., 2019)

2.1.7 Other Antispam Security modules

Drupal exists as an open-source web content management framework which is written in PHP and made free and distributed under the GNU General Public License. It is a widely acceptable backend framework making provision for a major chunk of available websites worldwide – be it for personal blogs or corporate, political, and government sites notwithstanding.

The Drupal open source framework has been compartmentalized so much that there exists several modules and chunks that helps facilitate ease of integration with indigent sites. There exists so vast a variance that finding just one module that is all-encompassing and does exactly what you need in one package is frowned at in Drupal. However, having several chunks of modules each doing their part and well harnessed is the way Drupal effectively functions. Fields and views are a prime example. Having several specialized separate modules different pages on the site is not a phenomenon embraced. Usually, you want Field and Views modules and friends that is made available for self-configuration to do all that and much more. A few of those modules help combat spam intrusions into systems and they will be considered in this segment.

2.1.7.1 Antibot

Antibots are services that provides effective defense mechanisms against bots in Web applications, HTML5 websites, mobile apps, and APIs. They help to reduce the risks emanating from specific web vulnerabilities. These services exists in all variance of user-interaction web forms. It is a self-defined custom protection policy that uses reverse proxy technology-based SaaS solution in identifying and controlling malicious traffic. The concept of Antibots rests solely on the ability of JavaScript to present the form to the user and withhold submission of same until human user interaction is initiated and detected (this can be either a mobile swipe gesture or a mouse movement, key press and so on the form-action is then switched to the appropriate path set for it).

2.1.7.2 Enforced Spaces

This concept is inspired by spambot usernames. It is common practice by spambots to come up with weird usernames with whitespaces. Hence, this antispam mechanism checks by default the possibility of the registering username having any space. When spaces are detected, the registration is refused, otherwise, it is permitted. Also, this mechanism thrives on a pre-defined configured number of user-defined characters.

2.1.7.3 Spamicide

The spamicide module helps to protect Drupal forms by embedding a hidden form field invisible to human users, but visible to spam bots. This makes bots make an attempt at entering data in that text-field when in actual fact, nothing is expected in that field. The module doesn't require appropriate permissions for administrative pages leading to an Access Bypass. The concept adds a visually hidden form element, and refuses input if form field is not empty.

2.1.8 IDS Evaluation Measures

Although the accuracy of an Intrusion Detection System is a very vital requirement, it is not the sole requirement. A system's response time has been found to be one of the most significant factors, as it is incorporated in rapidly evolving enterprise networks where minimal latency can produce a huge organizational monetary losses. Overtime, different datasets and metrics have been a yardstick for measuring the strength of a system in successfully identifying and mitigating normal traffic as well as different forms of attacks in a dataset, which poses a difficulty for result comparison of the deluge of proposed systems. However, some evaluation measures are very common and help with the detection ability of the system. They are:

2.1.8.1 Accuracy

This takes into account all results containing the true positives (TP) as well as false negatives (FN) and it is a measure of the ratio of the total number of instances as against the correctly classified samples.

2.1.8.2 Sensitivity

Commonly referred to as True Positive Rate (TPR), it is the rate of positives samples, in that way, correctly classified. Alternatively, specificity or True Negative Rate (TNR) focuses on the measure of instances correctly negatively classified in its measurement. Also, False Positive Rate (FPR) showcases the measure of samples that are erroneously identified as anomalies.

2.1.8.3 Confusion Matrix

It is also referred to as error matrix. It provides a visualization of the relationship between the expected (predicted) results actual results. Purposely utilized in supervised learning, it evaluates the prediction accuracy of a classifier. For every row of the table, there is a corresponding result predicted by the classifier, while for every column, there is a correspondence to an actual result.

2.1.8.4 Recall

Recall focuses on the portion of the true positives that have been successfully retrieved. Alternately, precision refers to the proportion of retrieved instances that can be accurately identified. Although recall, as well as precision focuses on the positive samples, in the actual sense, neither of them takes into cognizance how well the model handles negative cases (Vasilomanolakis, Karuppayah, Muhlhauser, and Fischer, 2015). The harmonic average of the two previous measures is referred to as F-measure (F1). Although F-measure is purported to be a single measure in capturing the effectiveness of a system, it still totally ignores True Negatives (TN) (Powers, 2020).

2.1.8.5 Usability

Where a device or system is deemed usable, some usability evaluation methods can help determine the extent of its usability, with the use of reliable, accepted and very robust metrics. In simple terms, usability evaluation helps us to assess the extent to which an interactive system is convenient and pleasant to use.

2.2 Conceptual Issues

On a sunny afternoon, June 27, 2017, an attack was launched against Ukrainian critical infrastructure that almost grounded the whole economy of the Nation. Although the damage did not include exploits targeting industrial systems, it did have a monumental impact on industries all around the country as well as world-wide, with great losses in major corporations all around Europe. Soon after NotPetya was first detected, several other infections were simultaneously detected around the world. But NotPetya is not Petya: the previous Petya ransomware released in 2016 and this new infection call NotPetya was not to be confused for same.

A ransomware is basically a malware that encrypts documents and storages thereby preventing file which will require payment to be decrypted. Petya, a ransomware was published in March 2016. The most significant difference is the fact that NotPetya is not a

ransomware. Successful execution of NotPetya on a platform encrypts the whole storage (hard drives) without any method to decrypt stored data. This only points to the fact that NotPetya's was not a crime aimed at financial exploitation. In line with the operations of Wannacry, NotPetya embeds an effective infection method targetting Windows SMB.
However, unlike Wannacry, NotPetya exploits all remote machines connected on the same local network. It has functionalities to access and extract passwords and some remote administration functionalities. It therefore brings us to the conclusion that NotPetya was not a cybercrime to make financial gains or control a BotNet, rather, it is targeted at infecting a precise target. The initial infection vector came from a malicious update of the Ukrainian software M.E.Doc that was carried out in only one of their remote locations thousands of miles away. Indeed, hackers took the control of a M.E.Doc's server update and infected an update with NotPetya (Butrimas, 2018).
The real world analogy above only points to the fact that the Web and its various forms of applications needs to be protected to avoid a total collapse in the world economy.
CAPTCHAs have been found to be one of those ways through which System Intrusion and Security breaches can be detected and over the years, major research works have been done, some of which this section of the literature reviews. DESIGN and RECOGNITION are two distinct major categories that the research works on CAPTCHA have focused on.
For CAPTCHA DESIGN, the research work is geared towards novel approaches to developing CAPTCHAs to further enhance the security of the system while the works in the area of CAPTCHA RECOGNITION verifies the security of existing and prevalent CAPTCHAs and thereby promote the development techniques that are novel especially

considering CAPTCHAs have become a very formidable component of AI (Artificial Intelligence) and a very important prerequisite in HIPs and Human-Computer Interactions. Suppose in a building which houses all company staff of an organization, the official closing time in 17:00 GMT. Suppose the janitor is supposed to shut all doors and windows firmly latest by 18:00 GMT. One would think that with such measures in place, it will be foolhardy for the organization to be paying security operatives to run night shifts on the building. However, virtually all organizations and companies still do same, having security operatives run night shifts on buildings not momentarily inhabited. This is because no matter how secured a system appears, it cannot absolve itself of the vulnerability of attacks. It is a common knowledge that the most vulnerable man is the one that thinks he is safe. Not losing guard is an essential part of Intrusion Detection/Prevention systems. The scenario painted above depicts what happens in a physical structure, however, the risk is much more volatile when the virtual world is considered. A system where evil can be perpetrated without the culprit not necessarily being physically present. Intrusion becomes a norm and consistently so.

Computer systems and indeed the virtual world cannot be said to be safe even with the most advanced protection module incorporated. As a matter of fact, majority of computer security experts conform to the fact that, with user-desired features in place, features such as network connectivity, the point of a totally secured system will never be achieved. With this in mind, it becomes important to develop and deploy intrusion detection/prevention tools and techniques to discover and mitigate computer attacks (Souley and Abubakar, 2018).

With the advent of diverse exploits being used in compromising networks, exploits capable of breaking into any secured networks, increasing the efficiency of network security has become a necessity, hence, the introduction of Honeypot. In 2018, a system was proposed which combined specific features and services of IDS, IPS and Honeypot. Honeypot in itself helps detect mitigations using IDS; it does this by trapping and deflecting the packets as received from attackers. The result of this work proves that the system is capable of handling multiple clients with the concept of honeypot. Intrusion detection system (IDS) monitor whole network and looks for intrusion. Honeypots are activated when any form of intrusion occurs. This will then divert the network traffic to a dummy/virtual servers and gets hold of the IP address of the attack source. The major drawback of this system is its support for multiple clients (that of an attacker inclusive), which compromises the system easily (Baykara and Das, 2018).

2.3 Related Work

Pope and Kaur (2005) protected the vast number of eCommerce sites across the globe through CAPTCHAs, hence, they came up with a research into discerning between Human users and Computers. They opined that CAPTCHAs can assume several programmable distinct ways to distinguish humans from computers and this makes it difficult for computer bots to have a field day with e-Commerce sites thereby reducing spamming activities. Authours proposed an image made up of letters and numbers that are pseudo-randomly arranged and either run through some degradation algorithm or strategically made to face an obfuscating background with the sole aim of making the final image of the optical character recognition (OCR) very much impractical. This work was a huge success at the time, however, with time, the vulnerability of the work to blind-guessing has been made obvious as much as it is with dictionary attacks.

Chellapilla and Simard (2005) used Machine Learning in breaking Human Interaction Proofs (HIP). They observed tasks where Human solutions produced a much more soothing results when compared to Machine Learning Algorithms. With this insight, they discovered most HIPs are simply recognition tasks and hence could be easily detected using machine learning. Thus, they were able to develop effective HIPs which capitalized on bringing up tasks focused on character-segmentation in order to confuse machine learning algorithms.

Shirali-Shahreza and Shirali-Shahreza (2008) came up with a research work that tests for human intuition and ability to solve distinct mathematical problems incorporated into CAPTCHAs. In the research, an elementary mathematical problem is generated according to a predefined pattern but rather than using the object's name, images are preferred. The entire problem is saved and portrayed to the user in form of an image to be deciphered by user. But since answering this problem requires four abilities of understanding text of question, detection of question images, understanding the problem, and solving the problem, it becomes almost impossible for bot programs to have functionalities such as the ones already mentioned. This technique is limited however to humans that possess the ability and knowledge of solving computational problems. Other users who have no prior knowledge or ability to solve these elementary mathematical problems are at a disadvantage and eventually become victims of the same predicament as bot programs making the purpose of CAPTCHAs of non-effect.

Kluever (2008) made an audacious quest by video-tagging CAPTCHA task for humans to solve, i.e. labelling of videos in a content-based way. Videos are extracted from public domains (For example YouTube) and are used as CAPTCHA problems. About one hundred and eighty human participants were involved in conducted experiments focusing on certain metrics and their responses were distinctly analyzed. The singular pass-mark for the CAPTCHA challenge amounts to the solver successfully providing not less than three (3) unique words to which the video just watched can be labelled. The experiment was a huge success at the time as relatively 70% to 90% of the respondents were able to tag the video with nothing less than three labels when compared to the bot programs success rate of 13% in tagging such videos. One of the major setbacks to this work is that of tag frequency estimates which are not publicly available thereby creating a huge problem. Additionally, distortion within the network poses a major hindrance too.

Hindle, Godfrey and Holt (2008) worked on a reverse-engineering CAPTCHA that focuses on the identification and recognition of such CAPTCHAs. The authours utilized simple available image processing techniques in solving real-world CAPTCHAs encumbering thresholding, dilation, fill-flood segmentation, bitmap comparison, and erosion. With the aid of white-box and black-box methodologies for reverse engineering, CAPTCHAs were successfully solved. The major limitation to the work was the fact that the model focuses solely on Image-processing CAPTCHAs neglecting all other forms of both OCR and Non-OCR CAPTCHAs thereby causing a vulnerability to bots program with a specific target at the OCR-enabled Image processing.

Li, Wang, Tian, Lu, and Young (2009) combined a slightly modified Random Mutation Hill Climbing (RMHC) Feature Selection algorithm with multiple linear SVMs through a wrapper system using no form of evaluation criterion. The algorithm ws used against a KDD99 dataset and the method was also taken through three distinct stages which includes the generation of initial data subsets, after which the process went through an iterative procedure: generating subset with modified Random Mutation Hill Climbing algoritm and comparison with the previous result. Finally, an optimal subset after a successful completion of iterations or predefined criterion is satisfied was sought after.

Banday and Shah (2009) proposed a class of clickable and flappable CAPTCHA design in an image-based CAPTCHA technique. The technique presents images and sub-images of CAPTCHA to a user. The proposed technique possess properties such as an improved security surpassing that of usual OCR-based techniques, improving webpage user-friendliness while also consuming less webpage area. The major limitation to the model however was that it highlights presentation and distortion issues as it relates to dimensions.

Yamamoto, Tygar, and Nishigaki (2010) knew humans possess the unique ability to identify any form of strangeness as presented in a CAPTCHA, hence leveraged on such ability. Their research work was the development of a new CAPTCHA series that utilizes translated and interpreted sentences by machines. Strangeness in Sentence CAPTCHA, also commonly referred to as SS-CAPTCHA works by determining and differentiating how proficient humans are at distinguishing between machine-generated sentence translations and natural sentences as constructed by humans. The work presents a set of natural sentences termed as “P” as created by humans (NSs) and a few other set of garbage sentences termed “Q” which was machine-generated (GS) from a set of natural sentence and presented to a user. Afterwards, both sets of generated sentences P and Q are combined (P+Q) and are then randomly placed. The puzzle for the user is then in selecting the set of Natural Sentences from the combined sequence of the pool of sentences, (P+Q). At the conclusion of selection, should all selection be that of humanly generated sentences, it is classified successful and that suffices to say the user is human. On the other hand, a deviation from selection of full natural sentences, and one or more errors certifies the opposite – presence of a bot program. This CAPTCHA as successful as it was then has a limitation which thrives on the comprehension of the user. Furthermore, language barrier becomes an issue here as a Chinese will be unable to take the CAPTCHA test should it be presented in English and vice versa whilst message translators will not be an option.

Almazyad, Ahmad, and Kouchay (2011) proposed a CAPTCHA that encompasses more than one mode in order to solve. Prevailing CAPTCHAs depends essentially on the enhanced distortion of text images, thereby making them appear unrecognizable to the recognition techniques being utilized. Social networking sites as well as various e-Commerce sites, email generation sites, auction sites amongst several others make the most of these text-based schemes. Their research focused on a CAPTCHA with a new anti-modal technique (Picture and Text based). An image is being rendered on a screen with many text labels drawn over it. The ability of a user to correctly identify the appropriate name of the displayed image from the set of text labels scattered all over it. This research however possess a bottleneck as it becomes an arduous task (not forgetting human impatience) for humans to solve so many CAPTCHA tests with many underlying layers, leading to frustration for humans should the tests be cumbersome.

Raj, Devassy, and Jagannivas (2011) introduced graphically-inclined Image-based CAPTCHA. The major advantage of this CAPTCHA over other text-based CAPTCHAs is that, no bot-program can perform any form of edge-detection random guessing, thresholding, segmentation nor shape matching. Its analysis however allows for security check processing for a better result of the mechanism. The limitation of this work is that in the near future to the time the work was done, novel algorithms were already a menace when it comes to breaking Graphics CAPTCHA.

Amiri et. al. (2011) used MMIFS (modified mutual information based feature selection algorithm) on some publicly available KDD99 datasets using the mutual information evaluation criterion. The filtering process went through a couple of phases which included selecting the first (1^{st}) feature, with the maximum Mutual Information as result. Next, was the Greedy selection: a process to compute feature-feature Mutual Information and select the one with the best and optimal result. Conclusively, a repeat of the sequence is carried out until the expected feature population is accomplished.

Zhang, et al. (2012) also used two different Feature Selection algorithm in Weighed Symmetrical Uncertainty_Area Under Roc (WSU_AUC) and Selection Robust Stable Features (SRSF) in weighted form using Weighted Symmetrical uncertainty as the evaluation criterion filtered with the WSU metric after which a selection of the optimal features with wrapper based on AUC and then chose the robust features as outcome.

Casas, Mazel, and Owezarski (2012a) in their work utilized a Tree-based subspace clustering (TCLUS) algorithm on some KDD99, NSL-KDD and TUIDS in a mixed input format. The process involved a generalized entropy and mutual information based feature selection technique, a tree-based sub-clustering and an outlier detection using ROS' score and a predefined threshold. The resulting effect of the hybrid was an almost perfect DoS system, producing a 99.99 efficiency in service denial amongst several others, one of which is the Remote 2 User (R2L) efficiency of 89.96.

Casas et al. (2012a) utilized a DBSCAN algorithm on some KDD99, MAWI and METROSEC with no further input. The process involved a multi-resolution traffic flow, a time series criterion for detecting a flows with potency of being malicious and a Sub-Space Clustering (SSC). Finally, an Evidence Accumulation Clustering (EAC) ranking.

Om and Kundu (2012) developed a hybrid IDS that focused on a combination of K-Means, K-NN and NAÏVE BAYES algorithm on a KDD99 mixed input stream. The work indulged a methodology that incorporated a three-module hybrid. The first module focused on an entropy based feature selection while the second module focused on clustering, both the normal and the attack, while the thrird module was for a succint classification of the various types of attack.

Casas, Mazel, and Owezarski (2012b) in another research on a hybrid IDS focusing on the DBSCAN algorithm using a METROSEC dataset and no further input stream. The method incorporated in the work involved a sub-space Clustering and ECA to partition the feature space into N different subspaces and then in a given lower dimension space, clustering is performed.

Bhuyan, Bhattacharyya, and Kalita (2012) in their bid to create an hybrid IDS utilized the Tree-based subspace clustering (TCLUS) algorithm with a focus on the KDD99 and TUIDS dataset in a combination of inputs. To establish the effectiveness of their work in finding all possible clusters, a stability analysis was performed on a cluster. Also, an effective cluster labelling technique (CLUSLab) was introduced in order to create labelled dataset based on the stable cluster set generated by TreeCLUS. The method used can be divided into four (4) cases, viz; iteration, stability analysis and unsupervised cluster formation through the procedure until stable clusters are attained and finally CLUSLab: cluster labelling technique.

Zhou, Huang, and Wang (2012) worked on a hybrid IDS used LDCGB algorithm with a focus on the KDD99 dataset without any further input. Although not applicable when it comes to attack mechanism, their work still followed a few known procedure which can be graph-based algorithm and an outlier detection based on the local deviation coefficient.

Song, Takakura, Okabe, and Nakao (2013) had their research focus on using Neural Network with Random weights (NNRw) on some KDD99 datasets as a single input. The method works such that a divide-and-conquer methodology is incorporated using the magnitude of fuzziness to separate un-labelled data. Afterwards, a neural network with random weights (NNRw) model is employed in identifying the attacks. Without separating into Probe, DoS, R2L and U2R, the method passed the KDDT attack accuracy test with a percentage of 84 and 68 percent respectively.

Nassar and Miller (2013) had their focus more on Honeypots and how they can effectively stop computer bots and spammers without any extra work on the part of the users. In achieving this, they introduced an innovative multilayer approach to honeypots in well secured web forms. They proposed a two dimensional honeypot for security access and this eliminates the vitality of form scanning automation so that the computer bot or spam script will not be able to differentiate the form field from the honeypot field. The limiting factor about this work is that application vulnerability scanners should not, in any way, pass the pages protected by this solution.

Fahad et al. (2013) utilised Local Optimization Algorithm (LOA) of the Feature Selection on some filtered KDD99 and MAWI datasets in measuring both goodness rate, stability and similarity evaluation. The process went through three stages which includes extracting an optimal Feature Selection subset for every of the five (5) Feature Selection techniques, after which the support is calculated for every of the feature in the optimal subsets. Finally, if support is found to be higher than the threshold, the feature will be moved to the final subset.

Elbasiony et al (2013) were focused on the combination of Random forest and weighted K-means algorithms in working on KDD99 dataset in a mixed input format. The method involves both an online and offline module. The online module involves a misuse signature comparison with the aid of the Random forest algorithm. If no match is discovered, the offline module is brought in for clustering and creation of new signatures. The resultant effect produced a maximum detection rate of 98.3% with a False Positive Rate of 1.6%.

Aljarah and Ludwig (2013) utilized Particle Swarm Optimization (PSO) clustering algorithm on the KDD99 datasets ina continuous input format. The method incorporated included using MapReduce to parallelize the PSO used in clustering the data. This action is carried out based on the global optimal centroids made available. The resultant procedure was a maximum AUC of 0.963 recorded.

Chandrasekhar and Raghuveer (2013) in their own IDS hybrid work used a combination of k-means, SVM and fuzzy NN algorithm on a KDD99 dataset with no further input. The methodology revolved around creating k clusters, assigning one neuro-fuzzy model for every of those created cluster, thereby producing some SVM vectors. Finally, radial SVM classification was used for the detection.

Gogoi, Bhattacharyya, Borah, and Kalita (2014) worked on a hybrid IDS that utilized a combination of CatSub+, K-point and GBBK algorithms to work on some KDD99, NSL-KDD and TUIDS datasets in a mixed input format. The method involved a process in which the supervised classifier detects Denial of Service and Probe attacks, leaving the unsupervised classifier with the detection of the normal attacks. Conclusively, the outlier based detection for R2L and U2R are also both detected.

Nguyen (2014) systematically investigated text-based CAPTCHAs focusing more on their security strength and on some of the animated texts (2D, 3D and 4D). Doing this involves the development of a tool box using some attacks and novel algorithms in helping with the analysis of design paradigms and the security alternative. The work proved that segmentation-resistance which was largely embraced at creation and the design of text-based CAPTCHAs are overwhelmingly indispensable design principles for animated CAPTCHAs such as 4D, 3D and 2D. Thus, the work postulated CAPTCHAs with a segmentation-resistant scheme identifying not only the characters themselves, but also the location of those characters. His approach significantly brought to fore segmentation-resistance principles and in the process, paved the way for secured and much more usable CAPTCHAs. The work is however limited in that it focused on enhancing the security of text-based CAPTCHA – which is just one kind of CAPTCHA.

Woo, Rey, and Kim (2014) revamped web Intrusion Prevention through a 3D CAPTCHA, another variant of Image CAPTCHAs involving image processing that helped in the transformation of plain text to 2D and then to 3D. Previous works, prior to that, only focused on pre-processing techniques without the use of the Internet in breaking 3D CAPTCHAs. This work, a novel 3D object-based CAPTCHA scheme, came into fruition to overlay the displayed image over a 3D object. A prototype was developed that can actually protect websites from computer bots and spammers and this was presented as POC (proof of concept) of 3D object-based CAPTCHA schemes. Their approach involves asking users to differentiate and identify the 3D object rotation task, e.g. Sketcha. Subsequently, users had to then find a way to solve the 3D text CAPTCHA. Much to general belief, 3D object rotation as well as text recognition, although easy for human to solve, becomes a herculean task for a machine to accurately solve. The work is limited in that these novel approaches exposes the vulnerability of state-of-the-art vision programs.

Fahad et al. (2014) used Global Optimization Algorithm (GOA) of the Feature Selection on some filtered Cambridge Lab datasets in measuring both stability and optimality evaluation. The process went through three stages which includes combining several Feature Selection techniques in a bid to find the optimal subset, after which an adaptive threshold based on maximum entropy and then concluding with Random Forest filtering. Hosseinpour, Amoli, Farahnakian, Plosila, and Hämäläinen, (2014) also had their work focused on using DBSCAN algorithm on a solitary KDD99 dataset without any other input. The method used primary innate immunity which involves clustering into self and non-self. Furthermore, secondary adaptive immunity which invariably meant from the results of the clustering detectors generated, every host will have a chunk of it when they eventually become mature.

The research of Vahdani et al., (2015) focused on using the DBSCAN algorithm on DARPA and ISCX datasets with no further input. The method makes use of two (2) different engines: The first engine focuses on clustering and outlier ranking using a

Dynamic Self-Adaptable Threshold while the second engine focused strictly on the detection of Botnet.

Costa, et. al. (2015) had their own work focusing on using Optimum Path Forest Clustering (OPF) algorithm on a combination of datasets involving ISCX, KDD99 and NSL-KDD with no other input. The method involves the use of nature inspired optimization techniques to set k parameter for Optimum Path Forest clustering of the data.

Liu et. al (2015) divided their research findings into two distinct phases using a filtering system. They used the Class-Oriented Feature Selection (COFS) Algorithm on a few datasets which included Cambridge, UNIBS and SCUT. The first phase produced an evaluation of both the global as well as the local score of its feature and also a search for relevant and discriminative subsets while the second phase focused on removal of redundant features from all subsets. The resulting evaluation yielded maximum entropy.

Lin et al. (2015) came up with a hybrid solution used as a combination of clustering and KNN algorithm on a KDD99 dataset with no further input. The methodology involved the extraction of nearest neighbors and cluster centers. Also, the calculation of dist1 and dist2, before conclusively summing up dist1 and dist2 towards the creation of a new distance feature assigned to each dataset point. Finally, k-NN classifier was then used for the newly generated dataset.

Abinaya et al. (2015) shed more light on distinguishing the deciding factors as regards phishing websites and in the process highlight the important features responsible for assessing the benefits rule-based data mining classification techniques brings in when it comes to predicting phishing websites while also identifying the more reliable classification techniques. The authours looked diligently at how the privacy of image CAPTCHAs can be preserved through segmentation of the original image into different shares which are then stored in separate servers and database such that the original CAPTCHA image is only made available at the instance both segmented images are simultaneously made available. The research highlighted the fact that both sheet images cannot reveal the identity of the real image CAPTCHA individually. The revelation of the real CAPTCHA image to the user will automatically serve as the password. The work is limited in that layers or transparent images are vital requirements before the information can be revealed.

Hernandez-Castro, Moreno and Barrero (2015) analyzed both a text-based and puzzle-based Image called Capy CAPTCHA, looking out for vulnerabilities that may exist from the perception of an attacker. The work involved the presentation of a side-channel attack which necessarily does not solve any of the recognition challenges. On the contrary, the continuity of the image's measurement and size as a low-cost attack was leveraged on. The work is limited in that image sizes of significant difference (say 10 pixels apart) becomes vulnerable.

Jha and Acharya (2016) worked on hybrid IDS utilizing a combination of Hidden Markov Model (HMM) and Decision tree algorithms to work extensively on KDD99 datasets without further input. The method involved a two-layered Immune system Inspired IDS (I3DS), T-cells layer. The Hidden Markov Model (HMM) focuses on identifying possible attacks and B-cells layer while a decision tree focuses on confirming only the true attacks. The detection rate was found to be a little above 60% and a Feature measure a little above 64%. Furthermore, it has a precision of 77.8%.

Bhuyan, Bhattacharyya and Kalita (2016) utilized Mutual Information and Generalized Entropy Feature Selection on some publicly available datasets (MIGE-FS) which includes KDD99, NSL-KDD, TUIDS, UCI in a filtering system. Mutual information was the method used for feature-class relevancy while generalized entropy was the method used for feature-feature non-redundancy. The resulting evaluation criteria was both mutual information and generalized entropy.

Bohara, Thakore, and Sanders (2016) worked on an hybrid system that utilized the DBSCAN and K-means algorithm in a probe-less system working on some VAST 2011 Mini Challenge 2 datasets without any input. The method incorporated involves Feature selection for redundant information using Pearson correlation and Clustering on the host and network data in determining attacks through cluster normalcy.

Bostani and Sheikhan (2017) proposed an hybrid IDS through the means of K-means, Modified Optimum Path Forest (MOPF) algorithm on a NSL_KDD dataset with no other input. The resultant effect was not as robust as the work of Casa et. al., 2012, however, it also produced some credible Figures in terms of probe-factor, DoS, R2L and U2R. The procedure revolved around three (3) basic modules: First, it was partitioning: using k-means algorithm in creating training subsets for detection module. Next, it was pruning: pruning the training data subsets with the aim of speeding up MOPF. Third module involved detection: using the MOPF from module two to detect other forms of attack.

In another research, the authours utilized the NSGA-II Feature Selection Algorithm and a GHSOM classification with probabilistic relabeling in carrying out its five-stages experimentation on the NSL_KDD datasets provided using a jaccard coefficient as its evaluation criterion. The 5-stage experrimentation involved selecting a convenient feature subset of choice, training the appropriate classifier, dataset classification, evaluation with the chosen criterio - Jaccard coefficient and finally, population evolution.

Chen et al. (2017) focused on CAPTCHA recognition reviewing some of the recent developments when it comes to text-based CAPTCHA recognition. They also proposed a distinct recognition scheme for text-based CAPTCHAs which thrives on a five-stage operational mechanism namely; Preprocessing, Segmentation, Combination, Recognition, Post-processing amongst others.

Zhu et al. (2017) provided a six step methodology using a combination of I-NSGA-III and GHSOM Feature Selection algorithm filtered in a wrapper method to work on some KDD99 datasets with Jaccard coefficient as the evaluation criterion. The six stages involves the initialization of the population datasets, training of the classifier used, the actual classification, result evaluation using the Jaccard coefficient, a comparison of the convergence, upon which evolution of the population is based on two parameters [first, special domination method (bias-selection) or second, a predefined multiple targeted search (fit selection)]. The final step in the six stage method involves a repitition of the second and fourth steps until convergence is attained.

Ashwini et. al. (2017) proposed the implementation of middle interaction production honeypot. The main goal of the work was to secure the server side from the attackers using honeypot. The resultant effect of the work indicated Clients can communicate to the servers with the aid of the honeypot only. The clients is in possession of the fake IP address of the honeypot but not that of the server's. Were the client a genuine client, the request is directed straight to the honeypot. Honeypot manipulates its IP address before pushing the request to the original server. The server then responds to honeypot. Again honeypot alters its IP address before returning the response to client. Should the client be a bot or spam, then honeypot tracks the information about the attacker, locates, identifies and saves such. Though an attacker, it still responds just to fool them. In doing all these, security is not in any way breached. However, the limitation here is the idleness of the honeypot system when there is no attacker in view. It therefore becomes imperative that the honeypot be combined with other IDS and IPS security tools.

Chen, Luo, Hu, Ye, and Gong (2018) proposed a CAPTCHA recognition attack on hollow CAPTCHA using accurate-filling and merging. A simple and novel framework was developed which supports individual character components and their classification through character segmentation. First, the character contours are repaired through a contour-filling algorithm which is presented to hold together the characters and also a thinning operation. Segmentation comes in thereafter to produce individual characters after which the nearest neighbor algorithm is implemented to obtain a non-redundant individual. Conclusively, CNN (Convolutionary Neural Network) is then introduced to help extract final recognition results. Conclusively, experimental results portrays the proposed method with a very high success rate and a much more superior efficiency in attack compared to the existing typical attack methods that usually follows hollow CAPTCHAs.

Divyashree (2018) opined that in certain environments, the transmission of passwords into the process of verification can help determine the strength of such password. The Authour therefore proposed the combination of two real-time authentication mechanisms, namely cognitive CAPTCHA and honeypot generation, essentially for protecting the humongous volume of information flying all over the internet. The important factor in his work is the development of a cognitive CAPTCHA that is non-reusable as an element within the webpage instead of relying on an external CAPTCHA service provider, building codes and URL's to connect to the service provider for authentication. The major reason for the design of Cognitive CAPTCHA is to have legitimate access and control of Image conversion in real-time, and implementing a honeypot trap incorporation therefore gives a deviation for the spam bots to get trapped thereby eliminating the robot spams. Honeypots and cognitive CAPTCHA are generated in real-time within the website, risks of relying on CAPTCHA service providers, facing denial of services and bearing its cost are completely eliminated.

Souley and Abubakar (2018) proposed a three-layer system. CAPTCHA, a very popular tool used in IDS, was incorporated to prevent spambots from intruding into the system. The technique used is in cognizance to the fact that smart bots exists with the capability of reading and solving CAPTCHA and thereby gain access into the system. Weakly-design patterned and fixed-length CAPTCHAs having varying colours on text was used in a web-form setting acting like IPS while in real sense it functions as IDS attempting to lure spambots using machine learning-based attack to successfully read the text-based CAPTCHA and gain access into the system. Likewise spambots that make use of human solvers unwittingly can easily access the CAPTCHAs and gain access into the system also. The characters in the CAPTCHA are not just to be read and returned back to the system, there is a level of understanding and compliance expected of the users. An example of such CAPTCHA can read "LEAVE TEXTBOX BELOW EMPTY" and a wise human user can easily comprehend that and will oblige the instruction, however, a spambot successfully reads the words and will ignorantly go ahead to type "LEAVE TEXTBOX BELOW EMPTY" or something similar in the textbox. The system quickly captures any typed text in the provided textbox as an intrusion and the intruder is then redirected to the honeypot model for post intrusion activities.

Baykara and Das (2018) proposed a honeypot-based approach, which is feasible on any network security for the real-time IDS or IPS. For this proposed novel approach, there was need for the creation of an effective software tool. The developed system encapsulates a hybrid honeypot that unites the individual properties of both high and low interaction honeypots into a single, harnessed structure. The developed product was tested on a simulated campus network in real time with cogent results obtained. In their comprehensive study, several honeypot components which includes usage viewing, reduction of installation, configuration, maintenance and management cost of the underlying virtualization technologies have been used. Also, the software interface allows for network traffic monitoring in an animation view. Thus providing information about the system in a much more convenient and user-friendly way.

Chen et al. (2019) developed a two-staged Deep Convolutionary Neural Network (DCNN) model that adopts the Selective Learning Confusion Class (SLCC) for its CAPTCHA recognition, although it is predominantly text-based CAPTCHAs. In contrast to other models, accuracy was significantly improved by increasing the training of confusion class samples on Deep Convolutionary Neural Network. Going this route meant a confusion relation Matrix needs be constructed that focuses on relationship between classes rather than the volume of obfuscating characters each class presents. A set partition algorithm was introduced and this algorithm divided various subsets determined by confusion relation matrix. In order to improve the CAPTCHA recognition accuracy of confusion classes, a training and validating learning algorithm was proposed. The experimental results based on the real CAPTCHA data sets show that compared with the existing methods for CAPTCHA recognition, the proposed method delivered higher success rate than its counterparts.

Yu and Darling (2019) illustrated an approach which is low-cost and leverages the nature of open source libraries to attack AI-based focused plaintext CAPTCAHs. Very large samples were created from the web-available open source libraries of Python CAPTCHA and then modified to substitute the profile of the character as placed in the image. This action is followed by a segmentation process introduced by means of a customized Convolutionary Neural Network (CNN) via peak segmentation. A procedure known as TensorFlow Object Detection is then used to identify the characters available in the segmented samples. Experimental results proved that the proposed ToD procedure (TensorFlow object detection) when combined with CNN (Convolutionary Neural Network), i.e. TOD+CNN, the hybrid model easily breaks open-source CAPTCHA libraries. Furthermore, Claptcha-like CAPTCHAs such as Delta40 benchmark were also found to be broken.

Ma et al. (2019) presented an adaptive median filtering algorithm that recognizes and breaks CAPTCHA using Divide and Conquer. In their research, the filtering window data was first sorted through correlation and this helped improve efficiency. Next, the size of the filtering window is adaptively readjusted and this helps eliminate the noise density. Their work achieved a superior performance when compared with the conventional median filter. However, a major bottleneck in their experiment lies in the denoising effect, which is simply very poor blurry or stroked images (Gimppy images, 2D, 3D and 4D CAPTCHAs). Denoising on blurry images has a side-effect which causes faulty or wrong CAPTCHA recognition. In recommending for future work, the Authours suggested that the filtering performance can be significantly improved by bringing it the under higher noise intensity amidst severe distortion.

Zhang et al. (2019) focused more on the subject of the kind of individual that it making an attempt at solving a CAPTCHA, a user or an external entity (which they commonly called a Typer). CAPTCHAs these days are being introduced into unrelated and strange web applications presenting to ignorant users (Typers) to solve them unknowingly. This research work therefore looked into the possibility of introducing private (login) data absolved into the CAPTCHA and in the process break the symmetry between crackers and the Typers. First, an indepth analysis of the online CAPTCHA solving procedure was carried out and this birthed the principle for prohibition of the cheating individuals. Next, the framework was tested with two live scenarios and the experimental result yielded the incorporation of a generation algorithm which tolerates any error made by human thereby complicating matter for Typers. Worthy of note is that Typers are known for their ability to randomly guess attack with a very low success rate, but on the other hand, an actual human user (with intent) can accurately solve CAPTCHA in a few seconds. Experimental results shows usability and the security results are better than the state-of-the-art of CAPTCHA method.

Rushikesh (2019) presented a proposal to combine the features, functions and methodology of IDS, IPS and Honeypot. This was in a bid to increase the accuracy, effectiveness and responsivity of IDS. The combination of Honeypot, IDS and IPS was eventually deployed on the network gateway for the purpose of analyzing incoming network traffic. The main server was connected to an ISP through an external router. All the incoming network packets were first redirected to the dummy server i.e. the Honeypot, for the purpose of capturing the logs. The work resulted in proving the stability and precision at the operating system platform. The system also introduced a much sophisticated and interactive friendly interface for network configuration and software monitoring to help with analysis and behavioural log of the events perpetrated by the intruder. The sole drawback was that the IDS system in its detection module did not include the mechanism behind the detection of spyware intrusions since it is evident that CAPTCHA on the IPS can be broken by spywares.

The availability of CAPTCHAs in this kind of system naturally becomes a problem for spambots, as CAPTCHAs are wholesomely seen as IPS and hence, its presence in the system will naturally be interpreted as an IPS system. This system-faking makes spambots to believe its IPS may not be that effective as some spambots may ignore the CAPTCHA and the response box provided for it in its entirety. This singular action will then process as the correct response expected of users and directs the intruder to the appropriate login authentication. This is an instance where the system may theoretically be bypassed and subsequent researches on similar issues are recommended studying this gap.

2.4 Empirical Review

From the works reviewed, it becomes clear that Human Interaction is the foundation for web application and hence the security of web application can only be tied to Human Interaction Proofs (HIPs). HIPs are procedures that distinguishes a specific class of humans from computer bots or programs over a network. They can be designed to differentiate a human from a computer, a class of humans from another class or one particular human from another human. In doing this, the challenges are presented that appears easy for that class of humans to pass, yet very much difficult for computers to pass. The results from attempts at these challenges must be proved by a computer, and the protocol must be publicly available (Banday and Shah, 2011).

2.4.1 Review of some Methodological Approaches

Over the years, as more and more security mechanism are been introduced and broken, many more keeps evolving. With these evolutions also comes innovations and inventions. The more computer programs and bots becomes more intelligent in solving CAPTCHAs, humans tried several other approaches in enhancing the security of these web applications. In this section, an overview is presented on some of the innovations in ensuring a secure CAPTCHA mechanism and the approaches the Authours took.

Table 2.1: Methodological approaches of some Authours

SN	AUTHOUR and TITLE	METHODOLOGY	APPROACH
1.	Pope and Kaur (2005) **Title:** *Is it Human or Computer? Defending e-Commerce with CAPTCHA*	Authours proposed an image composed of letters and numbers that are pseudorandomly placed either as run through some degradation algorithm or in directly before an obfuscating background to make optical character	The method is very much exposed to blind-guessing, which is not in any way different from a dictionary attack.

		recognition (OCR) of the final image impractical.	
2	Mohammed (2008) **Title:** *Question-based CAPTCHAs*	A less-complicated mathematical problem is presented based on a predefined pattern, however, rather than the name of an object, images are presented. The whole problem is afterwards saved and displayed in form of an image to the user for an attempt by user. However, considering an attempt at the problem requires four distinct human abilities of understanding question context, detecting images of question, problem understanding and problem solving.	User must be knowledgeable in solving computational problems.
3	Kluever and Zanibbi (2008) **Title:** *Video CAPTCHAs: Usability vs. Security*	Authours presented an attempt at `tagging' (or video-labelling) as a CAPTCHA task. Acceptable responses are defined by the predefined tags provided by the video creator and made available via a public database, such as YouTube.com, alongside video tags believed to be somehow linked or related in the database.	An issue arises with the public availability of tag frequency estimates. Furthermore, distortion within the network poses another form of hindrance here.
4.	Banday and Shah (2009) **Title:** *Image-Flip CAPTCHA*	The authours presented a new clickable image-based CAPTCHA technique. The work presented user with a CAPTCHA image and sub-images. Properties of the proposed technique includes; it grants highly improved security than that of usual OCR-based	The framework is believed to have presentation dimension issues as well as that of distortion.

		techniques, also, it consumes less webpage area while improving the webpage's user-friendliness.	
5.	Yamamoto et al. (2010) **Title:** *CAPTCHA using strangeness in Machine Translation.*	This authours focused on strangeness in machine-translated sentences proposing Strangeness in Sentences CAPTCHA (referred to as SSCAPTCHA), which helps in detecting malwares by confirming if users can correctly distinguish machine-generated sentences from human-created natural sentences.	Relay-attacks is a method that currently breaks the SS-CAPTCHA.
6	Hindle et al. (2010) **Title:** *Reverse-Engineering CAPTCHA*	Used bitmap comparison, dilation, fill-flood segmentation and erosion in solving CAPTCHAs. The authour presented white-box and black-box methodologies in a reverse engineering framework for solving CAPTCHAs.	The framework only leverages on image-based CAPTCHAs which makes it susceptible to attacks from bot-programs directly targeted at them.
7.	Almazyad et al. (2011) **Title:** *Multi-Modal CAPTCHA: A User Verification Scheme*	Proposed a new technique to build a multi-modal CAPTCHA comprising of both image and text-based. A displayed image on the screen coupled with many text labels drawn all over it. A user is then given the task of identifying the correct name of the exact displayed image amongst the scattered layer of text labels all over it.	Humans frown at additional tasks of solving multiple CAPTCHA tests especially when the tests are very difficult to solve.

8.	Nassar and Miller (2012) **Title:** *Method for Two Dimensional Honeypot in a Web Application*	Authours proposed using two dimensional honeypot for security access which removes the vitality of form scanning automation so that the spider cannot identify the exact form or field acting as the honeypot. This work centres on limiting the ability of the bot in determining the honeypot.	Scanners of application vulnerability are not meant to pass the pages protected by this solution.
9.	Nguyen (2014) **Title:** *Contributions to Text-based CAPTCHA*	Presented a prototype of that successfully breaks a large number of existing real world schemes. Publications resulting from the Thesis pioneers method of breaking animated 3D or 4D text-based CAPTCHAs.	The model only focuses on Text-based CAPTCHAs and bots program specifically targeted at those still makes it vulnerable.
10.	Woo et al. (2014) **Title:** *3DOC: 3D Object CAPTCHA*	In this work, a novel 3D object based CAPTCHA scheme was proposed that projects the CAPTCHA image over a 3D object. The prototype was developed and proof-of-concept was presented of 3D object based CAPTCHA scheme to protect websites against automated attacks.	Vulnerable to vision paired programmes that are state-of-the-art.
11.	Abinaya et al. (2015) **Title:** *Anti-Phishing Image Captcha Validation Scheme using Visual Cryptography*	The authours lightened the concept of the important features that separates websites used for phishing from the ones that are not whilst assessing the viability of rule-based data mining classification techniques in the prediction of phishing websites while providing more information on which of the techniques is reliable enough.	Layered-images or transparent images are a requirement in revealing the hidden information.

12.	Okesola et al. (2015) **Title:** *Towards the development of a Time-out multiple CR CAPTCHA Framework using Integrated Methematical modeling*	Authour proposed a Time-Out Multiple Challenge-Response (C-R) CAPTCHA Framework that Utilizes Mathematical Modelling as a basis for overcoming some of the challenges faced by current CAPTCHA Systems. Our approach ensures security during the authourization and authentication process.	Speed of form-filling and/or submission cannot be an accurate yardstick for distinguishing human interaction from bots.
13.	Mohammed (2016) **Title:** *Making defeating CAPTCHAs harder for Bots*	Authour provided a high-level comparison of effectiveness of safeguards in addressing the threat posed by CAPTCHA-defeating techniques. In order to focus on usable safeguards, attention was restricted to those which have minimal adverse effect on the user experience.	By chance or by luck, a bot program might find the CAPTCHA it was programmed to solve which still makes the work vulnerable.
14.	Divyashree (2018) **Title:** *Secured Conversion and Generation of Cognitive CAPTCHA Implementing Honeypot Technique*	Some environments allows for systems to be deemed formidable provided the password is blocked from being transmitted to the verification process. Two real-time mechanisms for authentication were proposed by the authour - cognitive CAPTCHA and honeypot generation which aids in protecting the humongous volume of information accessible over internet.	The performance only allows for a measurement coming from the server and not the client side
15.	Chen et al. (2019) **Title:** *Selective Learning Confusion Class for Text-Based*	The Authours constructed a confusion relation matrix to show relations among classes. Using the partition algorithm, a confusion class set was divided into	The more confusing the CAPTCHA is, the higher the frustration it

	CAPTCHA Recognition	multiple sub-sets based on confusion relation matrix. An interactive learning algorithm was proposed to improve the recognition accuracy.	brings to real human users of the system.
16.	Zhang, Hei, and Wang (2019) **Title:** *Typer vs. CAPTCHA: Private information based CAPTCHA to defend against crowdsourcing human cheating*	This work tends to decipher who exactly is solving CAPTCHAs, Human or Human? That is, human-users or human-typers? The Authours devised a generation algorithm to add the capability of human error tolerance and the difficulty of random guess attacks for protecting the humongous volume of information being made available over internet.	The more quacks (readily available software developers) are contacted to develop these mechanisms, the more vulnerable the CAPTCHA security becomes.

2.4.2 Gaps identified in Literature

In reviewing these literatures, some gaps were identified in single mechanisms such as CAPTCHAs and Honeypots, some of which are herein presented in Table 2.2.

Table 2.2: Literature gaps

SN	SECURITY MECHANISM	CATEGORY	DRAWBACK
1.	Text-based CAPTCHA	OCR	1. Regarding text and images, user experiences identification problems ranging from: • Blurry alphabets or numbers • More than one fonts. • Size of fonts. • Motion waves. 2. It can be easily identified by OCR techniques.
2.	Image-based CAPTCHA	OCR	1. Image identification is a major problem faced by users who are either visually-impaired or the quality of the images themselves.
3.	Audio-based CAPTCHA	Non-OCR	1. Users must have a firm grip of the English language vocabulary. 2. Some characters are known to be pronounced similarly. 3. Intonation problem
4.	Video-based CAPTCHA	OCR	1. Issues with size of file being overtly large hence the slow download rate of such videos poses a great challenge.
5.	Puzzle-based CAPTCHA	OCR	1. Two problems suffices here, first solving of the puzzle which will require users having a firm grip and knowledge of the puzzle as well as the length of time it takes to correctly solve such puzzle.
6.	Semantic CAPTCHA	Non-OCR	1. User must have a level of English proficiency. 2. A level of knowledge depth of environment.
6.	Honeypots	Non-OCR	1. Bots that are specifically targeted at Honeypots can be successful.

2.5 Summary

This chapter presented the essential ideas of this study; basically, layering and hybrid approaches for Intrusion Detection Systems, Intrusion Prevention Systems, CAPTCHAs and Honeypots with current development reviews in the literature to establish the gaps for the previous works. The next chapter gives a detail report on the methodology implemented in this study.

CHAPTER THREE

METHODOLOGY

This chapter discusses the research methodology process and strategies that outlines the study, including the identification of methods to be used. The tools and technologies used in the process of the hybrid technique are presented. The features of the proposed framework are shown with necessary procedures.

3.1 The Datasets

In this study, the VAST 2011 Mini Challenge 2 dataset for the real packet traces is publicly available, it was retrieved from a network data repository packets, and can be accessed at: http://www.vacommunity.org/tiki-index.php?page=VAST+Challenge+2014%3A+Mini-Challenge+2andok=yandiTRACKER=1#wikiplugin_tracker1

In addition to the network data, access has been given to the personal and business credit and debit card transactions for the local GAStech employees for the two weeks preceding the incident that the data is comprised of.

Other datasets were obtained from Saad et al. (2011) as well as Chen, Challita, Saad, Yin, and Debbah (2019).

Table 3.1 below shows online repositories for datasets that have over the years been used for CAPTCHAS.

Table 3.1: Online repositories available datasets

SN	DOMAIN	DATASET	# of FEATURES	SIZE	# of CLASSES
1	Image	PIX10P	10000	100	10
2	Image	ORL	1024	400	40
3	Image	JAFFE	676	213	10
4	Image	PIE10P	2420	210	10
5	Image	COIL20	1024	1440	20
6	Text	oh15	3100	913	10
7	Text	tr11	6429	414	9

3.2 Research Design

Cybersecurity is a field of computer science requiring continuous efforts to further advance various computer models and techniques gleaned from trained data. In this study, a hybrid of both CAPTCHA and Honeypot as a layered technique is suggested for the web application security.

This work incorporates JQuery, Javascript and JESS rules alongside the complete SSH protocol, Diffie-Hellman key exchange algorithm and Hidden Markov Models, followed by a comprehensive performance analysis carried out in determining its suitability for web application systems in terms of usability, robustness, the speed of execution and so on. The work then compares the results from the new framework against that of the existing framework using three major performance metrics (Usability, Accuracy and Security) to determine the most suitable framework

3.3 Research Design Layout

The Hybrid technique entails an encapsulation of Secure Shell Algorithm (SSH), Jess Rules, Diffie-Hellman Key exchange algorithms and Hidden Markov Model methods. The design works under the following procedures:

i. User data is captured from the front end and passed into the web application server
ii. The client then authenticates itself over the Adaptive Puzzle CAPTCHA.
iii. A successful authentication of the Puzzle CAPTCHA assumes the Hidden Markov Model on State Transitions and passes authentication to the Honeypot.
iv. The data is passed through the Honeypot system to authenticate the decoy field
v. Results in (iv) once unsuccessful is revoked and access is denied
vi. Results in (iv) once successful is encrypted and passed into the SSH
vii. In the transport layer in (vi), the client contacts the server and keys are exchanged using the Diffie-Hellman key exchange
viii. A public-key algorithm (AES) and the hash algorithm for the transmission are also selected.
ix. Results in (viii) is processed through the SSH and processes (v) and (vi) are repeated.
x. Successful processing of (ix) passes the request to the DB for processing.
xi. Results are then analyzed based on some performance metrics (Usability, Accuracy and Security).

3.4 Model/Framework Specification

3.4.1 Existing CAPTCHA Framework

The internet, computers and web applications are actually made up of diverse unique coding languages. These languages, due to the strangeness and how intricate they appear becomes difficult to understand for computers while human can easily decipher them alongside some other slangs common with humans.

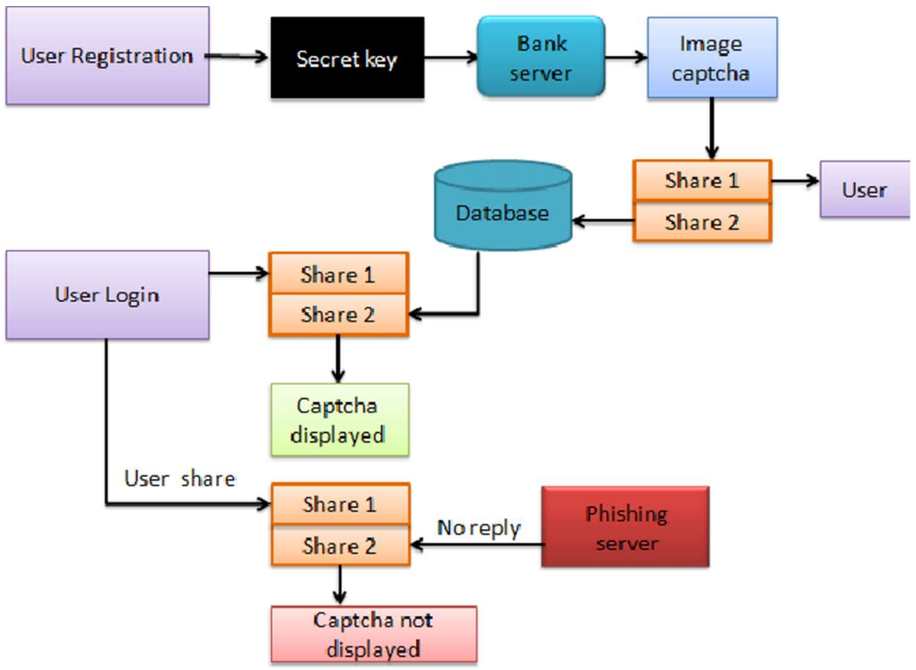


Figure 3.1: System Architecture of Image-based CAPTCHA (Source: Abinaya et al., 2015)

The operations of CAPTCHA begins with an expected interaction either from human users are automated processes (programs referred to as bots) in a form of user-registration exercise as depicted in Figure 3.1. The user-details are then passed through a secret key into a bank server that loads a CAPTCHA puzzle or image to be solved. This puzzle is made available to both the system user and the database and the key is shared for encryption and decryption of the CAPTCHA solution. A successful response to the CAPTCHA puzzle grants access to the system user while an unsuccessful response will lock the user off the system and categorizing the user and its activities as a phishing attack and forwarding activities to a phishing server.

3.4.2 Existing Honeypot Framework

The honeypot, unlike most of the other IDS systems such as firewalls and spamicide, is not set up with the purpose of addressing a specific problem, rather, it is setup as an information tool providing context and understanding to the inherent threats to a web application and in the process, detect the emergence of new threats. From the analysis of the data packets received by the honeypot, a web administrator can then prioritize and focus more on solving security loopholes.

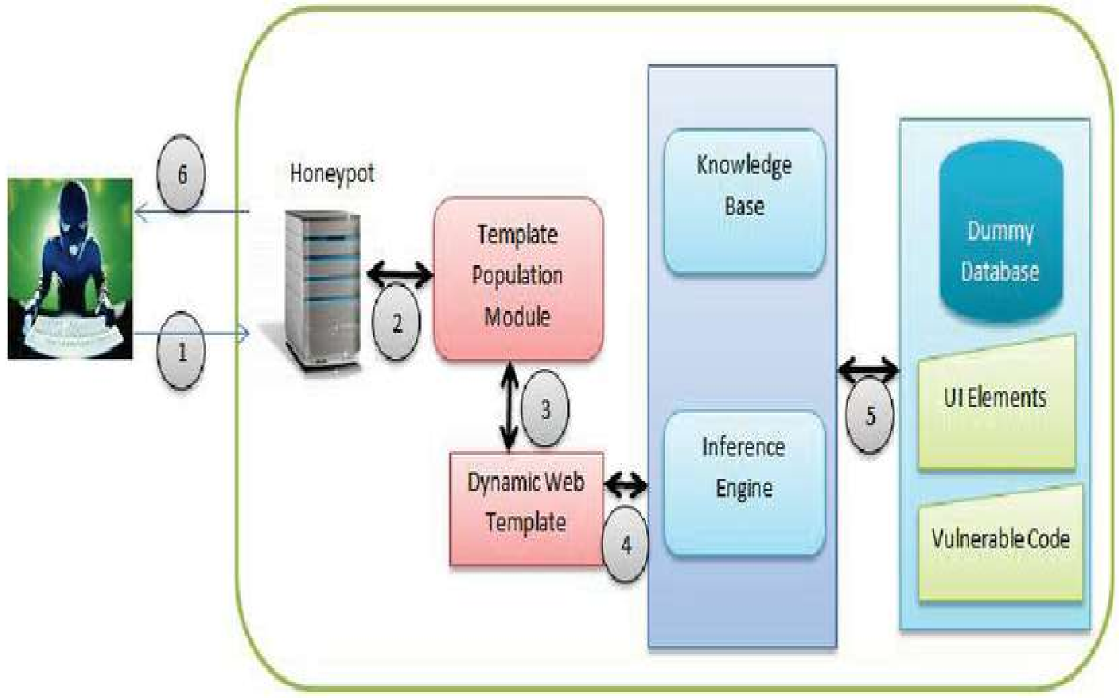


Figure 3.2: Generic Architecture for Web Page Generation (Source: Mphago, 2017)

The way the Honeypot works is that it sits in between the real system and the user acting as a decoy as well as a sensor and when the user interacts with the system, if malicious intent is detected, the decoy system harvests the operation into the Knowledge base as well as inference engines but does not pass data packets to the Database, but if no malicious intent is detected, a legitimate user is granted access to the systems' utilities as shown in Figure 3.2.

Honeypots are deliberately presented attractive to attackers by some inherent system security vulnerabilities, such as open ports (for scanning access) as well as weak passwords. Usually, these ports are left open to entice attackers into the honeypot environment, as opposed to the more secure live network.

3.4.3 Proposed Hybridized Framework

From the existing framework in Figure 3.1 and 3.2, the existing framework is evaluated, adapted and restructured in a hybridized form (as Figure 3.3a depicts) to fit into the software (Landmark University Web Application) that is used to carry out the experiments. Based on the nature of adaptive security mechanisms selected, the parameters of the hybridized framework is then tested accordingly. Once input has been received from external source (human or computer program), it is passed through a Secure Shell (SSH) which sends the Data from one layer (transport layer, user authentication layer and connection layer) to another.

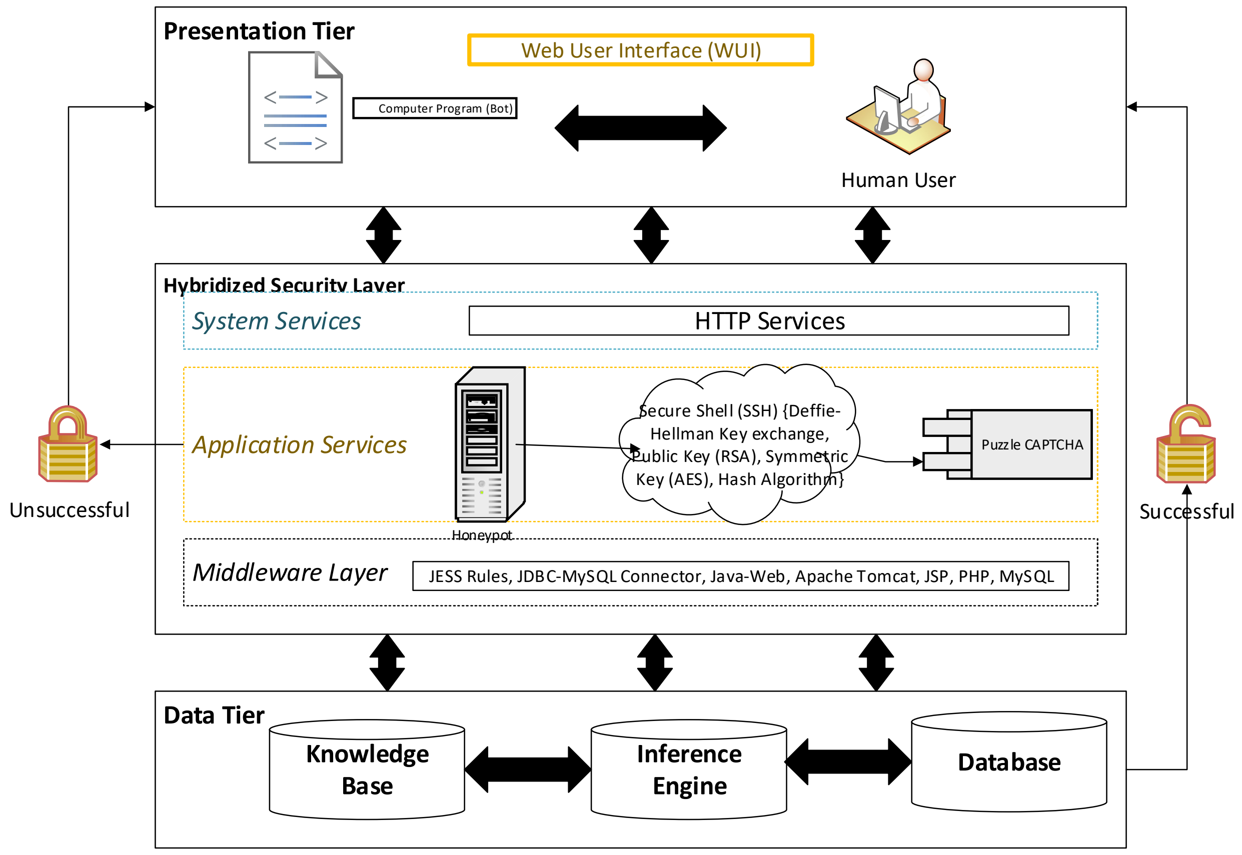


Figure 3.3a: Proposed Hybridized Adaptive CAPTCHA and Honeypot Framework

The transport layer utilizes the Diffie-Hellman key exchange Algorithm to contact the Server and a public key is generated using an AES and a Hash Algorithm for the transmission (Ylonen, 2006). The result is analyzed to examine the effect of the layered security framework on the Web Application defense system.

Security and privacy have become a major talking point in the advent of the Web of things, hence the need for an optimized protocol to aid the security mechanisms in web application systems. The existing asymmetric security mechanisms is thoroughly examined and a new symmetric mechanism is designed to take care of the shortfall of existing protocol.

This study looks intently at each of the layers and how the work utilizes the tools in achieving the unified aim and goal of the study.

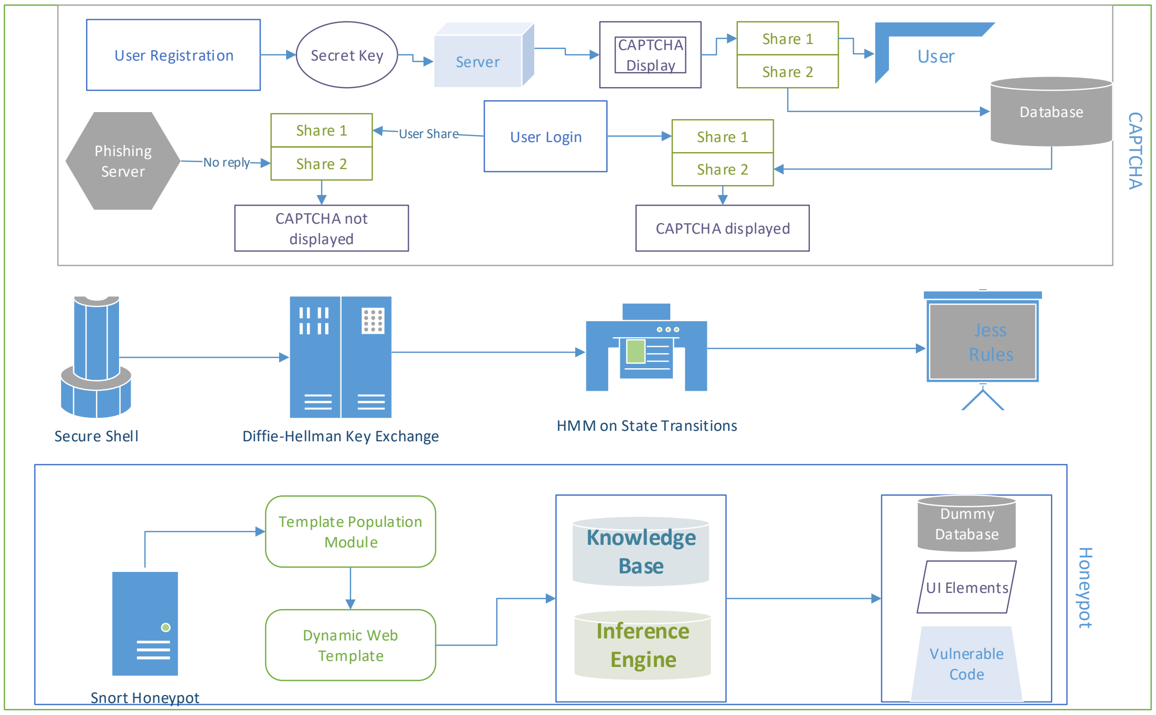


Figure 3.3b: Generic Architecture of Hybrid System

The hybridized security architecture is as shown in Figure 3.3b above. The architecture shows the interaction between the CAPTCHA system and the Honeypot system as well as how they combine.

3.4.3.1 Presentation Tier

The Presentation Tier is essentially the Web User Interface (WUI or GUI as it is commonly referred to). This is the layer that users (human or bots) directly interact with. Tools used in developing this tier includes HTML5, Javascript, Java Server Pages and JQuery. These tools were used to develop a user interface that any user will comprehend and have the ability to interact with. Essentially, the tools were used in creating a web-form which requires only the user access/authentication details as represented on virtually all web applications that requires user registration and any form of interaction. For the purpose of this simulation experiment, interaction has been limited to entering of username and password only. A pictorial representation of the design of the presentation tier is shown in Figure 3.4.

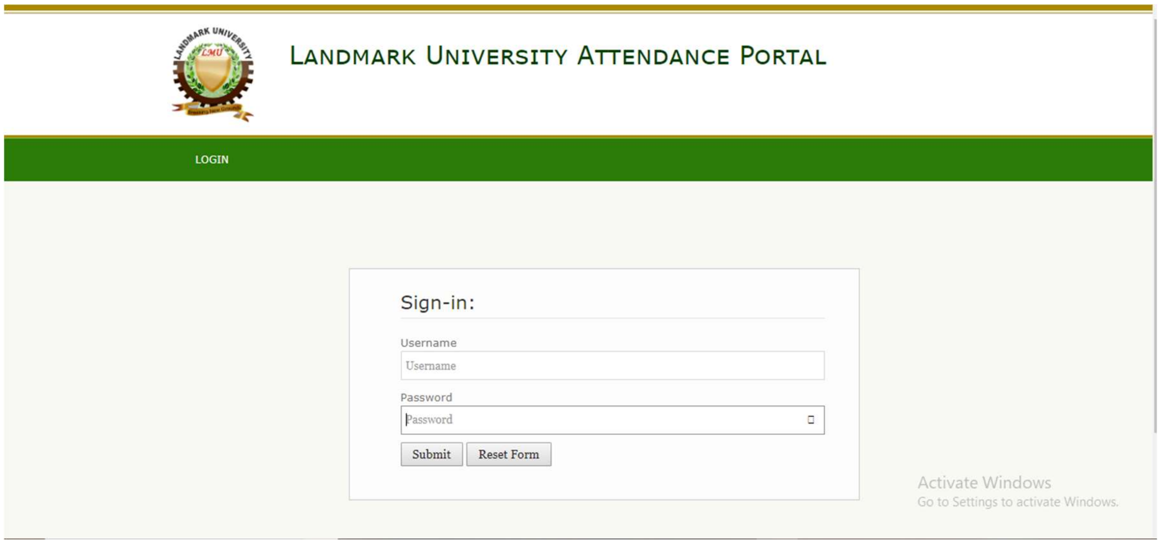


Figure 3.4: The Presentation tier development

3.4.3.2 Hybridized Security Layer

The hybrid layer is further subdivided into three broad layers namely; the System Services, the Application Services and the Middleware.

3.4.3.2.1 System Services

A key feature of this service is ensuring that whatever language in which the Presentation tier (WUI or GUI) is written in as depicted in Figure 3.5 below, it is able to communicate first with the web browser through exchange of data between clients and servers via a web service. The system user calls a web service through an HTTP or XML request, and the service receives an HTTP or XML response. These services are also often associated with

SOA (Service-Oriented Architectures). This process is carried out in six basic steps namely;

Step 1: User input directs browser to a given Universal Resource Locator (URL).

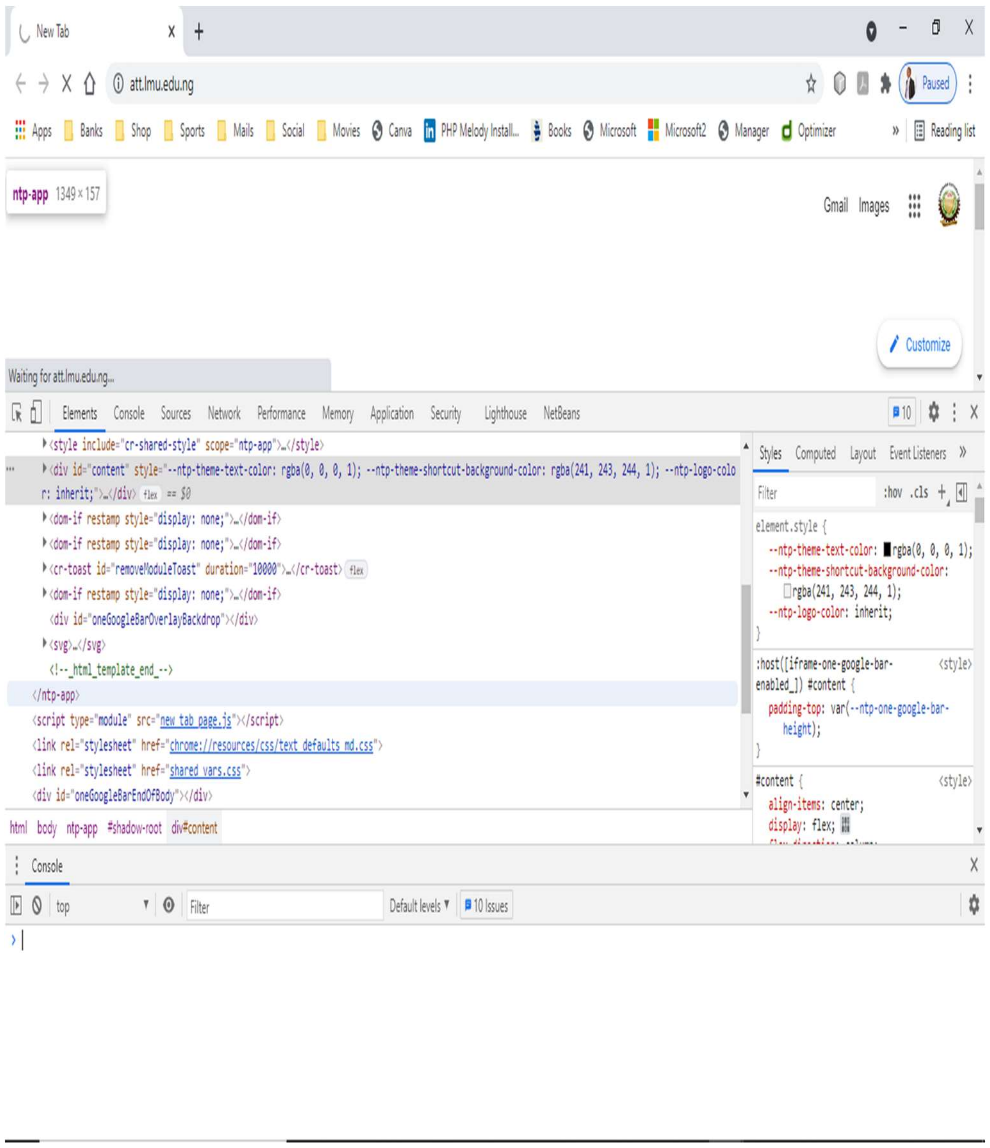
Step 2: Browser looks up Internet Protocol (IP) Address on a Domain Name Server (DNS)

Step 3: Browser sends a Hyper-Text Transfer Protocol (HTTP) request.

Step 4: Host responds with a HTTP response.

Step 5: The browser presents the response in a readable format to the user

Step 6: HTTP and TCP/IP.


New Tab
att.lmu.edu.ng
Paused
Apps Banks Shop Sports Mails Social Movies Canva PHP Melody Install... Books Microsoft Microsoft2 Manager Optimizer Reading list
ntp-app 1349 × 157
Gmail Images
Customize
Waiting for att.lmu.edu.ng...
Elements Console Sources Network Performance Memory Application Security Lighthouse NetBeans
10
Styles Computed Layout Event Listeners
Filter :hov .cls +
html body ntp-app #shadow-root div#content
Console
top Filter Default levels 10 Issues

```
▸<style include="cr-shared-style" scope="ntp-app">…</style>
▸<div id="content" style="--ntp-theme-text-color: rgba(0, 0, 0, 1); --ntp-theme-shortcut-background-color: rgba(241, 243, 244, 1); --ntp-logo-color: inherit;">…</div> flex == $0
▸<dom-if restamp style="display: none;">…</dom-if>
▸<dom-if restamp style="display: none;">…</dom-if>
▸<cr-toast id="removeModuleToast" duration="10000">…</cr-toast> flex
▸<dom-if restamp style="display: none;">…</dom-if>
 <div id="oneGoogleBarOverlayBackdrop"></div>
▸<svg>…</svg>
 <!--_html_template_end_-->
</ntp-app>
<script type="module" src="new_tab_page.js"></script>
<link rel="stylesheet" href="chrome://resources/css/text_defaults_md.css">
<link rel="stylesheet" href="shared_vars.css">
<div id="oneGoogleBarEndOfBody"></div>
```

```
element.style {
  --ntp-theme-text-color: ■rgba(0, 0, 0, 1);
  --ntp-theme-shortcut-background-color:
    □rgba(241, 243, 244, 1);
  --ntp-logo-color: inherit;
}
:host([iframe-one-google-bar-          <style>
enabled_]) #content {
  padding-top: var(--ntp-one-google-bar-
    height);
}
#content {                             <style>
  align-items: center;
  display: flex;
```

Figure 3.5: Components of the HTTP Services can be seen in action when a page loads

3.4.3.2.2 Application Services

Application Services are a class or group of management functions that aids the discovery, deployment and improvement of an application and how such applications are run. These pool of services revolve around monitoring of application performance, load balancing, micro-segmentation, acceleration, autoscaling, service proxy and service discovery. The services incorporated in this methodology includes the Secure Shell (SSH), Diffie-Hellman Key Exchange Algorithm (DHKA), Hidden Markov Model on state Transitions (HMM) and Jess Rules and this section reviews these components and the function each of them performs in the hybridized solution.

3.4.3.2.2.1 Secure Shell (SSH)

SSH (Secure Shell) is used to manage the networks, the operating systems, and the system configurations. It is also embedded inside the file transfer tools that passes sequence from one configuration management tool to another.

SSH keys enables the automation that makes the attendance server files and other computer-dependent services possible and cost-effective. They offer convenience and improved security when properly managed Also, the SSH key pair generated was used to guarantee continued availability of systems as well as the confidentiality and integrity of the whole process experiment.

The Secure Shell is incorporated into the methodology by providing a secure connection over the entire system network. It allows for a remote connection to the host for a terminal session. In its implementation, SSH was ported to run in Windows PowerShell in our Windows 2016 Datacentre server making it directly accessible by default to all applications running on the server.

The deployment was done using PuTTY, another open source implementation of SSH. It comprises three utilities - slogin (secure login), ssh scp (secure copy) which are secure versions of the earlier insecure Unix utilities: rlogin, rsh and rcp. The deployed SSH server uses public key cryptography in its authentication over the remote computer and when necessary, it enables remote computer user authentication.

The implementation runs through the following steps:

ssh att.lmu.edu.ng: This command connects to the destination, the LMU attendance portal; the destination host responds by prompting for a user logon details under which the client is running.

ssh remote_host_username@att.lmu.edu.ng: This is used when the username for the remote host is different, in which case, the command is expected to be issued with the remote host username.

Other SSH protocol that was used includes;

sshd: This helps initiate the SSH on the LMU Attendance server while waiting for incoming connection requests and enabling successfully authenticated systems to connect to the host server.

ssh-keygen: This command creates a new authentication key pair for SSH essentially for the purpose of automating logins, implementing SSO and also authenticating hosts.

ssh-copy-id: This essentially allows copy, install and configure functions with the aid of an SSH key on the LMU attendance server in automating logins without the use of passwords.

ssh-agent: This helps to tracks identity keys alongside their passphrases with which SSH server derives an encryption key.

ssh-add: When keys are to be added to the SSH authentication agent, this call is used alongside ssh-agent in implementing SSO using SSH.

scp: When files are to be copied from the attendance server to the client computer, this function call is used and it is an SSH-secured version of rcp.

sftp: When files are to be copied from the attendance server to the client computer, this function call is used and it is an SSH-secured version of ftp, the File Transfer Protocol.

3.4.3.2.2.2 Diffie-Hellman Key Exchange Algorithm

Diffie Hellman (DH) key exchange algorithm is the method used for securely passing and transmitting the cryptographic keys generated over a public communications channel. The transmitted keys were not actually exchanged, they are only jointly derived. It is named after their inventors Whitfield Diffie and Martin Hellman.

For the sake of simplicity and practical implementation of the algorithm, we have considered only 4 variables:

i. a prime A
ii. a primitive root of A, tagged B
iii. a private value, x to be picked by a Landmark Student or Faculty
iv. another private value, y to be picked by the Server Administrator.

A and B are both publicly available numbers. Table 3.2 explicitly defines the interaction between client and server as both Users (Landmark user and Server Administrator) pick private values **x** and **y** and they generate a key and send to each other publicly, the recipient received the key and from that generates a secret key after which both the Student and the Administrator possess the same secret key to encrypt.

Table 3.2: Step explanation of how the Diffie-Hellman Algorithm exchanges keys:

Proc	Landmark Student	Server Administrator
1	Public Keys available = A, B	Public Keys available = A, B
2	Private Key Selected = x	Private Key Selected = y
3	Key generated => $i = G^x \textit{ mod } A$	Key generated => $j = G^y \textit{ mod } A$
4	***Key exchange takes places***	
5	Key received = j	Key received = i
6	Generated Secret Key: $k_x = j^x \textit{ mod } A$	Generated Secret Key: $k_y = i^y \textit{ mod } A$
7	***Algebraically it can be shown that $k_x = k_y$***	
8	***Users now have a symmetric secret key to encrypt***	

- ✓ **Step 1**: Landmark User and Server Administrator get public numbers, say A = 23, B = 9
- ✓ **Step 2**: Landmark User selected a private key x = 4 and Server Administrator selected a private key, y = 3
- ✓ **Step 3**: Landmark User and Server Administrator compute public values
  - Landmark User: i =(9^4 mod 23) = (6561 mod 23) = 6
  - Server Administrator: j = (9^3 mod 23) = (729 mod 23) = 16
- ✓ **Step 4**: Landmark User and Server Administrator exchange public numbers
- ✓ **Step 5**: Landmark User receives public key j =16 and Server Administrator receives public key i = 6
- ✓ **Step 6**: Landmark User and Server Administrator compute symmetric keys
  - Landmark User: $k_x = j^x$ mod A = 65536 mod 23 = 9

- Server Administrator: $k_y = i^y \bmod B = 216 \bmod 23 = 9$

✓ **Step 7**: 9 is the shared secret.

3.4.3.2.2.3 Hidden Markov Model (HMM)

The Hidden Markov model (HMM) gives us an insight into both observed events. Our HMM is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N states
$A = a_{ii} \ldots a_{ij} \ldots a_{NN}$	a transition probability matrix A, each a_{ij} representing the probability of moving from state i to state j, s.t. PN j=1 $a_{ij} = 1$ ∀i
$O = o_1, o_2 \ldots o_T$	a sequence of T observations, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
B = bi(ot)	a sequence of observation likelihoods, also called emission probabilities, each expressing the probability of an observation ot being generated from a state i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an initial probability distribution over states. π_i is the probability that the Markov chain will start in state i. Some states j may have πj = 0, meaning that they cannot be initial states. Also, Pn i=1 $\pi_i = 1$

For our HMM represented by P hidden states and an observation sequence of V observations, there are P^V possible hidden sequences.

For the simulation experiment, where P and V are both large, P^V will assume a very large number, so it will be almost impossible to compute the total observation likelihood by computing a separate observation likelihood for each hidden state sequence and then summing them, rather than indulging such an extremely exponential algorithm, we use a much more efficient $O(P^2V)$ algorithm known as the forward algorithm.

From the numerous state paths which amounts to the same sequence x, it becomes pertinent to add the probabilities for all possible paths to obtain the full probability of x:

$$P(x) = \sum_{\pi} P(x, \pi)$$

where π is an event in which a specific path was taken through the HMM.

This forward algorithm is a dynamic programming algorithm, that is, an algorithm that store intermediate values using a table as it builds up the probability of the observation sequence. This forward algorithm computes the observation probability by adding up the probabilities of all possible hidden state paths that could generate the observation sequence. However, it so efficiently does it by implicitly folding each of these paths into a single forward trellis which then determines the state the sequence is in per time as depicted in Figure 3.6 below.

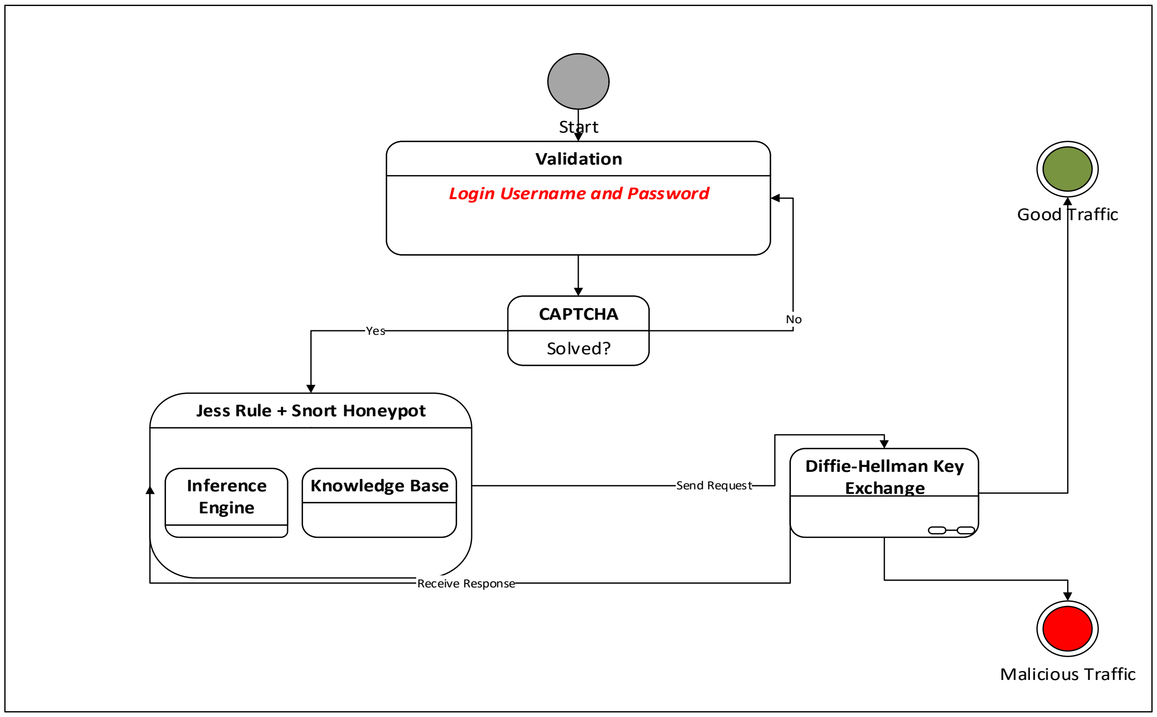


Figure 3.6: State transitions using the HMM algorithm

3.4.3.2.2.4 Java Expert System Shell (JESS) Rules

Jess is a tool for developing expert systems and has features such as editor and debugging tool. The tool helped us decipher the workability of the simulation experiment in ways which includes discerning:

a. Facts which are inclusive memory of data,
b. Knowledge-Base which comprises of rules and
c. Inference engine that carries out reasoning.

The Jess program developed consists of rules, facts and objects and a software can be developed using only rules, objects only or a combination of the two (rules and objects). Jess engine used the rete algorithm to check rules against the working memory. Rete algorithm is a pattern matching algorithm that uses a rooted acyclic directed graph where the nodes represent the patterns and path (from the root to the leaves) represents the left-hand side of the rule (IF part).

The sample Jess rules used for the expert system development is described below:

Pseudocode for the JESS Rule

MAIN MENU RULES

```
; *****
; RULE main-menu
; IF the user initiates CAPTCHA Parser
; THEN Launch CAPTCHA Parser
; IF the user enters username and password
; THEN Prompt the user for solution to a CAPTCHA
; Assign a default scrambled password or username to the CAPTCHA
; ELSE
; Open the default page containing textarea and wait for user’s entry
; Wait on user for entering of login details
; Get the user’s response
(defrule main-menu
(user dails into the system)
=>
(printout
“Welcome username”.
“This is the authentication page:”
“Select one of: Solve CAPTCHA, Reload CAPTCHA, Exit”)
(bind ?choice (read))
(if (= ?choice Solve CAPTCHA) then (assert (choice-is Initiate CAPTCHA Parser)))
(if (= ?choice Reload CAPTCHA) then (assert (choice-is Reload CAPTCHA Parser)))
```

The Rete algorithm is a pattern matching algorithm that is used for implementing rule-based systems such as the one used in this simulation experiment – JESS rules.

The Rete Algorithm is a unique decision engine algorithms preferably used in modern rule engines with the purpose of learning more about decision performance and the engines succinctly driving the automated processes and decision in today’s world. The algorithm operates on some appended conditions and it is depicted in Figure 3.7:

CAPTCHA

status is Solved

status is Unsolved

reload the CAPTCHA if number of selection < number Expected

verify the solution if number of selection >= number Expected

Honeypot

network Packets captured contains message data

network Packets captured does not contains message data network activities found in the Honeypot knowledge-base

new Network activities detected

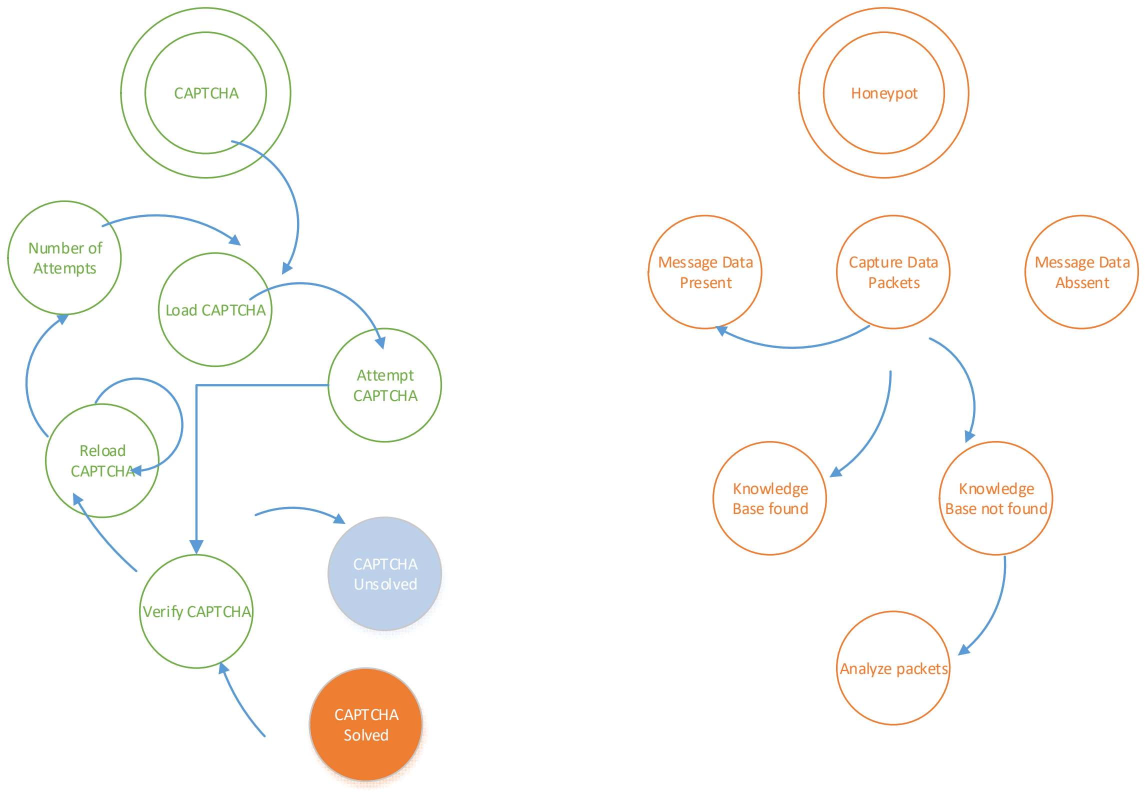


Fig 3.7: Jess Rules implementing the Rete pattern-matching Algorithm

3.4.3.2.3 Middleware Layer

This layer consists of the link-up between our hybrid system and the Data tier. Essentially, it comprises of the JDBC-MySQL Connector and the Apache Tomcat Services.

JDBC-MySQL Connector: This tool was used to facilitate the interconnection of our JESS rules and Java Server pages to the database server where MySQL was implemented.

Apache Tomcat Services: This service provided us with numerous benefits which proved essential when transferring the web application for the simulation experiment from a development set-up to a production environment. Some of the benefits include:

i. Stable automatic startup on system boot.

ii. Startup of the server without user login credentials.

iii. System Security allowing the experiment to run under a special system account, which is isolated and protected from rest of the user accounts.

3.4.3.3 Data Tier

This tier of the framework is essentially for the storage of Data (Database), known web application attack mechanisms (Knowledge Base) and the rules used in drawing inferences to determine if a server activity is rated malicious or not (Inference Engine).

The Knowledge base of the system contains facts acquired from human experts users through the medium of observations and interviews. This knowledge is then represented in the form of production rules ("if-then"): "Should any of the condition be true, then a corresponding action (referred to as "inference") follows". The knowledge base of the hybrid expert system contains several rules. This follows with a probability factor linked to the conclusion of each if-then rule and to the recommendation, so long the conclusion is not certain. For example, a system for the determination of the severity of a captured data packet might indicate, based on information supplied to it, a high risk potentially bad traffic whilst also listing conclusions with lower probabilities. The expert system displays the rules sequencing that leads to the arrived conclusion; A careful study of this flow helps the network administrator to confirm the credibility of its recommendation.

On the other hand, the inference engine enables the expert system to make deductions from the rules in the Knowledge Base. For example, if the Knowledge Base contains the production rules "if captured data packets pattern exists in knowledge base, rate severity of traffic – level 5 (less risk traffic)" and "if captured data packets pattern does not exist in knowledge base, then rate severity of traffic – level 1 (high risk traffic)," from the established production rule, the inference engine makes deductions that translates into a combination of both or more rules.

3.5 Data Collection Techniques

In carrying out the simulation experiment, the methods highlighted below were deployed in discovering the effect and impact of cybercrime as regards breaching of the CAPTCHA and Honeypot hybrid scheme and why it keeps thriving.

Some simulation metrics were engaged in computing the response time of the experiments, some of these metrics include:

i. **Top-k** (k): The accuracy of any experiment can either be higher or remain unchanging. This gives an insight into how the hybrid model works. For example, should the Top-1 Accuracy produce a low outcome, there is a tendency to conclude the hybrid model doesn't have much correlation with the dataset. However, if K accuracy significantly increases, it can be concluded that it is learning albeit needing some fine-tuning. This can be especially helpful for classification problems with a high number of classes.

ii. **Number of QoS attributes** (q): This metric allows for measurement of the quality of the service rendered along some major parameters such as genuineness, tangibility, responsiveness, assurance and empathy. For this purpose of the hybrid model, two distinct dimensions - reliability and responsiveness, have been chosen.

iii. **Number of Trials**: There is a steady progression in the number of times the experiment is carried out. The bulk of the simulation experiment will not all be carried out simultaneously.

iv. **Priority Weight** (w): This is a function of the number of quality of service (QoS) attribute.

v. **Number of successful trial runs** (t): This metric indicates a fraction of the total number of trials that ran successfully to the end.

In listening to these respondents voicing their concerns and opinions, themes were observed as they used the design. The techniques put in place for this phase essentially was **Think-Aloud activity** and **Thematic Analysis**.

A **Think-Aloud Activity** is a strategy that helps thoughts to be verbalized for absolute comprehension and in-depth knowledge. This activity with the selected respondents was conducted for the following reasons:

1. To understand what they know about the CAPTCHA and Honeypot amongst other Intrusion Detection System schemes.
2. To see what they envisage as benefit from the CAPTCHA and Honeypot scheme
3. To see if they understand the design of the Intrusion Detection System schemes
4. To see if they have a clear information about the scheme as they iterate through
5. To see if there are ways they could have a better understanding of the design
6. To point out the most important aspect of the scheme
7. To see how the design fits into what they already know about the scheme.

On the other hand, **Thematic Analysis** seeks to identify themes or patterns within qualitative data (Maguire and Delahunt, 2017). Hence this method was incorporated to identify a common trend or pattern amongst the participants at each and every stage of the usability test of the design as depicted in Figure 3.8 below. The analysis revolves around the following reasons:

1. Examining collated data to identify significant themes.
2. Collating data relevant to each participant's pattern.
3. Work with the data
4. Review the viability of each participant's theme.

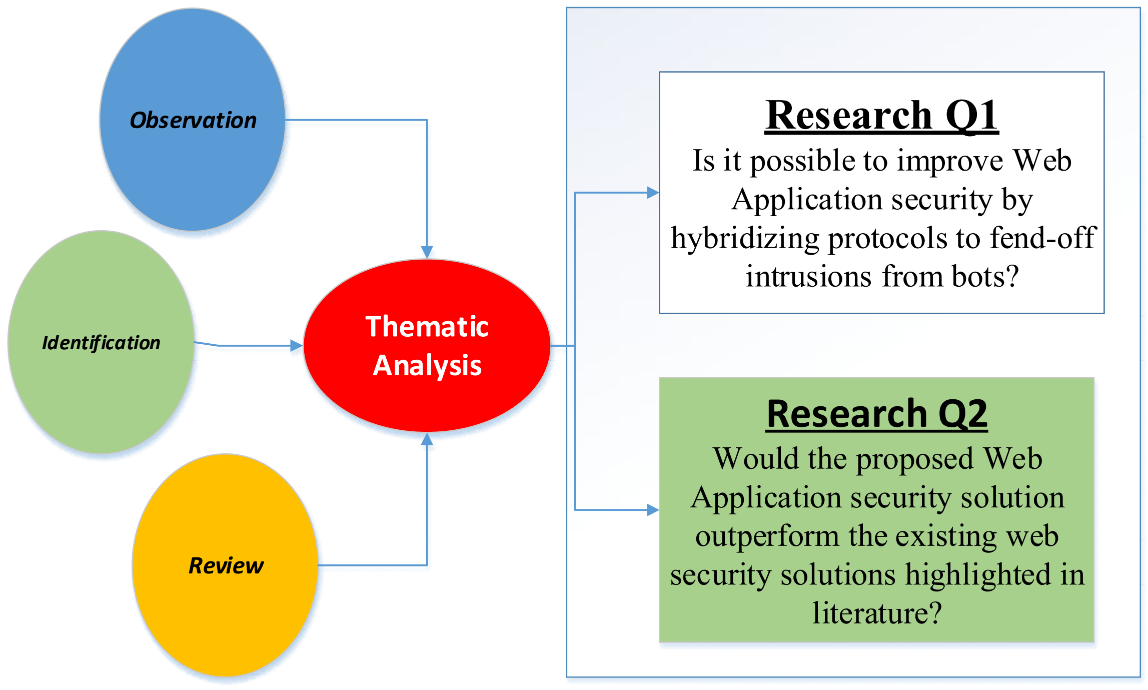

Observation
Identification
Review
Thematic Analysis
Research Q1
Is it possible to improve Web Application security by hybridizing protocols to fend-off intrusions from bots?
Research Q2
Would the proposed Web Application security solution outperform the existing web security solutions highlighted in literature?

Fig 3.8: Thematic Analysis of the developed method

3.6 Hybrid Technique Deployment

3.6.1 Hardware Requirements

Client Devices

Laptops and PCs

Devices in modern times take on a variety of features, significantly being the ability to connect to the Internet via diverse means. With this ability comes variety of features such as virtualization, email, web surfing and so on.

Other Client Devices include:

Processor:	Intel © Core 2 Duo CPU
Hard disk:	512GB or more
RAM:	10GB or more

3.6.2 Software Requirements

ES Shell:	JESS 7.1
Backend:	MySQL and JDBC
Middle Tier:	Java Servlets
Frontend:	HTML5 and Java Server Pages
Web Server:	Windows 2016 Server, Apache Tomcat

3.7 Research Instrument and Tools

Several tools were readily available over the Web that aided this research work with attention focusing on these few:

i. Snort
ii. Secure Shell (SSH)
iii. HoneyBOT
iv. Hardware write Blocker
v. Registry Viewer
vi. JESS
vii. Rete Algorithm
viii. Deffie-Hellman key exchange
ix. Hash Functions
x. Apache Tomcat
xi. JDBC-MySQL Connector

3.8 Performance Evaluation Metrics

This study analyzes the performance evaluation metrics of the hybrid system in terms of usability, accuracy and security.

The system usability evaluation was carried out so as to measure user satisfaction, effectiveness and efficiency. Effectiveness defines precision and completeness with which users can reach stated goals, while efficiency defines the effort and resources required in order to reach goal attainment. User's satisfaction is defined as user's attitude towards using a system.

Accuracy is the possibility that an analytical test is correctly performed.

In carrying out the usability evaluation, questionnaires were designed and administered. The designed questionnaire has three sections, Background information of respondents, User satisfaction, effectiveness and efficiency of the system. The questions were administered via a five-point rating scale where 1 = Strongly Disagree, 2 = Disagree, 3 = Undecided, 4 = Agree and 5 = Strongly Agree. A total of 29 questions were administered and all responses were recorded and analyzed.

The accessibility of the efficiency of Diffie-Hellman Key Exchange and Rete algorithms largely depends on selected and concise validation metrics. The confusion matrix utilizes four major characteristics for evaluating the classification models; True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Examples are discovered and outlined before correctly and incorrectly categorizing from the given dataset samples in testing the model (Karthik and Sudha, 2018).

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)\ \%$$

Where:

TP (True Positives) = correctly classified positive cases,

TN (True Negatives) = correctly classified negative cases,

FP (False Positives) = incorrectly classified negative cases,

FN (False Negatives) = incorrectly classified positive cases.

3.9 Implementation and Screenshots

A Virtual Machine (VMWare Workstation) was used to carry out the simulation experiment as depicted in Figure 3.9:

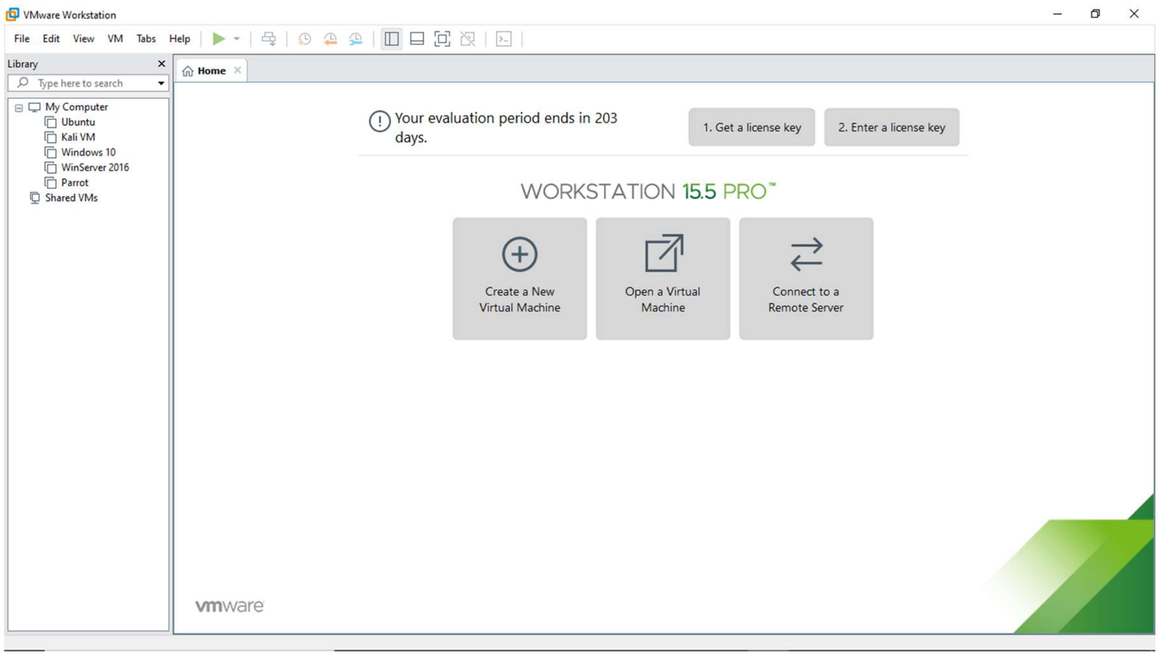


Figure 3.9: Image depicting the installation of the VMWare Workstation.

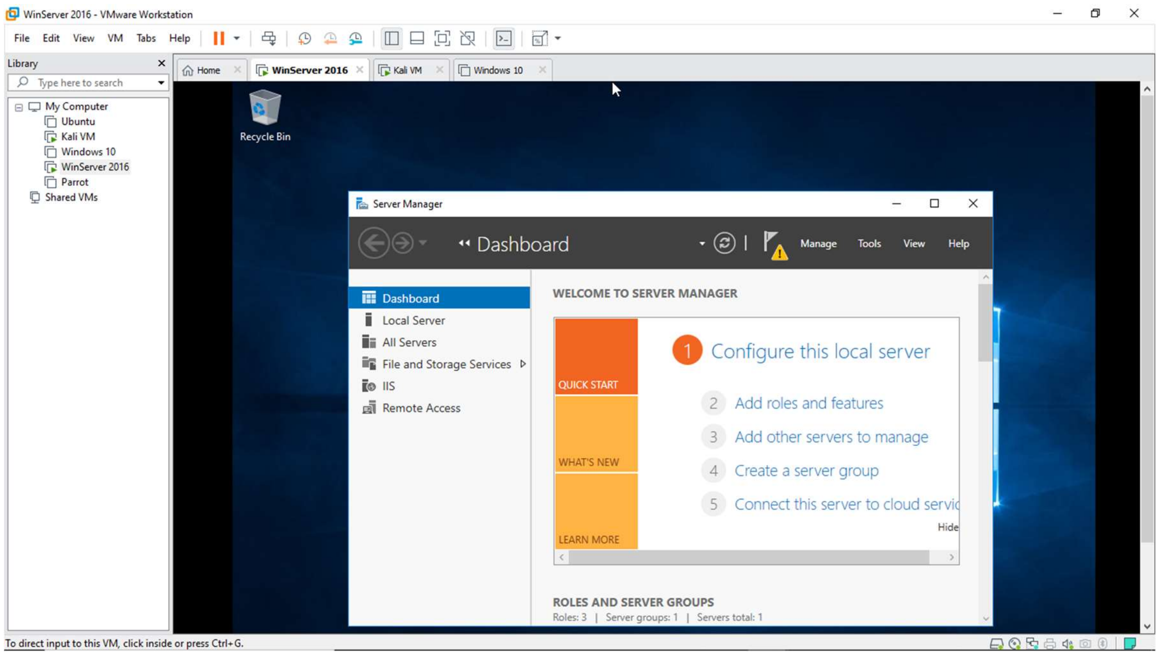


Figure 3.10: Image depicting the installation of the Windows Server 2016.

The objective of this experiment is to help us understand how Intrusion Detection/Prevention Systems is installed and the workings as shown in Figure 3.11 to 3.30. In this experiment, the following will be carried out:

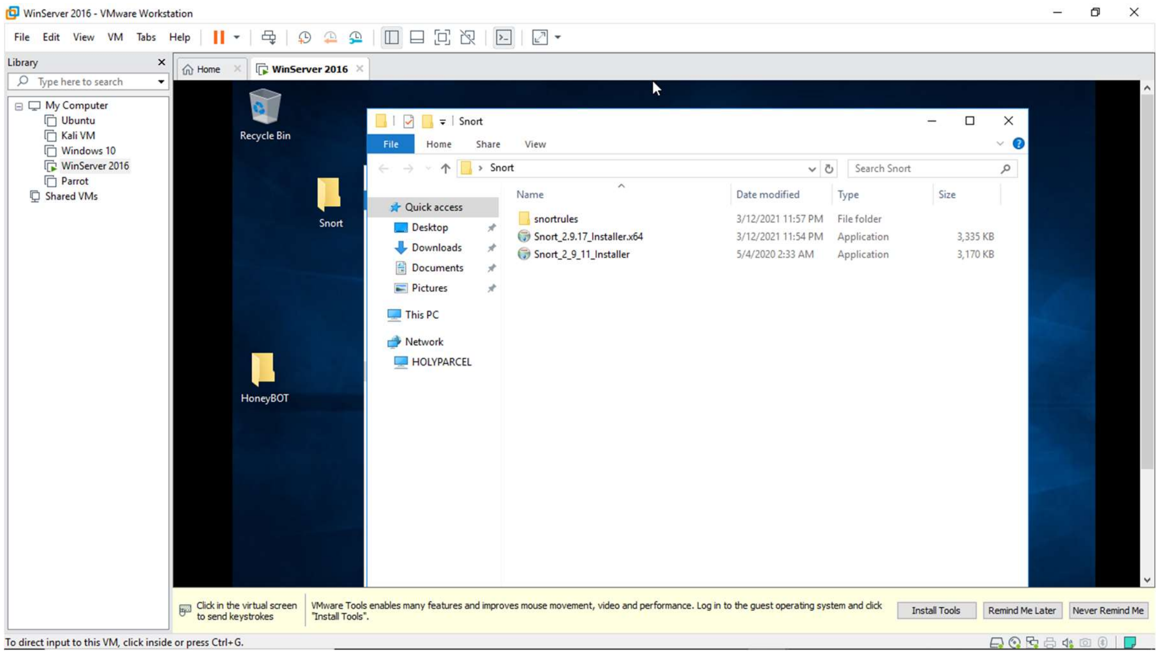
a) Install Snort and verify Snort alerts

b) Configure and validate snort.conf file

c) Test working of Snort by carrying out attack test

d) Perform Intrusion detection

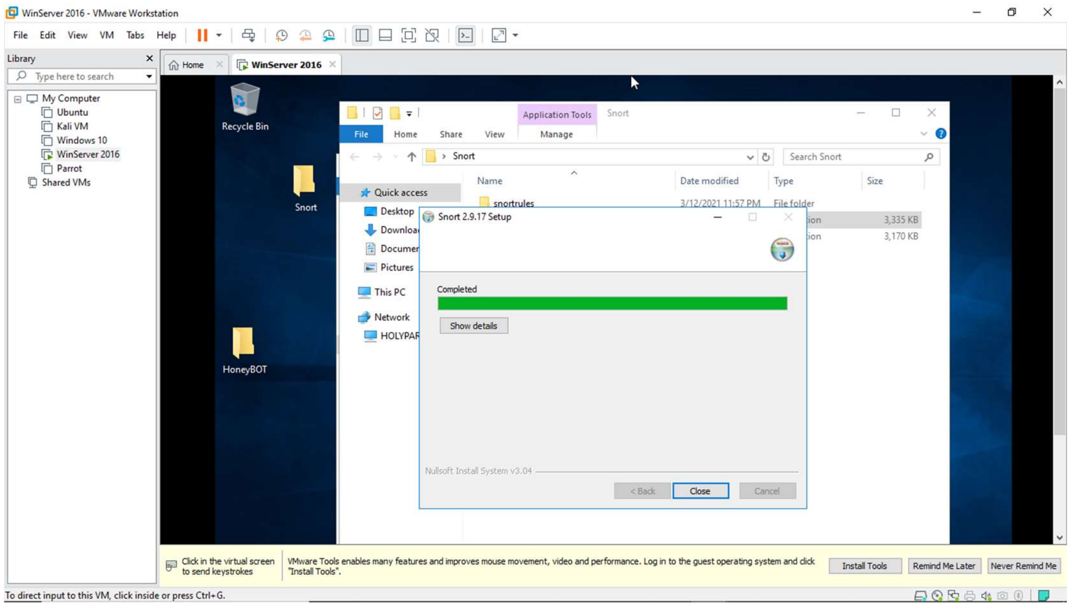
A. Install and verify Snort Alerts



Name	Date modified	Type	Size
snortrules	3/12/2021 11:57 PM	File folder	
Snort_2.9.17_Installer.x64	3/12/2021 11:54 PM	Application	3,335 KB
Snort_2_9_11_Installer	5/4/2020 2:33 AM	Application	3,170 KB

VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".

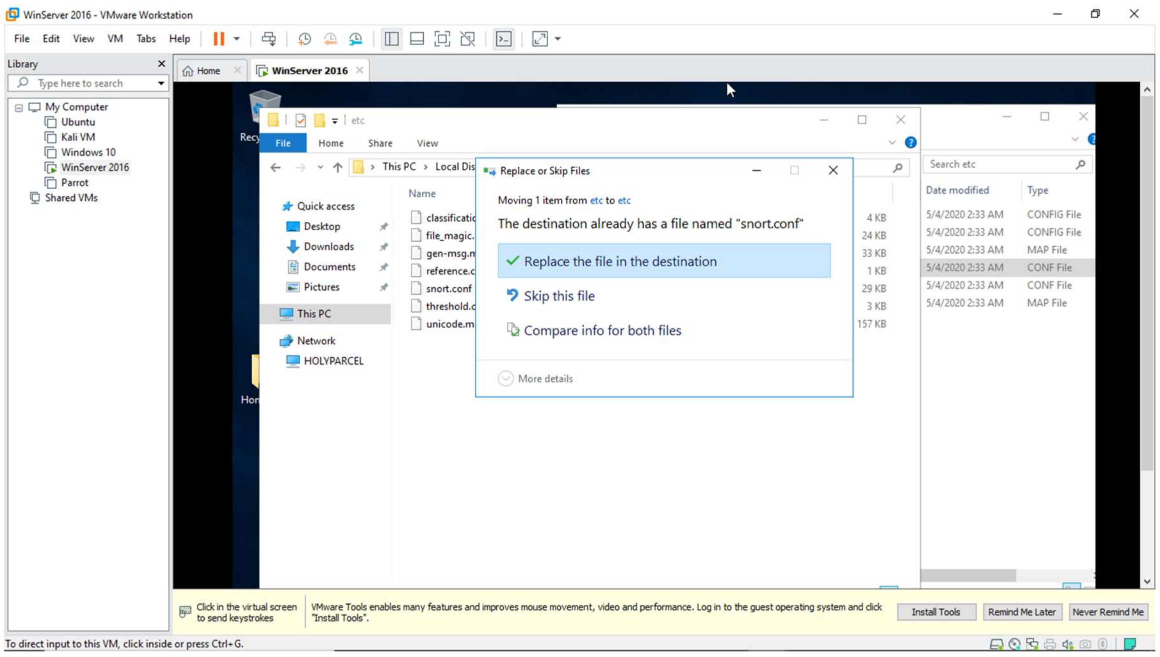
Figure 3.11: Image showing the Snort Honeypot exe files on the Windows Server 2016



Completed

VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".

Figure 3.12: Installation of Snort completed on the Windows Server 2016



Replace or Skip Files

Moving 1 item from etc to etc

The destination already has a file named "snort.conf"

✓ Replace the file in the destination

Skip this file

Compare info for both files

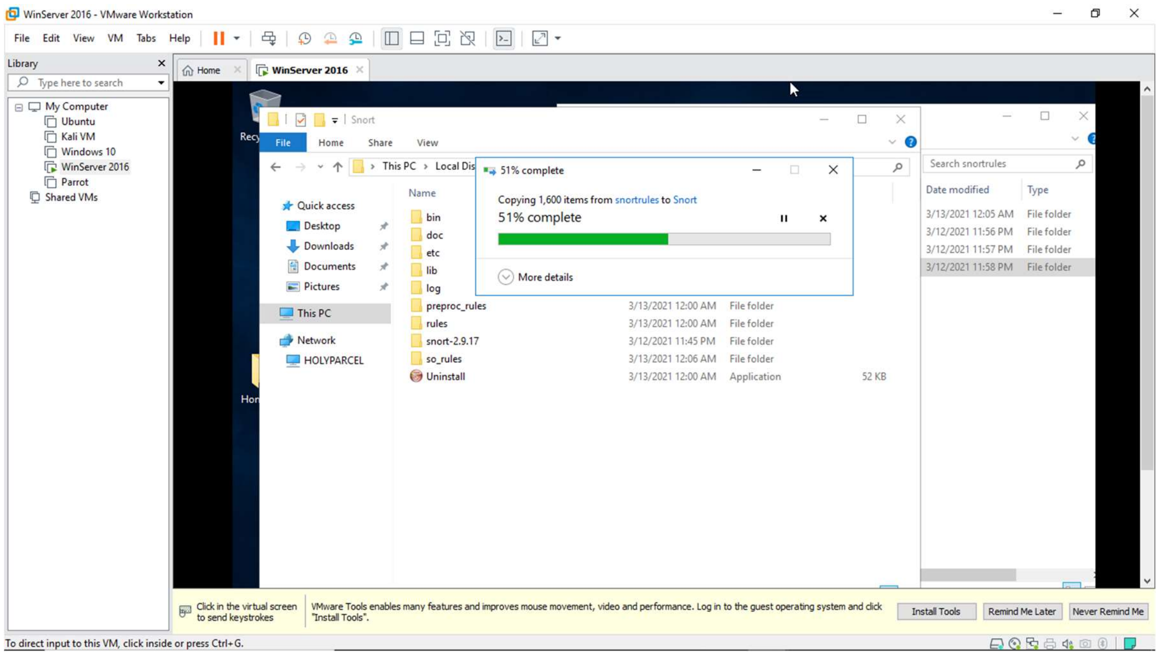
Name	Size	Date modified	Type
classificatio	4 KB	5/4/2020 2:33 AM	CONFIG File
file_magic.	24 KB	5/4/2020 2:33 AM	CONFIG File
gen-msg.m	33 KB	5/4/2020 2:33 AM	MAP File
reference.c	1 KB	5/4/2020 2:33 AM	CONF File
snort.conf	29 KB	5/4/2020 2:33 AM	CONF File
threshold.c	3 KB	5/4/2020 2:33 AM	MAP File
unicode.m	157 KB		

Click in the virtual screen to send keystrokes

VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".

To direct input to this VM, click inside or press Ctrl+G.

Figure 3.13a: Snort rules verification on the Windows Server 2016



51% complete

Copying 1,600 items from snortrules to Snort

51% complete

Name	Date modified	Type	Size
bin	3/13/2021 12:05 AM	File folder	
doc	3/12/2021 11:56 PM	File folder	
etc	3/12/2021 11:57 PM	File folder	
lib	3/12/2021 11:58 PM	File folder	
log			
preproc_rules	3/13/2021 12:00 AM	File folder	
rules	3/13/2021 12:00 AM	File folder	
snort-2.9.17	3/12/2021 11:45 PM	File folder	
so_rules	3/13/2021 12:06 AM	File folder	
Uninstall	3/13/2021 12:00 AM	Application	52 KB

Click in the virtual screen to send keystrokes

VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".

To direct input to this VM, click inside or press Ctrl+G.

Figure 3.13b: Snort rules verification on the Windows Server 2016

The verification part of the Snort installation requires the copying of some of the established IDS rules into the new Snort installation that was just installed. Those files include navigating to the ***etc*** folder on our Windows 2016 Server Desktop, C:\Users\Administrator\Desktop\Snort\snortrules\etc of the Snort rules, copy ***snort.conf***, and paste it in the installation directory C:\Snort\etc. ***Snort.conf*** is already present in C:\Snort\etc; replace it with the snortrules' snort.conf file.

Same follows for the ***so_rules***, ***preproc_rules*** and the ***rules*** folders from the C:\Users\Administrator\Desktop\Snort\snortrules\etc folder into our Snort installation folder – C:\Snort.

Next, navigate to C:\Snort, and ***Shift+right-click*** on ***bin*** folder; click ***Open command window here*** from the context menu to open it in a command prompt as shown in Figure 3.14a, 3.14b and 3.14c.

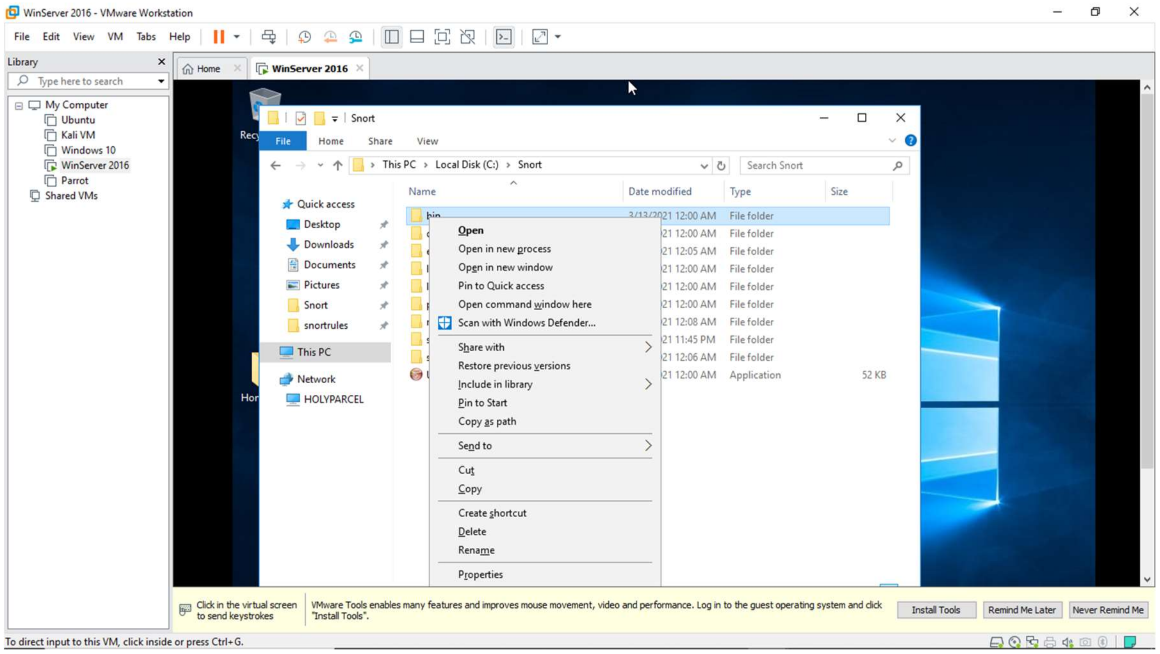
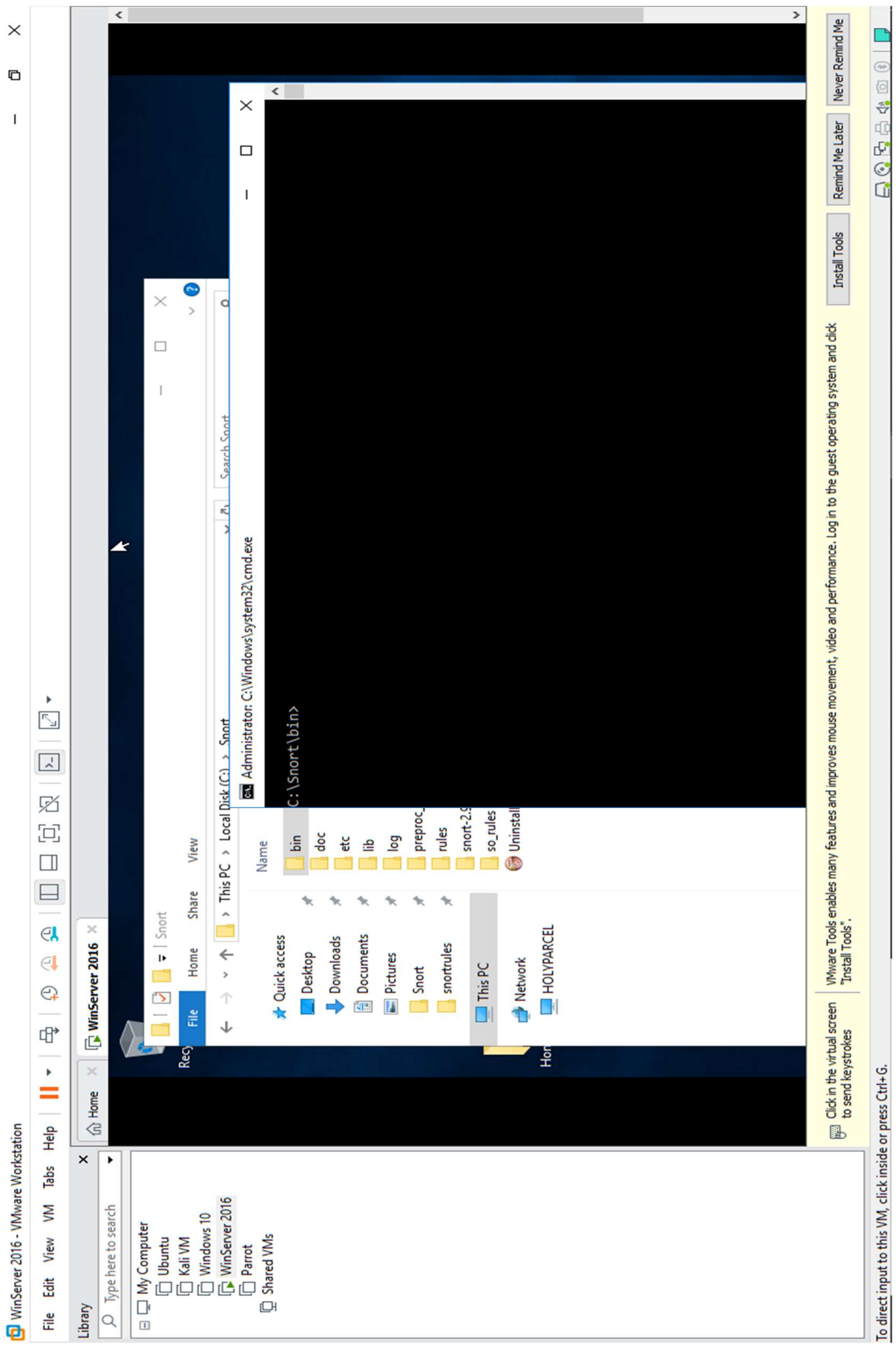
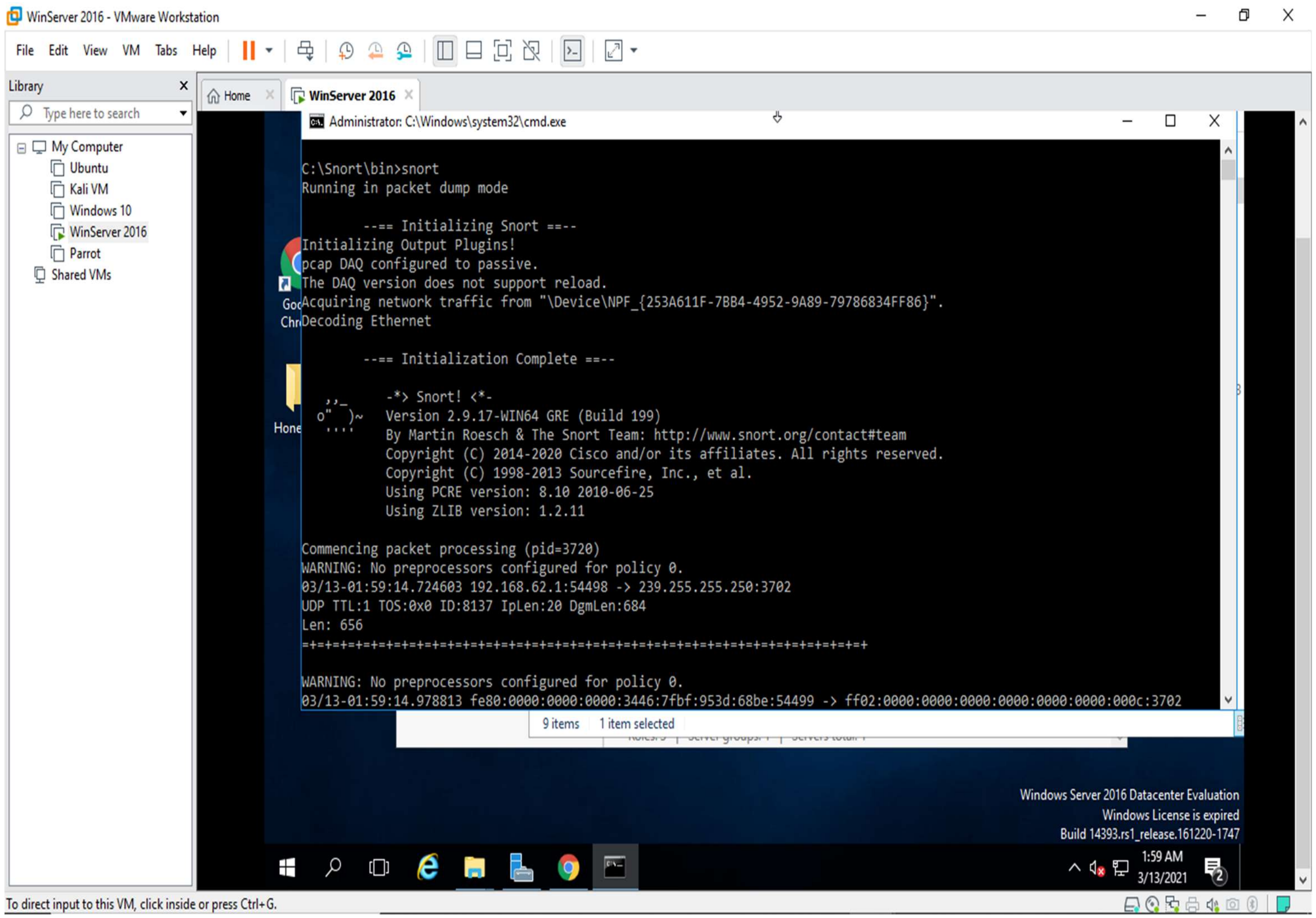


Figure 3.14a: Snort command line opening on the Windows Server 2016


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Snort
File Home Share View
This PC > Local Disk (C:) > Snort
Search Snort
Quick access
Desktop
Downloads
Documents
Pictures
Snort
snortrules
This PC
Network
HOLYPARCEL
Name
bin
doc
etc
lib
log
preproc_
rules
snort-2.9
so_rules
Uninstall
Administrator: C:\Windows\system32\cmd.exe
Click in the virtual screen to send keystrokes
VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".
Install Tools
Remind Me Later
Never Remind Me
To direct input to this VM, click inside or press Ctrl+G.

```
C:\Snort\bin>
```

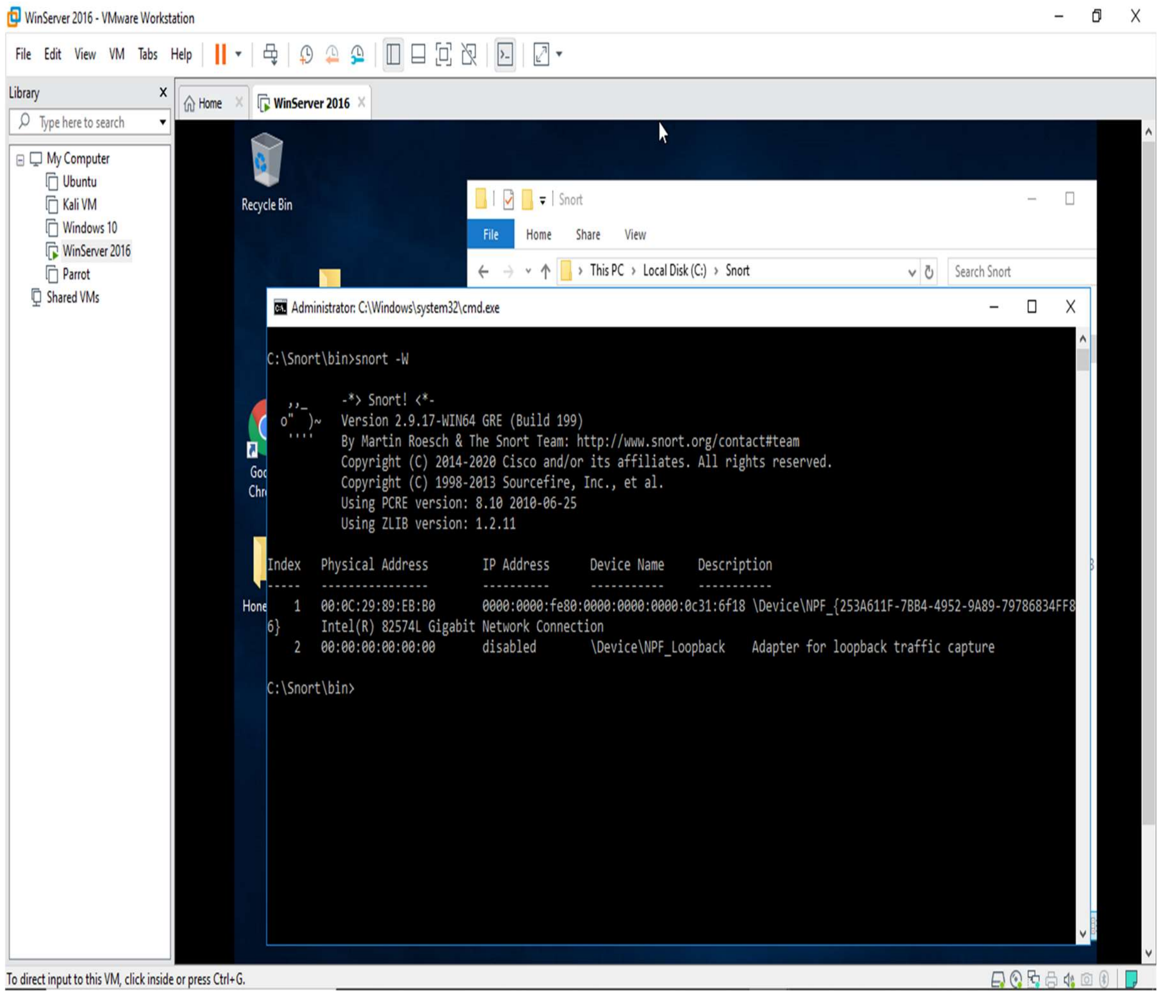
Figure 3.14b: Snort command line opening on the Windows Server 2016


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Administrator: C:\Windows\system32\cmd.exe
9 items
1 item selected
Windows Server 2016 Datacenter Evaluation
Windows License is expired
Build 14393.rs1_release.161220-1747
1:59 AM
3/13/2021
To direct input to this VM, click inside or press Ctrl+G.

```
C:\Snort\bin>snort
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{253A611F-7BB4-4952-9A89-79786834FF86}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.17-WIN64 GRE (Build 199)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=3720)
WARNING: No preprocessors configured for policy 0.
03/13-01:59:14.724603 192.168.62.1:54498 -> 239.255.255.250:3702
UDP TTL:1 TOS:0x0 ID:8137 IpLen:20 DgmLen:684
Len: 656
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
03/13-01:59:14.978813 fe80:0000:0000:0000:3446:7fbf:953d:68be:54499 -> ff02:0000:0000:0000:0000:0000:0000:000c:3702
```

Figure 3.14c: Snort installation and verification on the Windows Server 2016

It’s important to check the server’s parameters, hence we enter the command ***snort -W*** and press Enter. This lists our machine’s Physical Address, IP Address, and Ethernet Drivers, which are all disabled by default. It’s important to note that our Ethernet Driver index number and write it down (it is 1 for this experiment as depicted in Figure 3.15a).

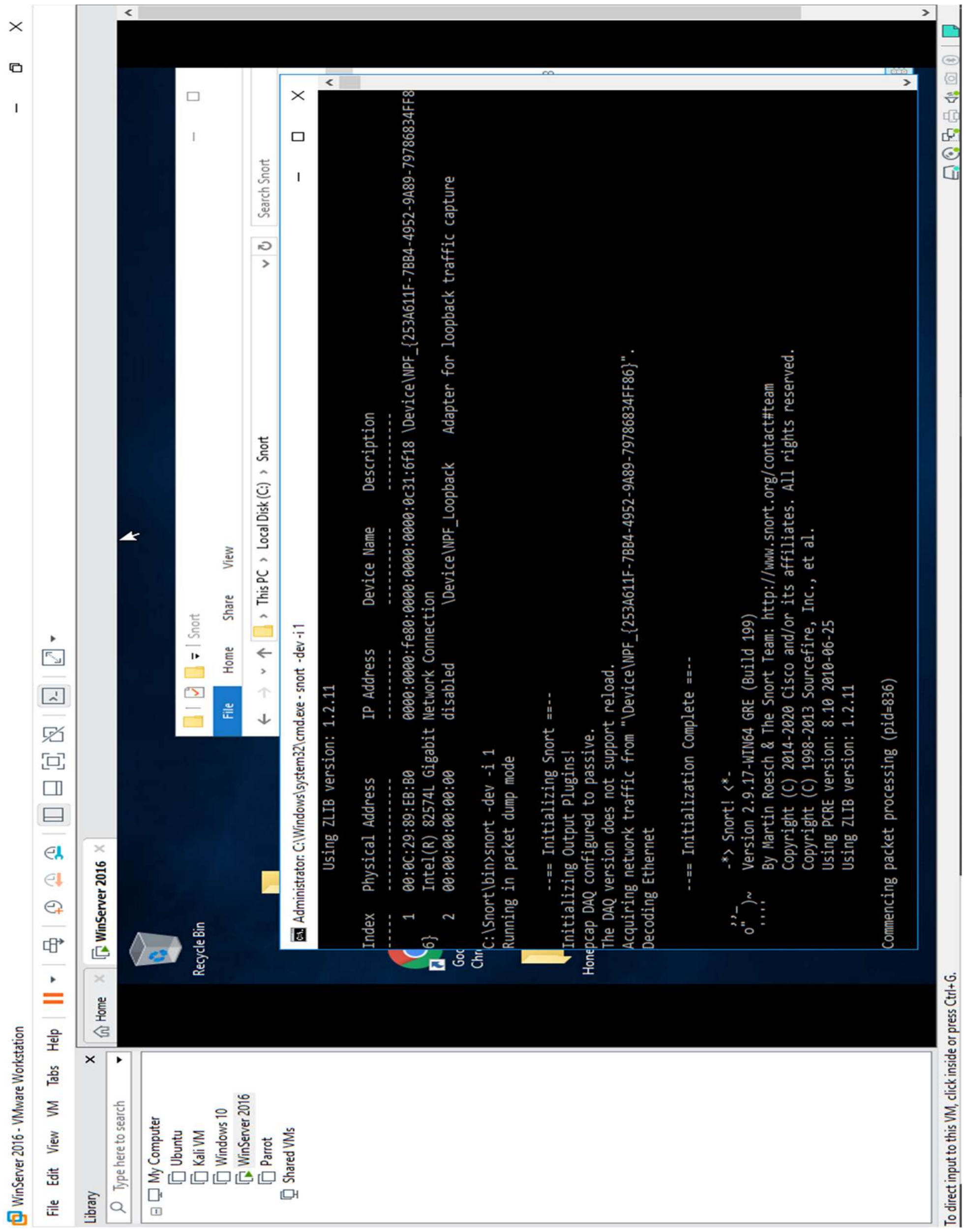

WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Recycle Bin
Snort
File Home Share View
This PC › Local Disk (C:) › Snort
Search Snort
Administrator: C:\Windows\system32\cmd.exe
To direct input to this VM, click inside or press Ctrl+G.

```
C:\Snort\bin>snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.17-WIN64 GRE (Build 199)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Index  Physical Address      IP Address      Device Name     Description
-----  ----------------      ----------      -----------     -----------
    1  00:0C:29:89:EB:B0     0000:0000:fe80:0000:0000:0000:0c31:6f18 \Device\NPF_{253A611F-7BB4-4952-9A89-79786834FF8
6}     Intel(R) 82574L Gigabit Network Connection
    2  00:00:00:00:00:00     disabled        \Device\NPF_Loopback    Adapter for loopback traffic capture

C:\Snort\bin>
```

Figure 3.15a: Our Windows Server 2016 details as seen by Snort

In Figure 3.15b, the Ethernet Driver needs to be activated, hence, in the command prompt, ***snort –dev –i 1*** is typed and press Enter. A rapid scroll text is seen in the command prompt, which means that the Ethernet Driver is enabled and working properly.

Upon this activation, Snort is left running in the command prompt window as it is and launch another command prompt window.

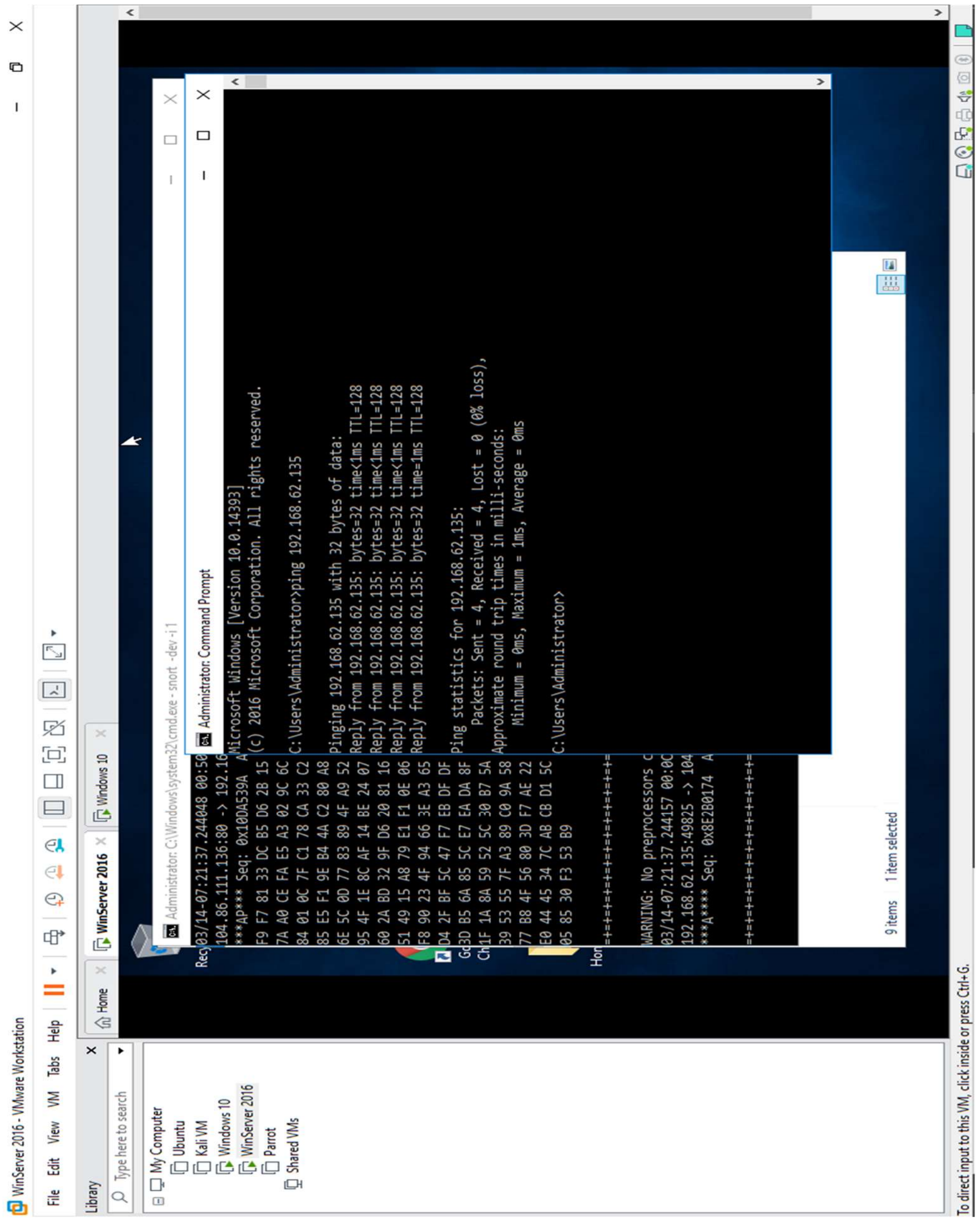

WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Recycle Bin
Snort
File Home Share View
This PC › Local Disk (C:) › Snort
Search Snort
Administrator: C:\Windows\system32\cmd.exe - snort -dev -i 1
To direct input to this VM, click inside or press Ctrl+G.

```
        Using ZLIB version: 1.2.11

Index  Physical Address      IP Address      Device Name     Description
-----  ----------------      ----------      -----------     -----------
    1  00:0C:29:89:EB:B0     0000:0000:fe80:0000:0000:0000:0c31:6f18 \Device\NPF_{253A611F-7BB4-4952-9A89-79786834FF8
6}     Intel(R) 82574L Gigabit Network Connection
    2  00:00:00:00:00:00     disabled        \Device\NPF_Loopback    Adapter for loopback traffic capture

C:\Snort\bin>snort -dev -i 1
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{253A611F-7BB4-4952-9A89-79786834FF86}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.17-WIN64 GRE (Build 199)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=836)
```

Figure 3.15b: Our Windows Server 2016 Ethernet driver now activated

Next, open another command prompt window and ping the IP address of the Windows 10 system from there. The IP address of the Windows 10 system is 192.168.62.135 for this simulation experiment as depicted in Figure 3.16.

This means type ***ping 192.168.62.135*** and press ***Enter*** in the new command prompt window. Also, it is possible to verify for any other machine that is present in the network.



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ping 192.168.62.135

Pinging 192.168.62.135 with 32 bytes of data:
Reply from 192.168.62.135: bytes=32 time<1ms TTL=128
Reply from 192.168.62.135: bytes=32 time<1ms TTL=128
Reply from 192.168.62.135: bytes=32 time<1ms TTL=128
Reply from 192.168.62.135: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.62.135:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\Administrator>
```

```
03/14-07:21:37.244048 00:50
104.86.111.136:80 -> 192.16
***AP*** Seq: 0x10DA539A  A
F9 F7 81 33 DC B5 D6 2B 15
7A A0 CE FA E5 A3 02 9C 6C
84 01 0C 7F C1 78 CA 33 C2
85 E5 F1 9E B4 4A C2 80 A8
6E 5C 0D 77 83 89 4F A9 52
95 4F 1E 8C AF 14 BE 24 07
60 2A BD 32 9F D6 20 81 16
51 49 15 A8 79 E1 F1 0E 06
F8 90 23 4F 94 66 3E A3 65
D4 2F BF 5C 47 F7 EB DF DF
3D B5 6A 85 5C E7 EA DA 8F
1F 1A 8A 59 52 5C 30 B7 5A
39 53 55 7F A3 89 C0 9A 58
77 B8 4F 56 80 3D F7 AE 22
E0 44 45 34 7C AB CB D1 5C
05 85 30 F3 53 B9

=+=+=+=+=+=+=+=+=+=+=+=+=+=

WARNING: No preprocessors c
03/14-07:21:37.244157 00:0C
192.168.62.135:49825 -> 104
***A**** Seq: 0x8E2B0174  A

=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

Figure 3.16: Image showing the successful setting up of the Snort alerts

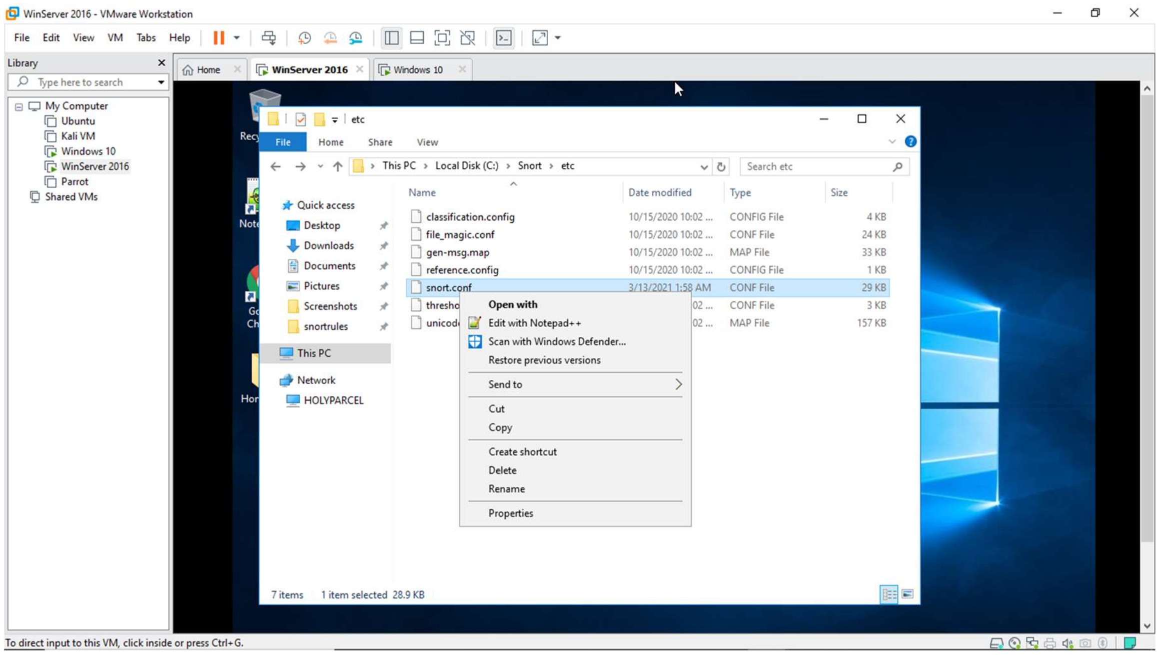
The ping command activates a Snort alert in the Snort command prompt with rapid scrolling text. Hitting the Enter button in the new command prompt switch to Snort command prompt window immediately (as fast as you can) to see the snort alert. Close both command prompt windows.

The verification of Snort installation and triggering alert is complete, and Snort is now working correctly in verbose mode.

B. Configure and validate snort.conf file

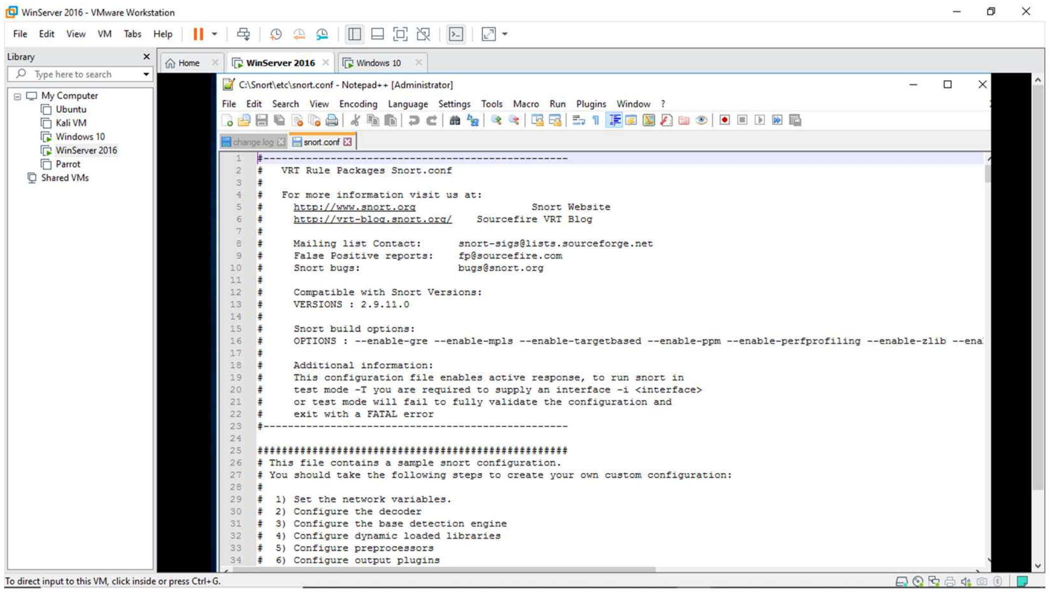
The next thing to do is to edit the Snort configuration file. To do this, navigate to C:\Snort\etc, and ***right-click on snort.conf*** file and click ***Edit with Notepad++*** from the context menu to start editing the snort.conf file.

The snort.conf file opens in Notepad++, as shown in Figure 3.17. Scroll down to the Step #1: Set the network variables section (Line 41) of snort.conf file. In the **HOME_NET** line (Line 45), replace ***any*** with the IP addresses of the machine (target machine) on which Snort is running. Here, the target machine is Windows 2016 server, and the IP address is 192.168.62.137. Leave the **EXTERNAL_NET** any line as it is.


WinServer 2016 - VMware Workstation
Library
Open with
Edit with Notepad++
Scan with Windows Defender...
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties

```
This PC > Local Disk (C:) > Snort > etc

Name                  Date modified        Type         Size
classification.config 10/15/2020 10:02 ... CONFIG File  4 KB
file_magic.conf       10/15/2020 10:02 ... CONF File    24 KB
gen-msg.map           10/15/2020 10:02 ... MAP File     33 KB
reference.config      10/15/2020 10:02 ... CONFIG File  1 KB
snort.conf            3/13/2021 1:58 AM    CONF File    29 KB
thresho                            02 ... CONF File    3 KB
unicod                             02 ... MAP File     157 KB

7 items    1 item selected 28.9 KB
```

Figure 3.17a: Image showing the opening of the configuration file for editing


WinServer 2016 - VMware Workstation
C:\Snort\etc\snort.conf - Notepad++ [Administrator]

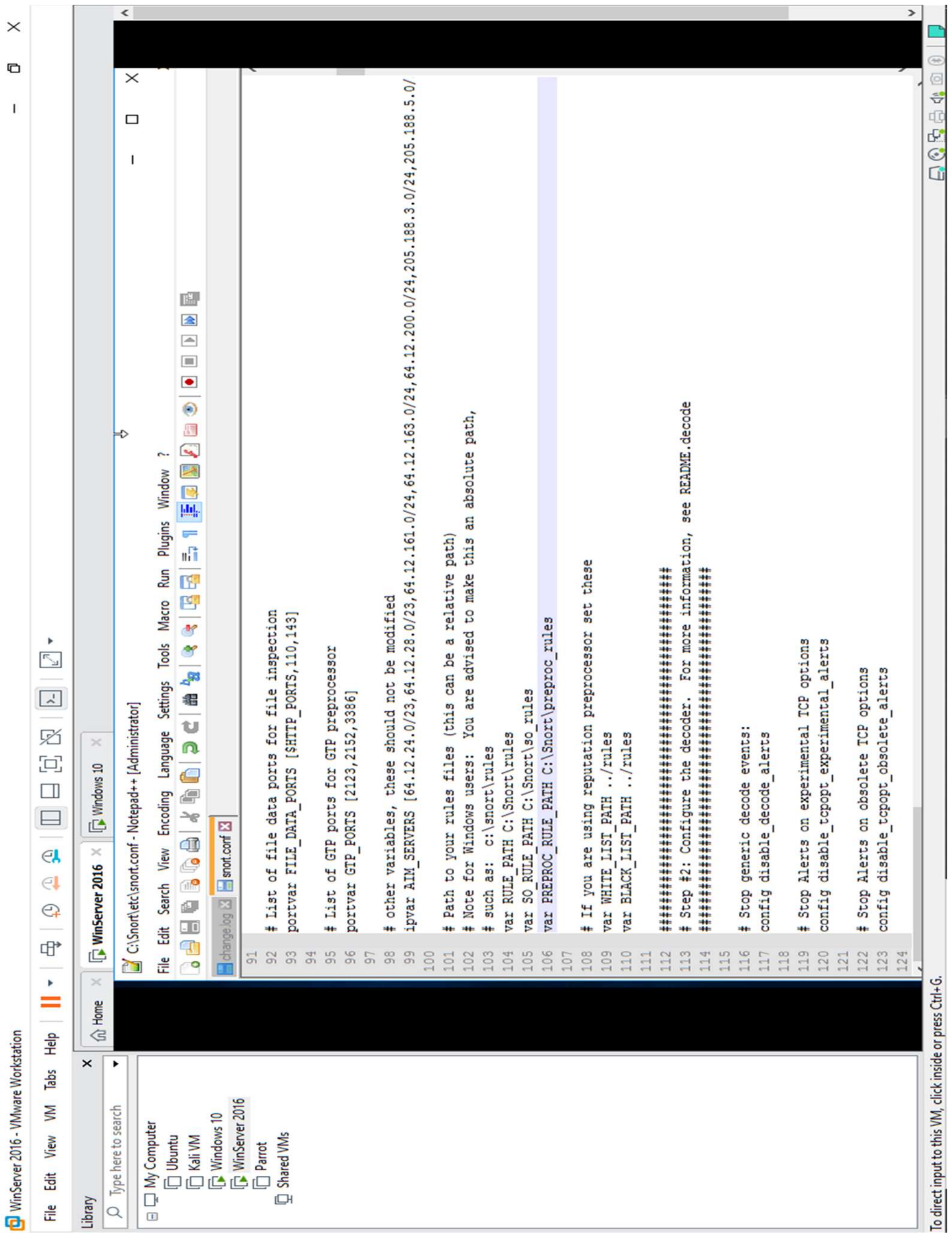
```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                   Snort Website
#     http://vrt-blog.snort.org/    Sourcefire VRT Blog
#
#     Mailing list Contact:      snort-sigs@lists.sourceforge.net
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.11.0
#
#     Snort build options:
#     OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling --enable-zlib --ena
#
#     Additional information:
#     This configuration file enables active response, to run snort in
#     test mode -T you are required to supply an interface -i <interface>
#     or test mode will fail to fully validate the configuration and
#     exit with a FATAL error
#--------------------------------------------------

###################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom configuration:
#
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
```

Figure 3.17b: Image showing the opened configuration file for editing

Since there is no DNS Server present, this line is left as it is. The same applies to SMTP_SERVERS, HTTP_SERVERS, SQL_SERVERS, TELNET_SERVERS, and SSH_SERVERS. DO NOT make any changes in that line.

In this experiment a Live Internet Connectivity is used but the DNS Server line is still left as it is.

Furthermore, scroll down to RULE_PATH (Line 104). In Line 104, replace ../rules with C:\Snort\rules, in Line 105 ../so_rules is replaced with C:\Snort\so_rules, and in Line 106, replace ../preproc_rules with C:\Snort\preproc_rules as shown in Figure 3.18:

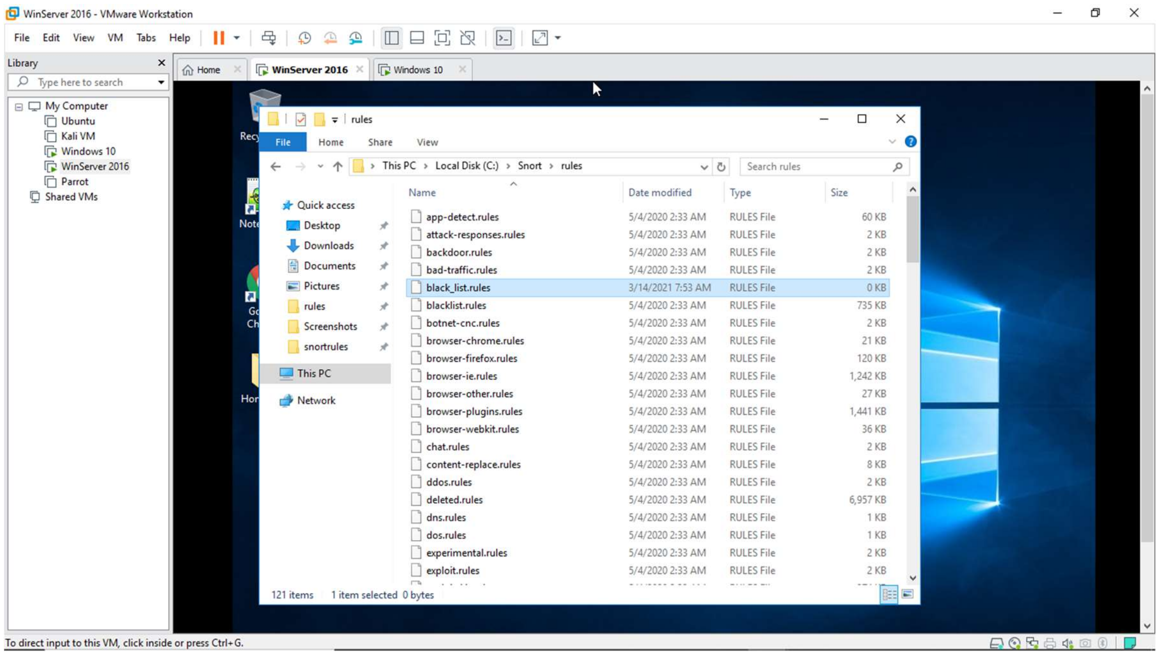

WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
C:\Snort\etc\snort.conf - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
snort.conf
To direct input to this VM, click inside or press Ctrl+G.

```
91
92   # List of file data ports for file inspection
93   portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]
94
95   # List of GTP ports for GTP preprocessor
96   portvar GTP_PORTS [2123,2152,3386]
97
98   # other variables, these should not be modified
99   ipvar AIM_SERVERS [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/
100
101  # Path to your rules files (this can be a relative path)
102  # Note for Windows users:  You are advised to make this an absolute path,
103  # such as:  c:\snort\rules
104  var RULE_PATH C:\Snort\rules
105  var SO_RULE_PATH C:\Snort\so_rules
106  var PREPROC_RULE_PATH C:\Snort\preproc_rules
107
108  # If you are using reputation preprocessor set these
109  var WHITE_LIST_PATH ../rules
110  var BLACK_LIST_PATH ../rules
111
112  ###################################################
113  # Step #2: Configure the decoder.  For more information, see README.decode
114  ###################################################
115
116  # Stop generic decode events:
117  config disable_decode_alerts
118
119  # Stop Alerts on experimental TCP options
120  config disable_tcpopt_experimental_alerts
121
122  # Stop Alerts on obsolete TCP options
123  config disable_tcpopt_obsolete_alerts
124
```

Figure 3.18: Image showing the edited Snort configuration file

Next, navigate to C:\Snort\rules, and create two text files; name them ***white_list*** and ***black_list*** and edit the file extensions from ***.txt*** to ***.rules***. While going through the change of the extension, if any pop-up appears, click ***Yes***.

Then go back to our ***snort.conf*** file in Notepad ++ and scroll down to Step #4: Configure dynamic loaded libraries section (Line 238) where configure dynamic loaded libraries in this section as shown in Figure 3.19.

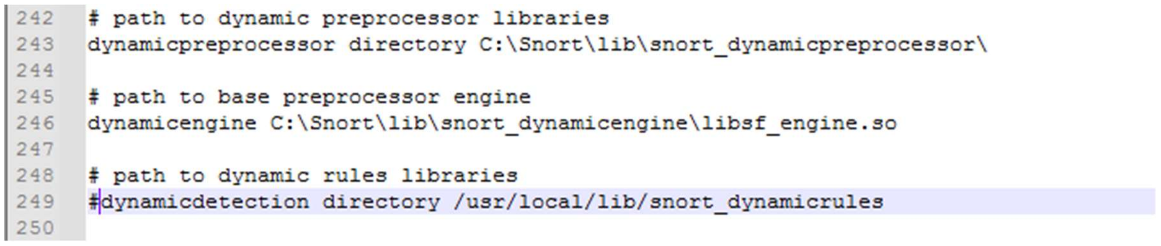


Name	Date modified	Type	Size
app-detect.rules	5/4/2020 2:33 AM	RULES File	60 KB
attack-responses.rules	5/4/2020 2:33 AM	RULES File	2 KB
backdoor.rules	5/4/2020 2:33 AM	RULES File	2 KB
bad-traffic.rules	5/4/2020 2:33 AM	RULES File	2 KB
black_list.rules	3/14/2021 7:53 AM	RULES File	0 KB
blacklist.rules	5/4/2020 2:33 AM	RULES File	735 KB
botnet-cnc.rules	5/4/2020 2:33 AM	RULES File	2 KB
browser-chrome.rules	5/4/2020 2:33 AM	RULES File	21 KB
browser-firefox.rules	5/4/2020 2:33 AM	RULES File	120 KB
browser-ie.rules	5/4/2020 2:33 AM	RULES File	1,242 KB
browser-other.rules	5/4/2020 2:33 AM	RULES File	27 KB
browser-plugins.rules	5/4/2020 2:33 AM	RULES File	1,441 KB
browser-webkit.rules	5/4/2020 2:33 AM	RULES File	36 KB
chat.rules	5/4/2020 2:33 AM	RULES File	2 KB
content-replace.rules	5/4/2020 2:33 AM	RULES File	8 KB
ddos.rules	5/4/2020 2:33 AM	RULES File	2 KB
deleted.rules	5/4/2020 2:33 AM	RULES File	6,957 KB
dns.rules	5/4/2020 2:33 AM	RULES File	1 KB
dos.rules	5/4/2020 2:33 AM	RULES File	1 KB
experimental.rules	5/4/2020 2:33 AM	RULES File	2 KB
exploit.rules	5/4/2020 2:33 AM	RULES File	2 KB

Figure 3.19: Image showing the newly created ***black_list*** and ***white_list.rules*** file

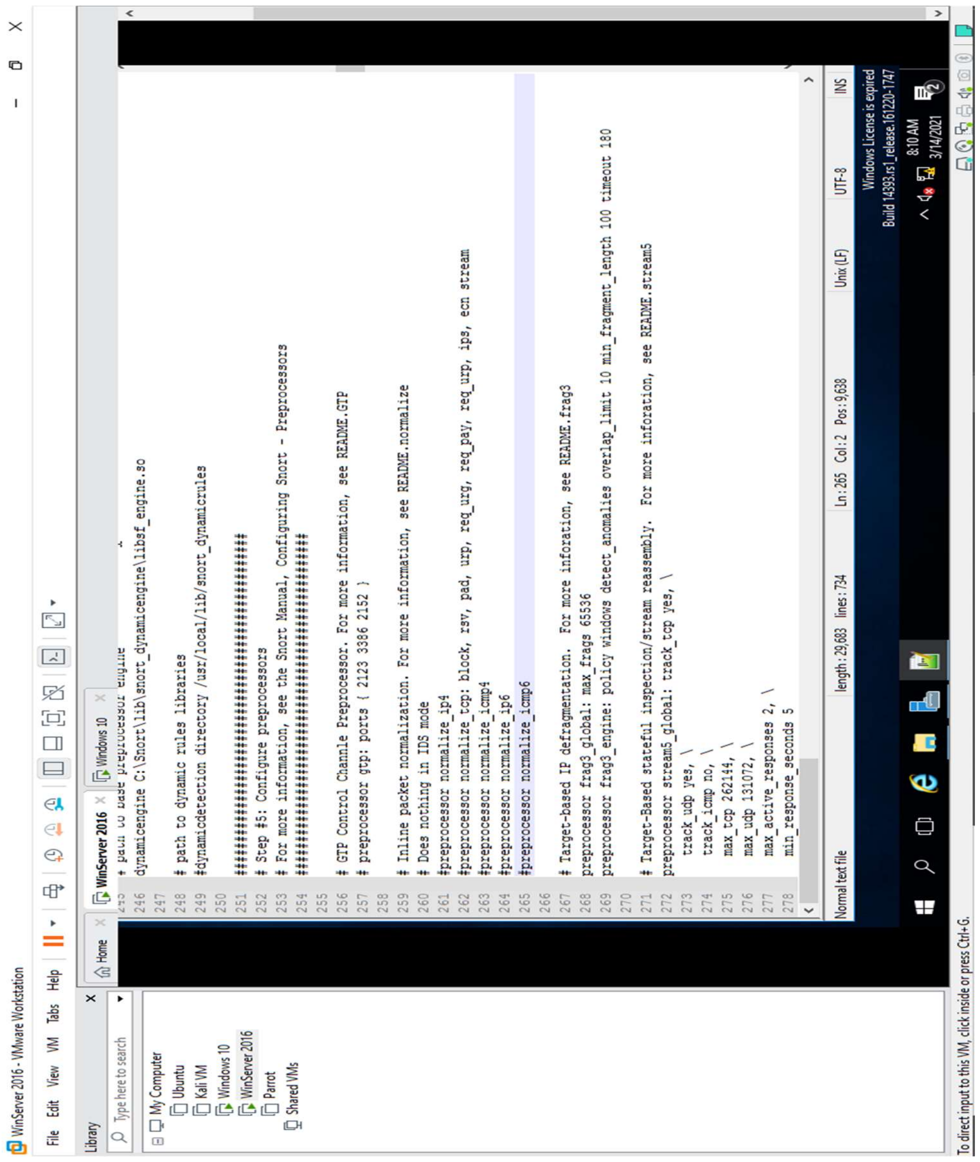
At the path for dynamic preprocessor libraries (Line 243), replace /usr/local/lib/snort_dynamicpreprocessor/with your dynamic preprocessor libraries folder location. In this lab, dynamic preprocessor libraries are located at C:\Snort\lib\snort_dynamicpreprocessor. At the path for base preprocessor engine (Line 246); /usr/local/lib/snort_dynamicengine/libsf_engine.so is replaced with the base preprocessor engine C:\Snort\lib\snort_dynamicengine\sf_engine.dll.

Having already configured the dynamic preprocessor libraries, comment (using #) out line 249 as shown in Figure 3.20.



```
242  # path to dynamic preprocessor libraries
243  dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor\
244
245  # path to base preprocessor engine
246  dynamicengine C:\Snort\lib\snort_dynamicengine\libsf_engine.so
247
248  # path to dynamic rules libraries
249  #dynamicdetection directory /usr/local/lib/snort_dynamicrules
250
```

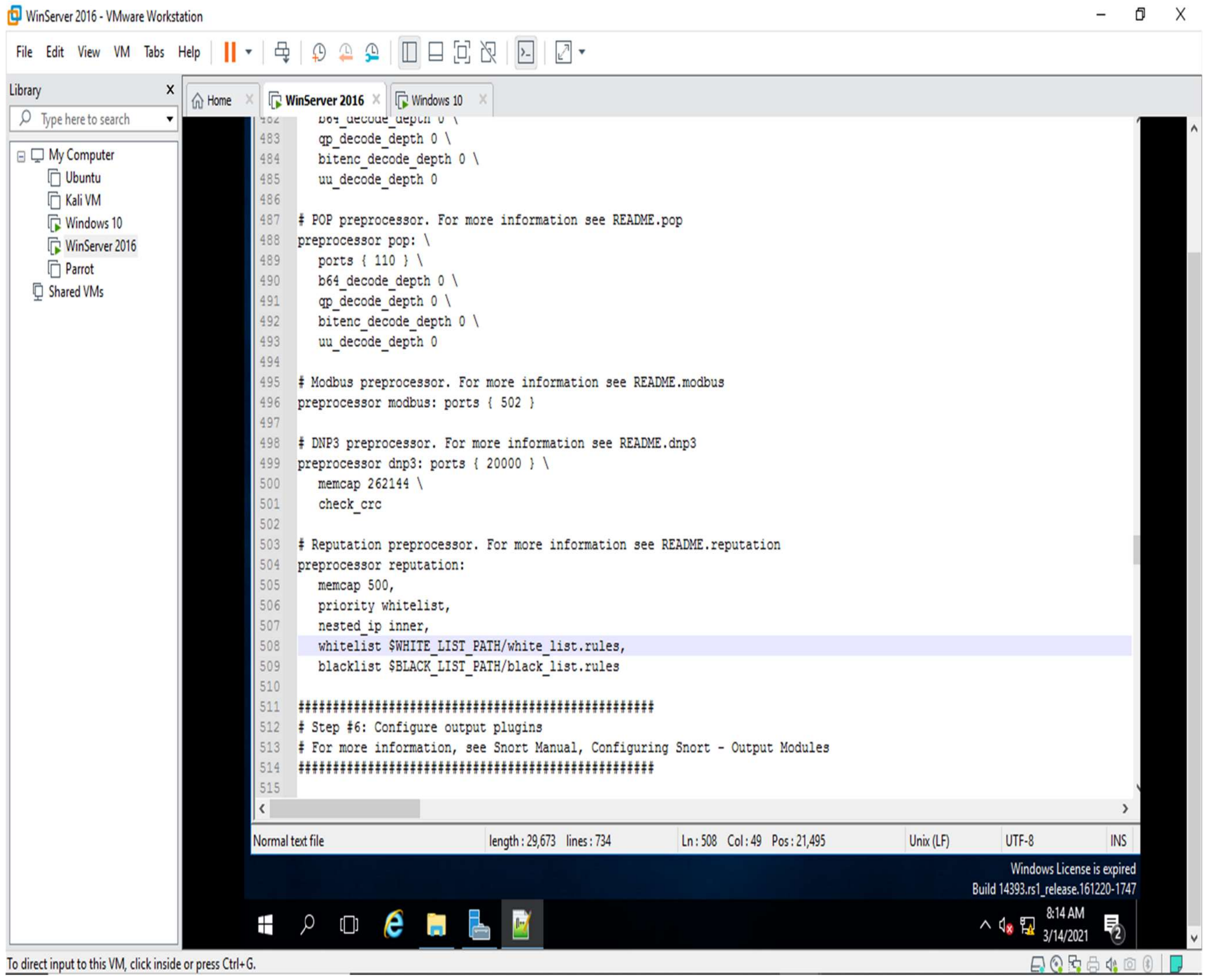
Figure 3.20: Commenting out the dynamic rule libraries in our Snort config file


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
Normal text file
length : 29,683 lines : 734
Ln : 265 Col : 2 Pos : 9,638
Unix (LF)
UTF-8
INS
Windows License is expired
Build 14393.rs1_release.161220-1747
8:10 AM
3/14/2021
To direct input to this VM, click inside or press Ctrl+G.

```
246  dynamicengine C:\Snort\lib\snort_dynamicengine\libsf_engine.so
247
248  # path to dynamic rules libraries
249  #dynamicdetection directory /usr/local/lib/snort_dynamicrules
250
251  ###################################################
252  # Step #5: Configure preprocessors
253  # For more information, see the Snort Manual, Configuring Snort - Preprocessors
254  ###################################################
255
256  # GTP Control Channle Preprocessor. For more information, see README.GTP
257  # preprocessor gtp: ports { 2123 3386 2152 }
258
259  # Inline packet normalization. For more information, see README.normalize
260  # Does nothing in IDS mode
261  #preprocessor normalize_ip4
262  #preprocessor normalize_tcp: block, rsv, pad, urp, req_urg, req_pay, req_urp, ips, ecn stream
263  #preprocessor normalize_icmp4
264  #preprocessor normalize_ip6
265  #preprocessor normalize_icmp6
266
267  # Target-based IP defragmentation.  For more inforation, see README.frag3
268  preprocessor frag3_global: max_frags 65536
269  preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180
270
271  # Target-Based stateful inspection/stream reassembly.  For more inforation, see README.stream5
272  preprocessor stream5_global: track_tcp yes, \
273     track_udp yes, \
274     track_icmp no, \
275     max_tcp 262144, \
276     max_udp 131072, \
277     max_active_responses 2, \
278     min_response_seconds 5
```

Figure 3.21: Commenting out the preprocessor rules in our Snort config file

Going forward, move to the segment Step #5: Configure Preprocessors section (Line 252), the listed preprocessor. Do nothing in IDS mode, but generate errors at runtime. Then comment all the preprocessors listed in this section. This is done by adding # before each preprocessor rule (261-265) as shown in Figure 3.21.



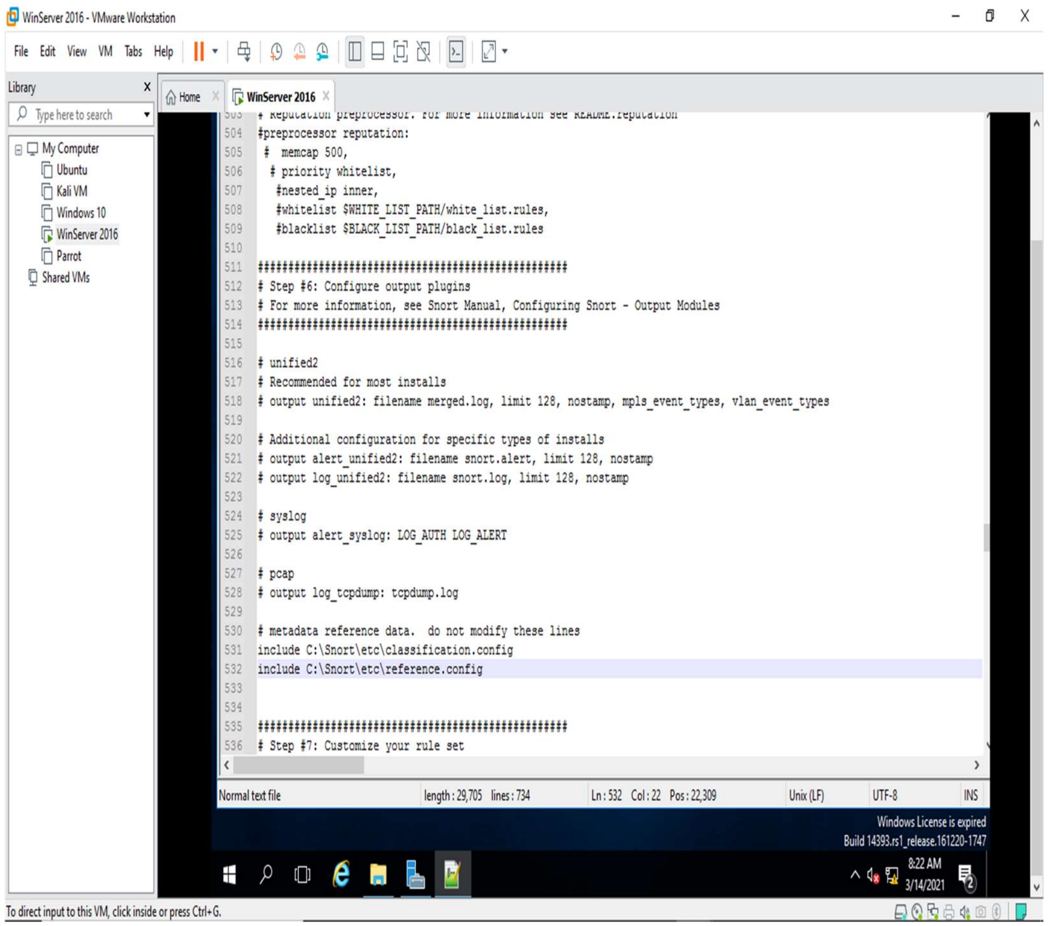
```
482     b64_decode_depth 0 \
483     qp_decode_depth 0 \
484     bitenc_decode_depth 0 \
485     uu_decode_depth 0
486
487  # POP preprocessor. For more information see README.pop
488  preprocessor pop: \
489     ports { 110 } \
490     b64_decode_depth 0 \
491     qp_decode_depth 0 \
492     bitenc_decode_depth 0 \
493     uu_decode_depth 0
494
495  # Modbus preprocessor. For more information see README.modbus
496  preprocessor modbus: ports { 502 }
497
498  # DNP3 preprocessor. For more information see README.dnp3
499  preprocessor dnp3: ports { 20000 } \
500     memcap 262144 \
501     check_crc
502
503  # Reputation preprocessor. For more information see README.reputation
504  preprocessor reputation:
505     memcap 500,
506     priority whitelist,
507     nested_ip inner,
508     whitelist $WHITE_LIST_PATH/white_list.rules,
509     blacklist $BLACK_LIST_PATH/black_list.rules
510
511  ###################################################
512  # Step #6: Configure output plugins
513  # For more information, see Snort Manual, Configuring Snort - Output Modules
514  ###################################################
515
```

WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
Normal text file
length : 29,673 lines : 734
Ln : 508 Col : 49 Pos : 21,495
Unix (LF)
UTF-8
INS
Windows License is expired
Build 14393.rs1_release.161220-1747
8:14 AM
3/14/2021
To direct input to this VM, click inside or press Ctrl+G.

Figure 3.22: All backslashes (\) removed at the end of lines 504 – 509

Next, scroll to line 325 and delete ***lzma*** keyword. Note that, only the ***lzma*** keyword should be deleted from the line. Also, move to the lines 504-509 and delete all backslashes (\) at the end of each of the lines as shown in Figure 3.22 and then comment out all the lines:

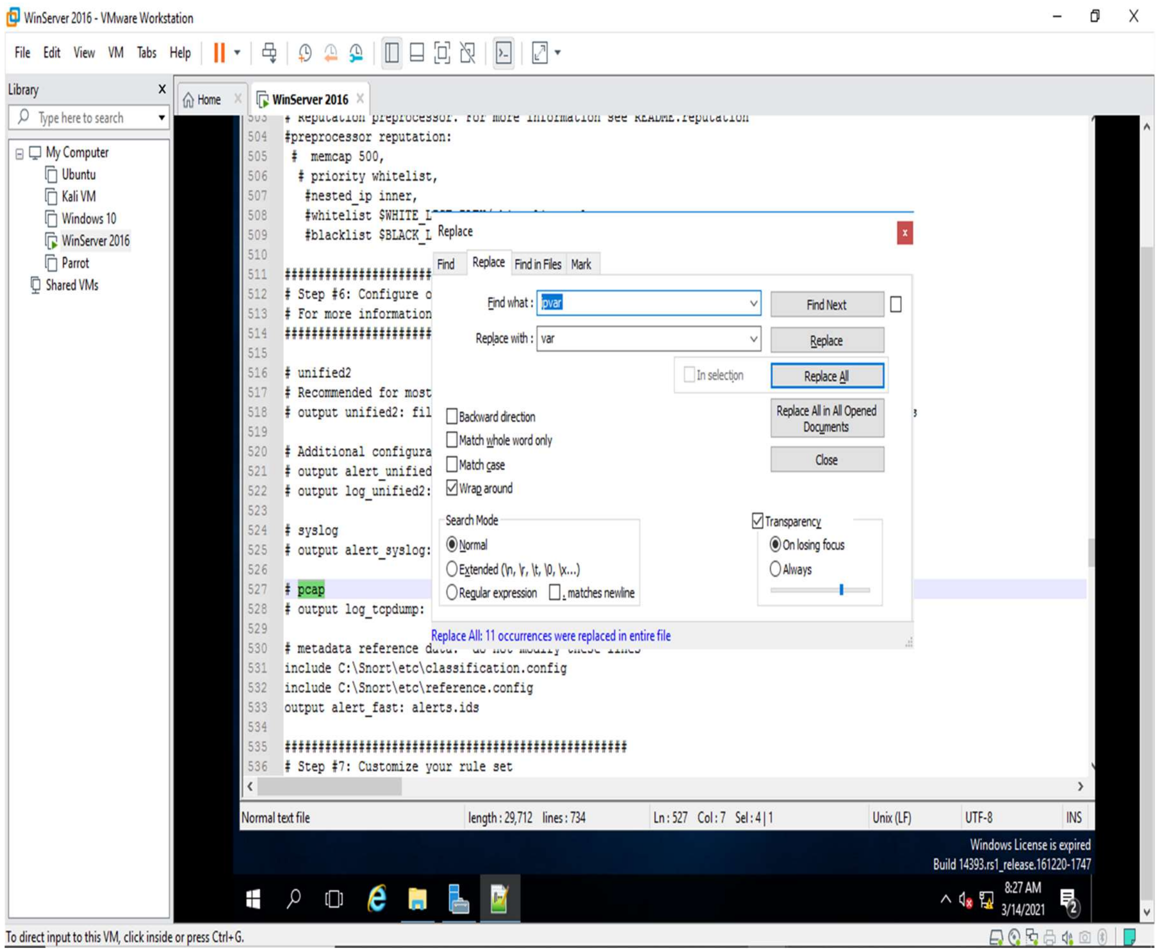
Moving ahead to Step #6: Configure output plugins (Line 512). This step particularly requires location of the ***classification.config*** and ***reference.config*** files be provided as shown in Figure 3.23. These two files are located in C:\Snort\etc. These locations of files in configure output plugins (in Lines 531 and 532) is provided, that is, C:\Snort\etc\classification.config and C:\Snort\etc\reference.config.


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Normal text file
length : 29,705 lines : 734
Ln : 532 Col : 22 Pos : 22,309
Unix (LF)
UTF-8
INS
Windows License is expired
Build 14393.rs1_release.161220-1747
8:22 AM
3/14/2021
To direct input to this VM, click inside or press Ctrl+G.

```
503 # Reputation preprocessor. For more information see README.reputation
504 #preprocessor reputation:
505  #  memcap 500,
506   # priority whitelist,
507    #nested_ip inner,
508    #whitelist $WHITE_LIST_PATH/white_list.rules,
509    #blacklist $BLACK_LIST_PATH/black_list.rules
510
511 ###################################################
512 # Step #6: Configure output plugins
513 # For more information, see Snort Manual, Configuring Snort - Output Modules
514 ###################################################
515
516 # unified2
517 # Recommended for most installs
518 # output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types
519
520 # Additional configuration for specific types of installs
521 # output alert_unified2: filename snort.alert, limit 128, nostamp
522 # output log_unified2: filename snort.log, limit 128, nostamp
523
524 # syslog
525 # output alert_syslog: LOG_AUTH LOG_ALERT
526
527 # pcap
528 # output log_tcpdump: tcpdump.log
529
530 # metadata reference data.  do not modify these lines
531 include C:\Snort\etc\classification.config
532 include C:\Snort\etc\reference.config
533
534
535 ###################################################
536 # Step #7: Customize your rule set
```

Figure 3.23: Classification and Reference line configuration in Snort rules

Still under step #6, the added output alert_fast: ***alerts.ids*** in line #533 for Snort to dump all logs in the alerts.ids file. In the snort.conf file, find and replace the ***ipvar*** string with ***var***. In doing this, press Ctrl+h on keyboard. The Replace window appears, enter ***ipvar*** in the ***Find what*** : text field, enter ***var*** in the ***Replace with*** : text field and click ***Replace All***. By default, the string is ***ipvar***, which is not a command recognized by Snort, so it's important we replace it with the ***var*** string, and then close the window.


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Find
Replace
Find in Files
Mark
Find Next
Replace
Replace All
Replace All in All Opened Documents
Close
Normal text file
Windows License is expired
Build 14393.rs1_release.161220-1747
8:27 AM
3/14/2021
To direct input to this VM, click inside or press Ctrl+G.

```
504 #preprocessor reputation:
505  #  memcap 500,
506   # priority whitelist,
507   #nested_ip inner,
508   #whitelist $WHITE_L
509   #blacklist $BLACK_L
510
511 ####################
512 # Step #6: Configure o
513 # For more information
514 ####################
515
516 # unified2
517 # Recommended for most
518 # output unified2: fil
519
520 # Additional configura
521 # output alert_unified
522 # output log_unified2:
523
524 # syslog
525 # output alert_syslog:
526
527 # pcap
528 # output log_tcpdump:
529
530 # metadata reference
531 include C:\Snort\etc\classification.config
532 include C:\Snort\etc\reference.config
533 output alert_fast: alerts.ids
534
535 ###################################################
536 # Step #7: Customize your rule set
```

Find what : ipvar

Replace with : var

☐ In selection

☐ Backward direction
☐ Match whole word only
☐ Match case
☑ Wrap around

Search Mode: [● Normal] ○ Extended (\n, \r, \t, \0, \x...) ○ Regular expression ☐ . matches newline

☑ Transparency: [● On losing focus] ○ Always

Replace All: 11 occurrences were replaced in entire file

length : 29,712 lines : 734 Ln : 527 Col : 7 Sel : 4 | 1 Unix (LF) UTF-8 INS

Figure 3.24: Replacing ***ipvar*** with ***var*** in the Snort configuration file

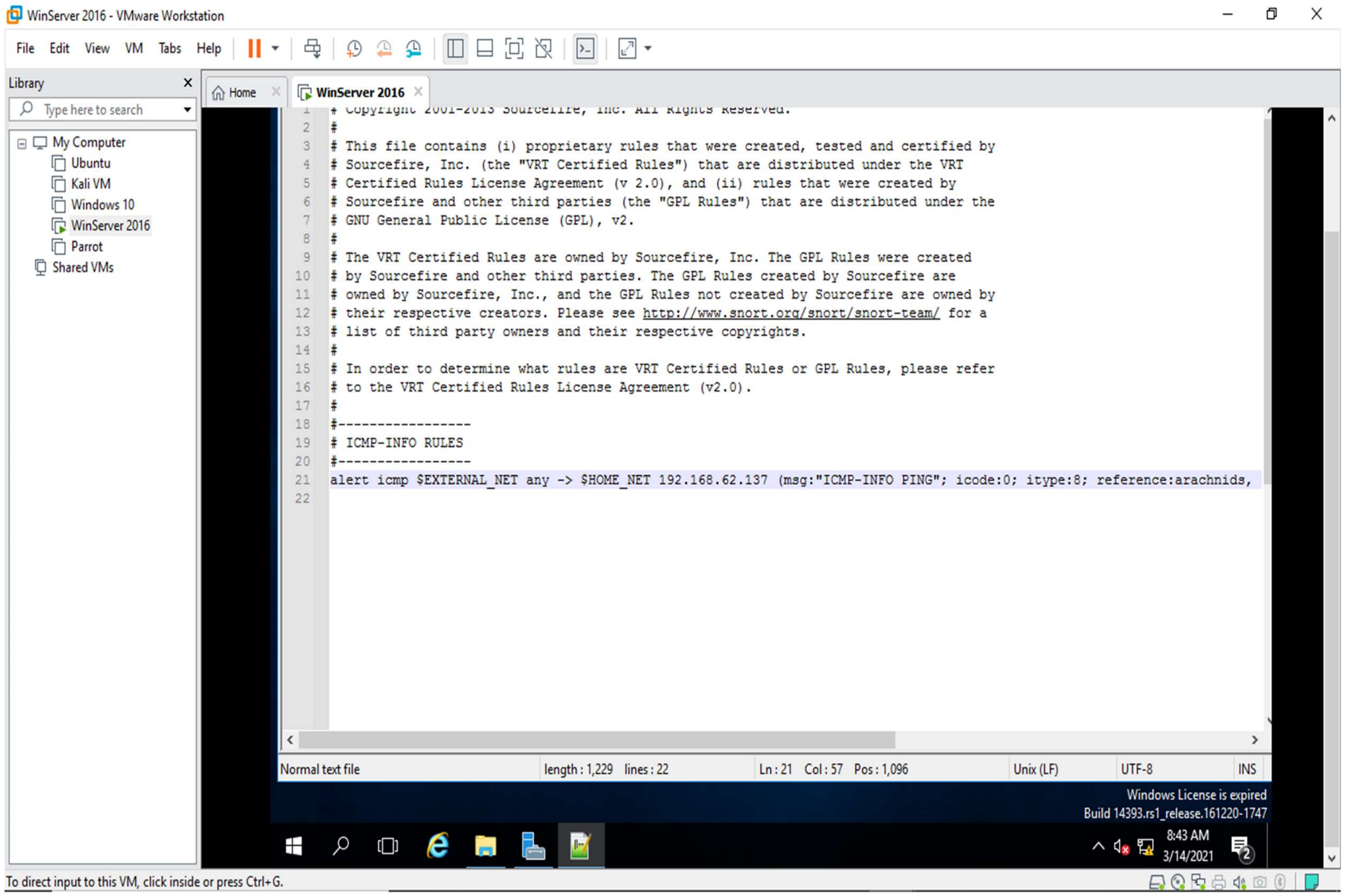
In Figure 3.24, the Snort IDS now has a vast support for multiple configurations based on VLAN Id or IP subnet all in a single instance of Snort. This particularly gives administrators the opportunity to specify multiple snort configuration files while also binding each configuration to one or more VLANs or subnets instead of running one Snort for each configuration required. This is a huge positive for the snort IDS.

C. Test working of Snort by carrying out attack test

Finally, before running Snort, detection rules in the Snort rules file needs to be enabled. In this simulation experiment, ICMP rule allowing Snort to detect any host discovery ping probes to the system running Snort has been enabled.

Navigate to C:\Snort\rules and open the ***icmp-info.rules*** file with Notepad ++. Type the command below in line 21, and save it as shown in Figure 3.25:

alert icmp $EXTERNAL_NET any -> $HOME_NET 192.168.62.137 (msg:"ICMP-INFO PING"; icode:0; itype:8; reference:arachnids,135; reference:cve,1999-0265; classtype:bad-unknown; sid:472; rev:7;)



```
# Copyright 2001-2013 Sourcefire, Inc. All Rights Reserved.
#
# This file contains (i) proprietary rules that were created, tested and certified by
# Sourcefire, Inc. (the "VRT Certified Rules") that are distributed under the VRT
# Certified Rules License Agreement (v 2.0), and (ii) rules that were created by
# Sourcefire and other third parties (the "GPL Rules") that are distributed under the
# GNU General Public License (GPL), v2.
#
# The VRT Certified Rules are owned by Sourcefire, Inc. The GPL Rules were created
# by Sourcefire and other third parties. The GPL Rules created by Sourcefire are
# owned by Sourcefire, Inc., and the GPL Rules not created by Sourcefire are owned by
# their respective creators. Please see http://www.snort.org/snort/snort-team/ for a
# list of third party owners and their respective copyrights.
#
# In order to determine what rules are VRT Certified Rules or GPL Rules, please refer
# to the VRT Certified Rules License Agreement (v2.0).
#
#-----------------
# ICMP-INFO RULES
#-----------------
alert icmp $EXTERNAL_NET any -> $HOME_NET 192.168.62.137 (msg:"ICMP-INFO PING"; icode:0; itype:8; reference:arachnids,
```

WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Normal text file
length : 1,229 lines : 22
Ln : 21 Col : 57 Pos : 1,096
Unix (LF)
UTF-8
INS
Windows License is expired
Build 14393.rs1_release.161220-1747
8:43 AM
3/14/2021
To direct input to this VM, click inside or press Ctrl+G.

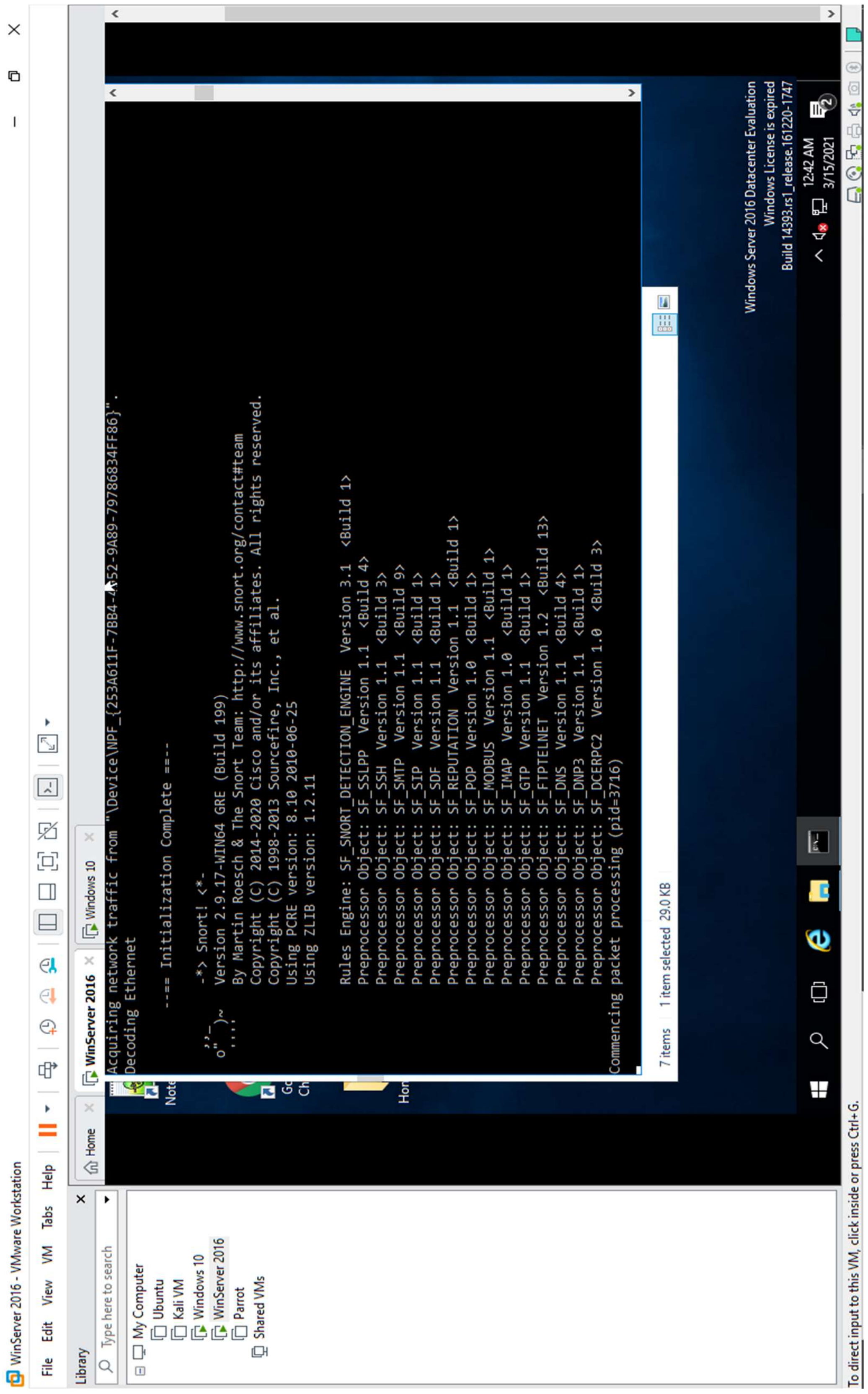
Figure 3.25: Setting the alert triggers in the icmp-info rule file.

Here the 192.168.62.137 is the IP address of the Windows 2016 Server machine where snort is running. Minimize Notepad++ window once the configuration is done.

Navigate to C:\Snort and ***Shift+right-click*** on ***bin*** folder, select ***Open command window here*** from the drop-down menu to open it in the command prompt. We then type: ***snort -iX -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K asci*** and press Enter to start ***Snort*** (our X for this experiment is our device index number; 1: therefore, X is 1).

With the above command, Snort commences operation in IDS mode. First, it initializes output preprocessors, load dynamic preprocessors libraries, plug-ins, rule chains of Snort, before it logs all signatures. If all command information is entered correctly, a comment is received implying ***Commencing packet processing <pid=xxxx>*** (Note that xxxx assumes any number; in this experiment, it is **3716**), as shown in Figure 3.26. With all initializing interface and logged signatures completed, Snort is started and awaiting an attack so it can trigger alert when attacks occur on the machine. Snort command prompt is left open and running as an attack is launched on our the machine, and observe Snort, if it detects it or not.

If an error message is received stating "***Could not create the registry key***," then it is probably because Snort as not started as an Administrator. The Warnings can be ignored while validation.


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
7 items 1 item selected 29.0 KB
Windows Server 2016 Datacenter Evaluation
Windows License is expired
Build 14393.rs1_release.161220-1747
12:42 AM
3/15/2021
To direct input to this VM, click inside or press Ctrl+G.

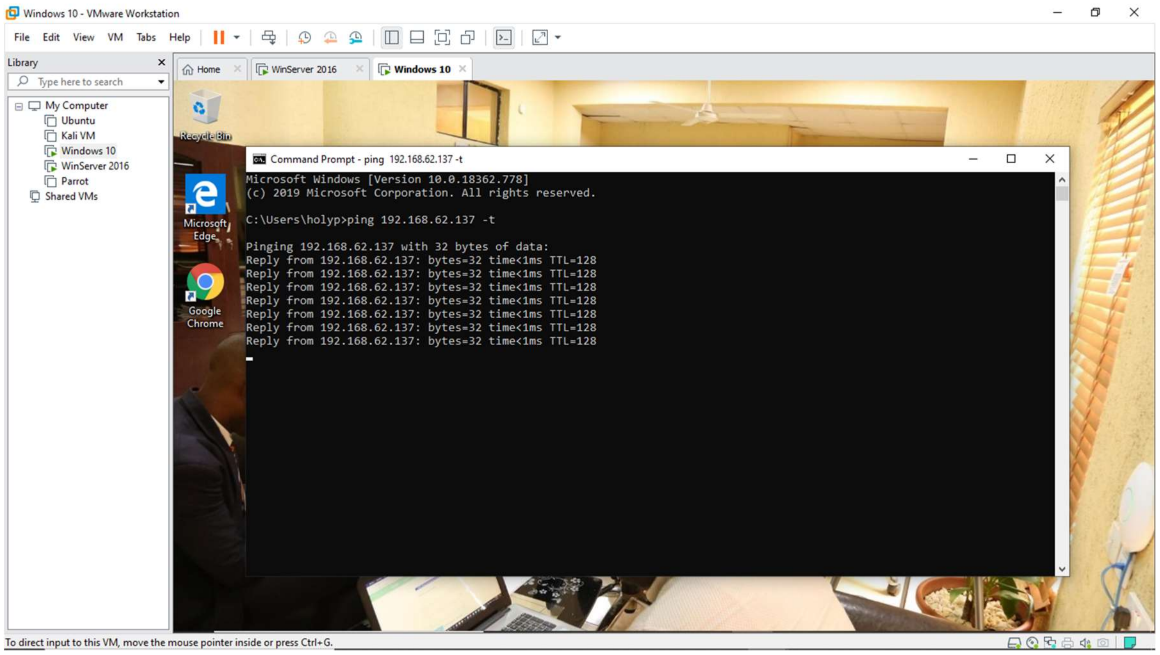
```
Acquiring network traffic from "\Device\NPF_{253A611F-7BB4-4[illegible]52-9A89-79786834FF86}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.17-WIN64 GRE (Build 199)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=3716)
```

Figure 3.26: Snort Honeypot is set in IDS mode and ready to start capturing.

A. Perform Intrusion Detection

The next thing to do was to navigate to the Windows 10 machine which will serve as the attack machine from where malicious traffic will be sent.

On the Windows 10 machine as shown in Figure 3.27, Command Prompt window is launched and type ***ping 192.168.62.137 –t*** and press ***Enter***. 192.168.62.137 is the IP address of the Windows Server 2016 on which we have the Snort Honeypot installed.

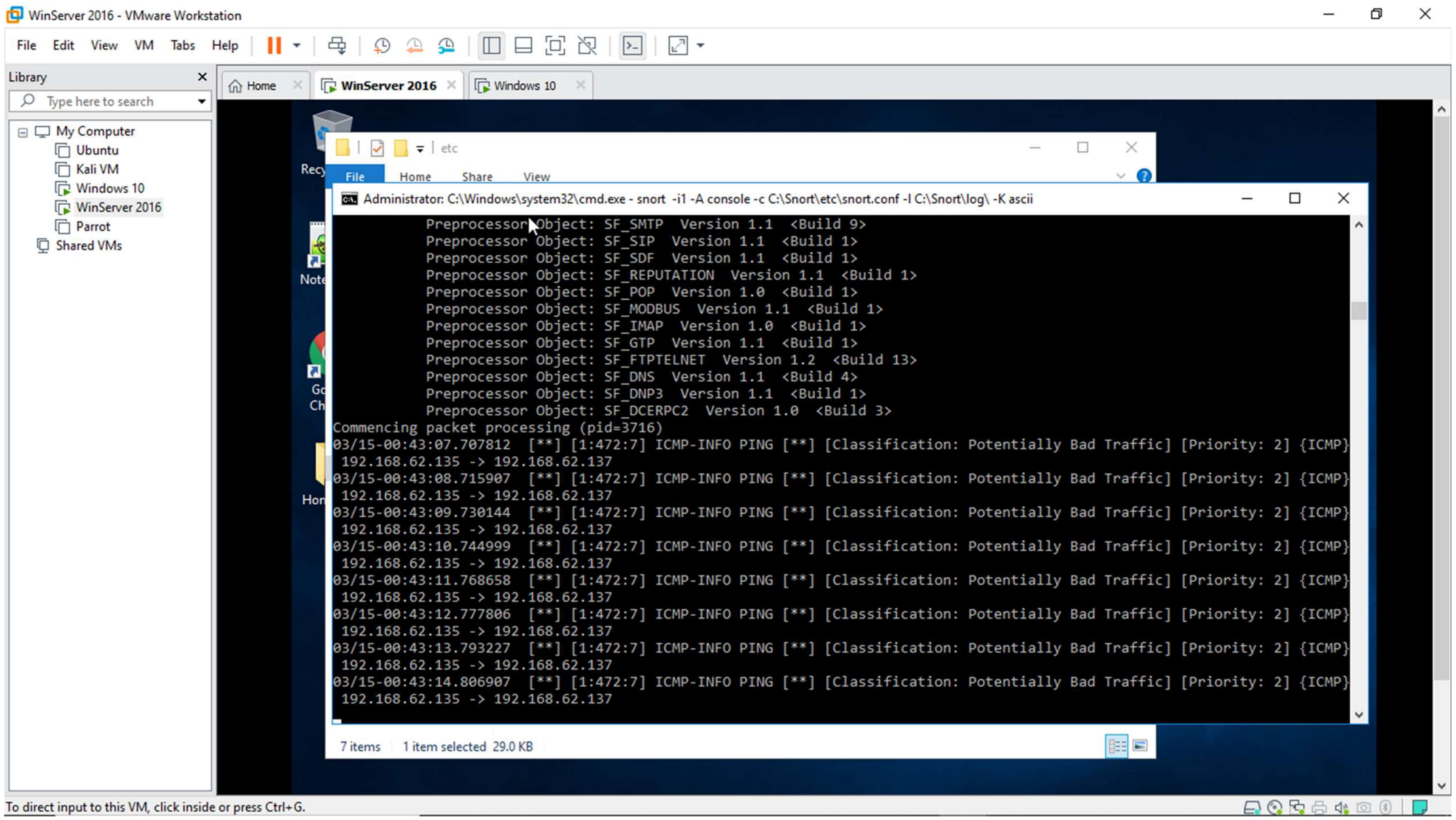


```
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\holyp>ping 192.168.62.137 -t

Pinging 192.168.62.137 with 32 bytes of data:
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
Reply from 192.168.62.137: bytes=32 time<1ms TTL=128
```

Figure 3.27: Windows 10 machine sending malicious traffic to the Windows 2016 server

As soon as this is done, navigation to our Windows 2016 server machine is done to see what happens with the malicious traffic being sent from the attack machine.

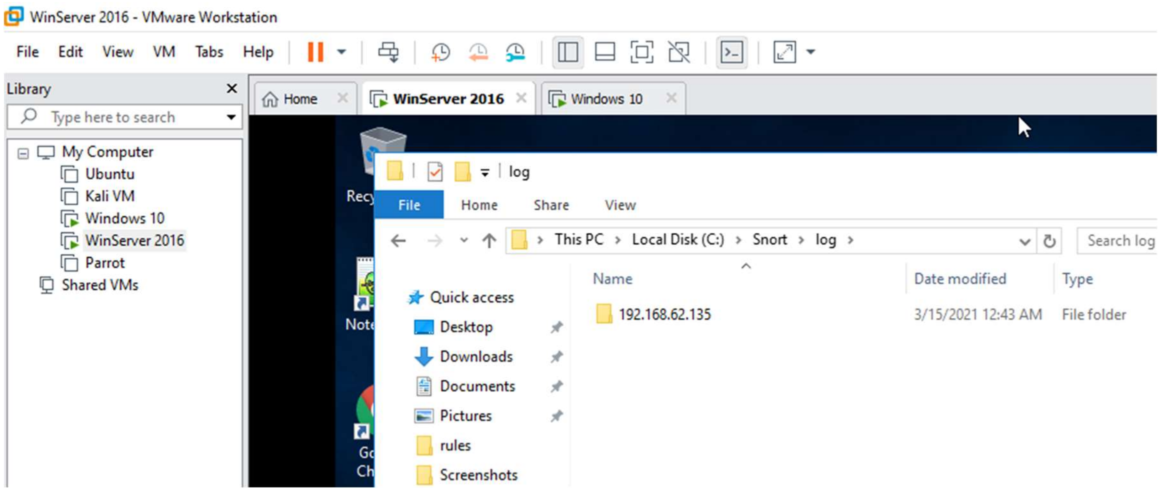
It is observed that the Snort Honeypot triggers alarm, as shown in Figure 3.28. Press ***Ctrl+C*** to stop Snort. Snort exits.


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
etc
File Home Share View
7 items 1 item selected 29.0 KB
To direct input to this VM, click inside or press Ctrl+G.

```
Administrator: C:\Windows\system32\cmd.exe - snort -i1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log\ -K ascii
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=3716)
03/15-00:43:07.707812  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:08.715907  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:09.730144  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:10.744999  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:11.768658  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:12.777806  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:13.793227  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/15-00:43:14.806907  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
```

Figure 3.28: Windows 2016 Server receiving malicious traffic from Windows 10 server

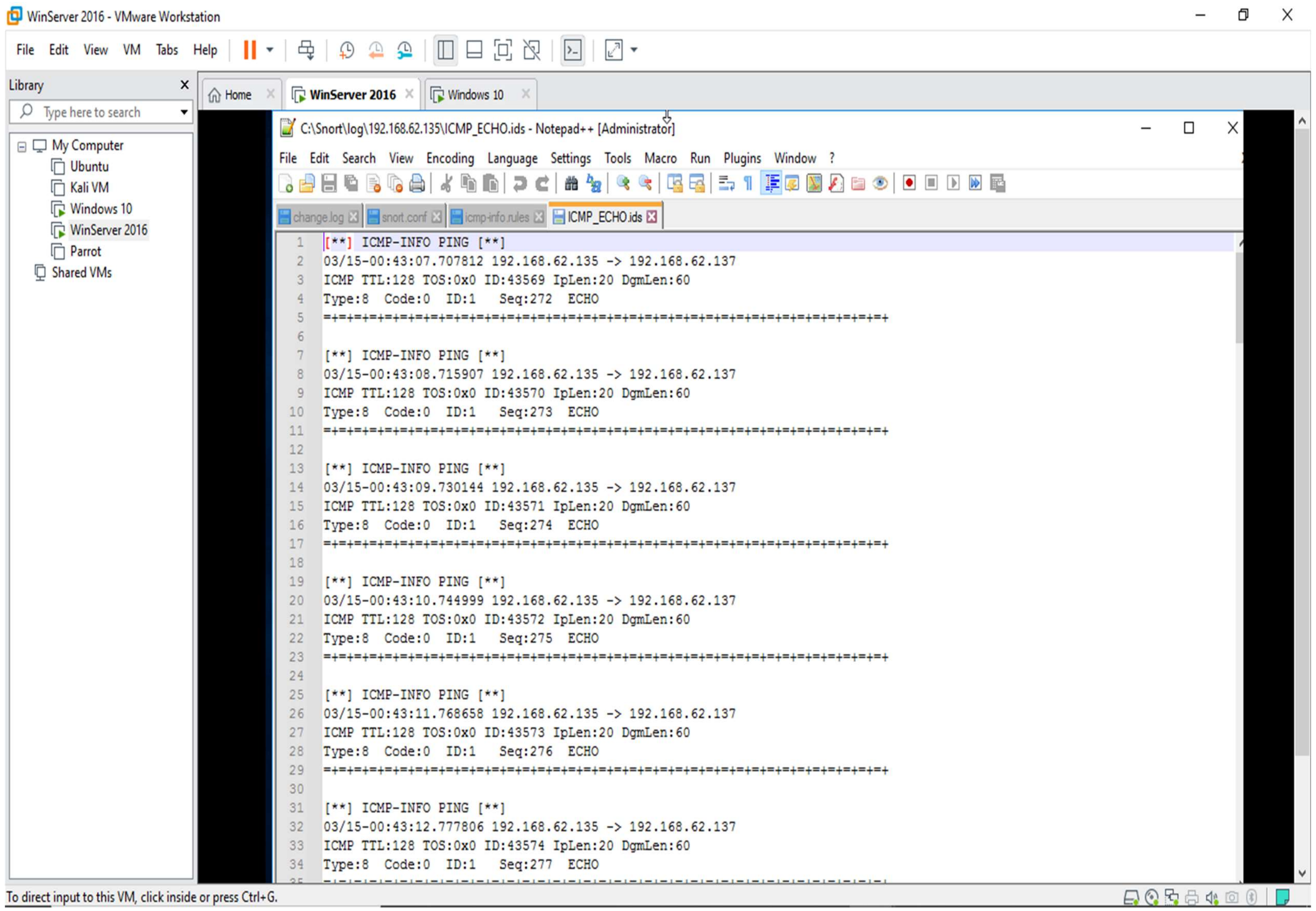
It is now safe to navigate to the ***C:\Snort\log\192.168.62.135*** folder, and open the ***ICMP_ECHO.ids*** file with Notepad++.



Name	Date modified	Type
192.168.62.135	3/15/2021 12:43 AM	File folder

Figure 3.29: A folder has been created for the attack machine where the log of activities are stored.

In Figure 3.29, it is seen that all the log entries from the particular Windows 10 machine are saved in the ***ICMP_ECHO.ids*** file. This invariably signifies that whenever an attacker tries to connect or communicate with the machine, Snort immediately triggers an alarm as shown in Figure 3.30. This makes the network administrator to become alert and take certain extra security measures to break the communication with the attacker's machine.


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
C:\Snort\log\192.168.62.135\ICMP_ECHO.ids - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
snort.conf
icmp-info.rules
ICMP_ECHO.ids
To direct input to this VM, click inside or press Ctrl+G.

```
[**] ICMP-INFO PING [**]
03/15-00:43:07.707812 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:43569 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:272  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/15-00:43:08.715907 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:43570 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:273  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/15-00:43:09.730144 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:43571 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:274  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/15-00:43:10.744999 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:43572 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:275  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/15-00:43:11.768658 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:43573 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:276  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/15-00:43:12.777806 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:43574 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:277  ECHO
```

Figure 3.30: The ICMP_ECHO.ids file content showing traffic logs

CAPTCHA Deployment

Every Web Administrator is familiar with some of the issues faced on web applications by spammers and bots. If not properly checked, these spams and bots can degrade the quality of sites by manipulating comments sections, multiple registrations or attack your contact forms. In recent times, Completely Automated Public Turing Tests To Tell Computers and Humans Apart (CAPTCHAs) remains one of the most effective and efficient ways to prevent spams and bots from spamming your website. Although very easy to implement, and even hackers often try to bypass them, it remains a solid line of defense for most web forms. In this section, CAPTCHA is implemented on one of our already existing site depicted in Figure 3.31 (https://att.lmu.edu.ng).

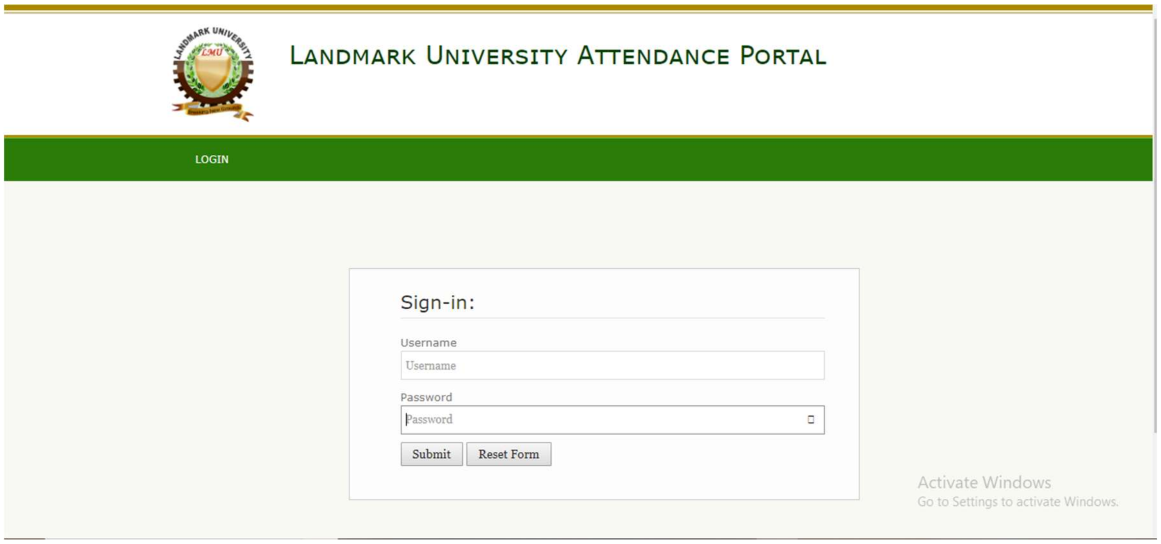
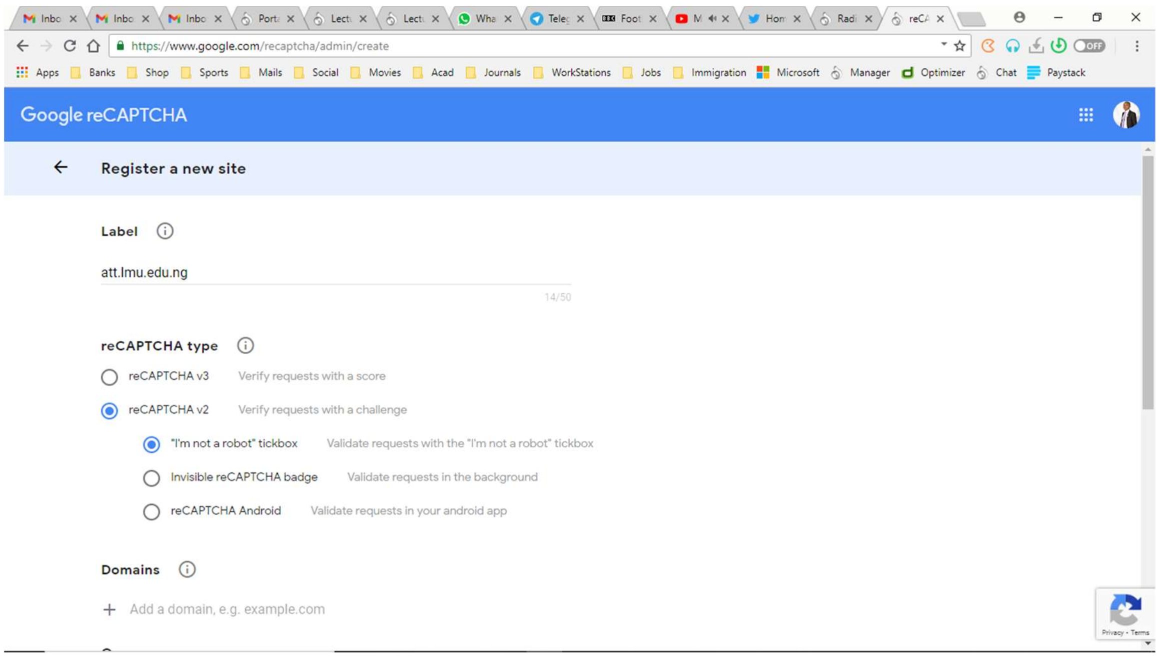

LANDMARK UNIVERSITY ATTENDANCE PORTAL
LOGIN
Sign-in:
Username
Password
Submit
Reset Form
Activate Windows
Go to Settings to activate Windows.

Figure 3.31: The Default login screen of the Attendance portal



Register a new site

Label

att.lmu.edu.ng

14/50

reCAPTCHA type

○ reCAPTCHA v3 Verify requests with a score

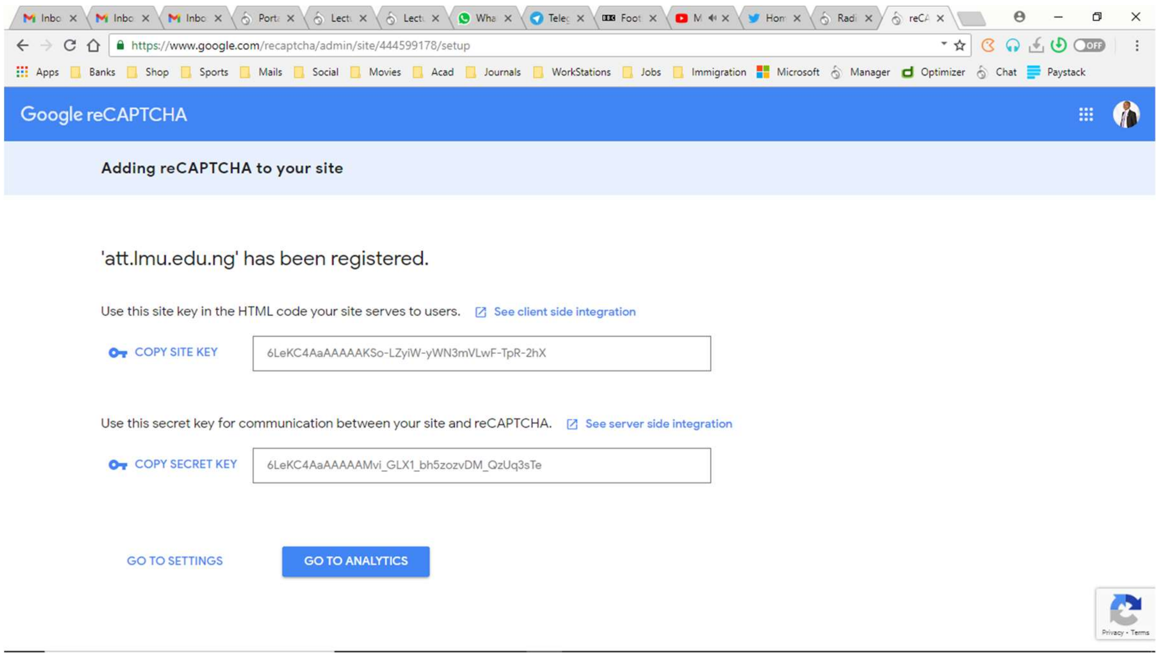
● reCAPTCHA v2 Verify requests with a challenge

- ● "I'm not a robot" tickbox Validate requests with the "I'm not a robot" tickbox
- ○ Invisible reCAPTCHA badge Validate requests in the background
- ○ reCAPTCHA Android Validate requests in your android app

Domains

+ Add a domain, e.g. example.com

Figure 3.32a: Domain registration of the attendance portal on the CAPTCHA platform



Adding reCAPTCHA to your site

'att.lmu.edu.ng' has been registered.

Use this site key in the HTML code your site serves to users. See client side integration

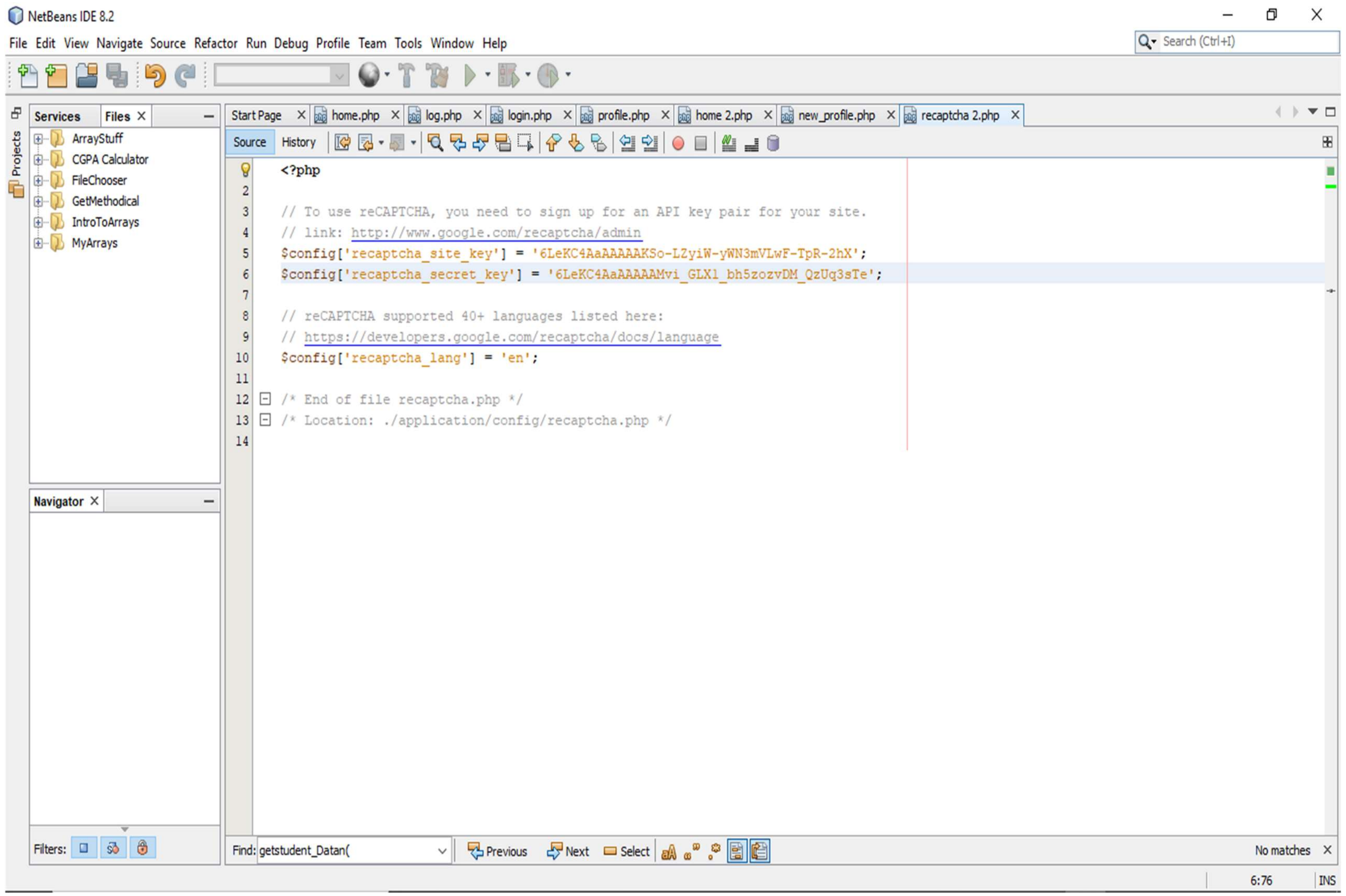
COPY SITE KEY 6LeKC4AaAAAAAKSo-LZyiW-yWN3mVLwF-TpR-2hX

Use this secret key for communication between your site and reCAPTCHA. See server side integration

COPY SECRET KEY 6LeKC4AaAAAAAMvi_GLX1_bh5zozvDM_QzUq3sTe

Figure 3.32b: The assigned site key as well as the secret key for the Attendance portal

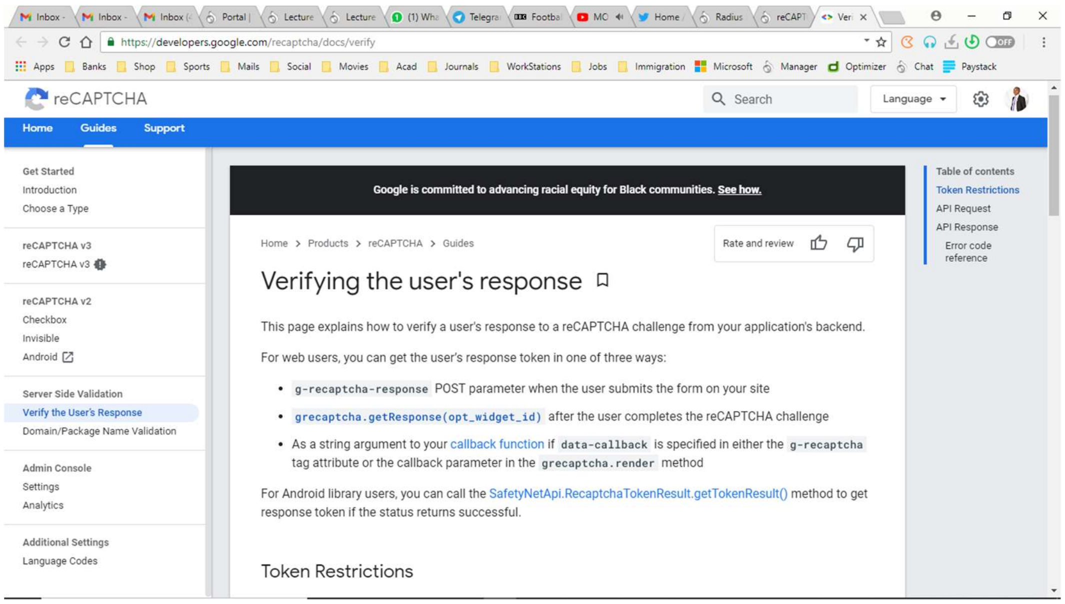
First, there has to be a registration on the site on the CAPTCHA platform so a personal encryption and decryption key can be assigned to the site as shown in Figure 3.32a and 3.32b.



```php
<?php

// To use reCAPTCHA, you need to sign up for an API key pair for your site.
// link: http://www.google.com/recaptcha/admin
$config['recaptcha_site_key'] = '6LeKC4AaAAAAAKSo-LZyiW-yWN3mVLwF-TpR-2hX';
$config['recaptcha_secret_key'] = '6LeKC4AaAAAAAMvi_GLX1_bh5zozvDM_QzUq3sTe';

// reCAPTCHA supported 40+ languages listed here:
// https://developers.google.com/recaptcha/docs/language
$config['recaptcha_lang'] = 'en';

/* End of file recaptcha.php */
/* Location: ./application/config/recaptcha.php */
```

Figure 3.33: Implementing the given keys in the config folder of the attendance portal

Next, the code for the verification and approval process from the backend is implemented on the application server. Having received both the site key and the secret key shown in Figure 3.33, it is implemented in the config folder of the application being developed. Figures 3.34a to 3.34d shows the steps in implementing the verification and approval procedures.



Google is committed to advancing racial equity for Black communities. See how.

Home > Products > reCAPTCHA > Guides

Verifying the user's response

This page explains how to verify a user's response to a reCAPTCHA challenge from your application's backend.

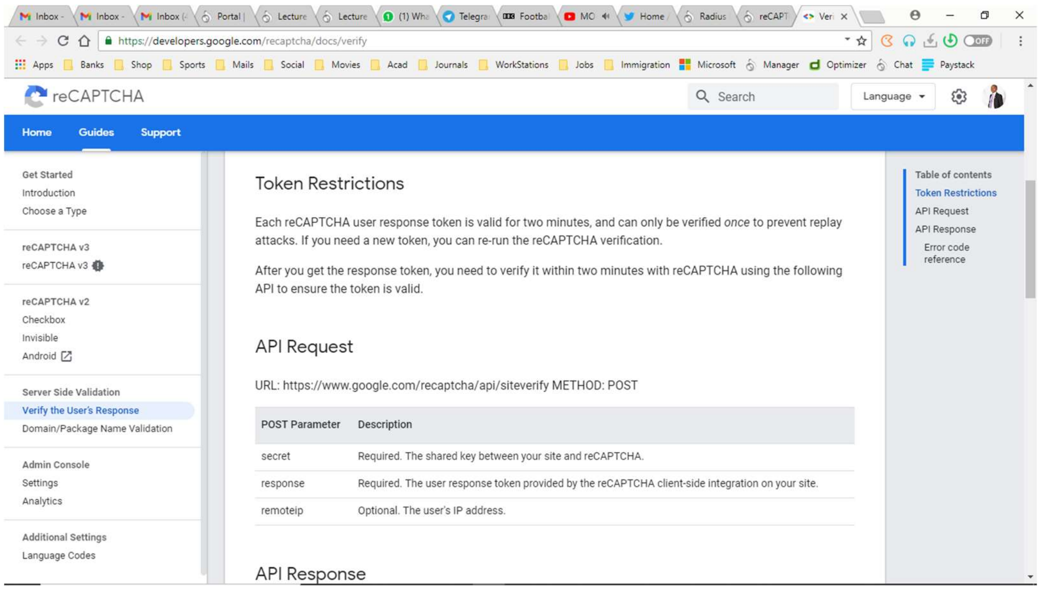
For web users, you can get the user's response token in one of three ways:

- `g-recaptcha-response` POST parameter when the user submits the form on your site
- `grecaptcha.getResponse(opt_widget_id)` after the user completes the reCAPTCHA challenge
- As a string argument to your callback function if `data-callback` is specified in either the `g-recaptcha` tag attribute or the callback parameter in the `grecaptcha.render` method

For Android library users, you can call the SafetyNetApi.RecaptchaTokenResult.getTokenResult() method to get response token if the status returns successful.

Token Restrictions

Figure 3.34a: Verification of the user's CAPTCHA response



Token Restrictions

Each reCAPTCHA user response token is valid for two minutes, and can only be verified *once* to prevent replay attacks. If you need a new token, you can re-run the reCAPTCHA verification.

After you get the response token, you need to verify it within two minutes with reCAPTCHA using the following API to ensure the token is valid.

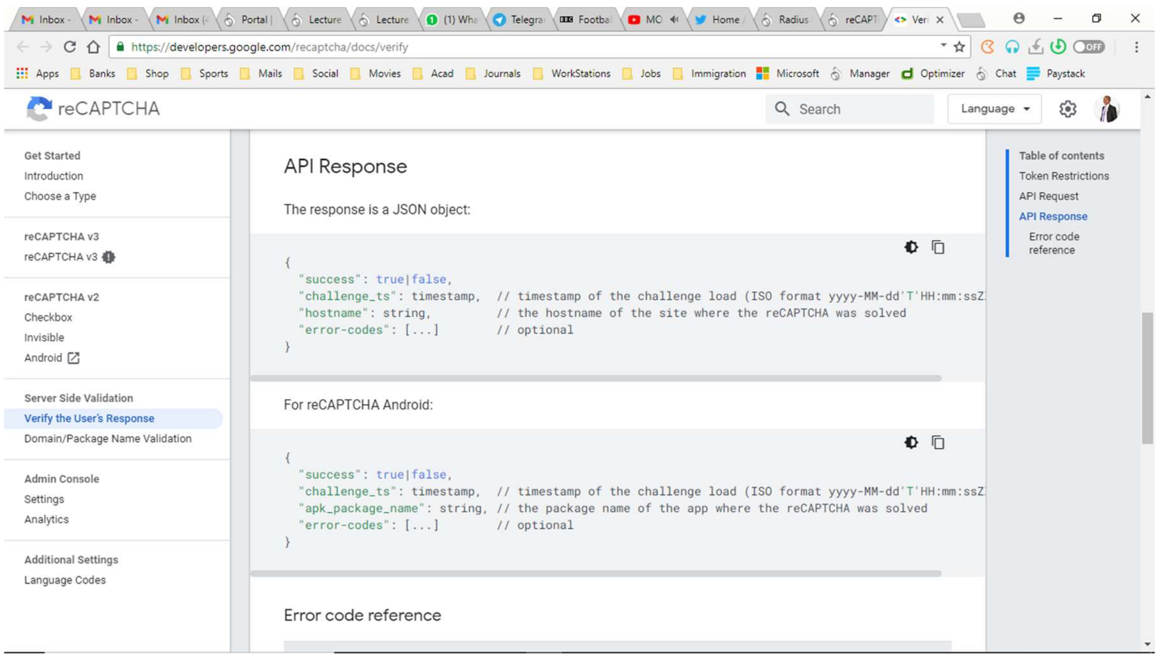
API Request

URL: https://www.google.com/recaptcha/api/siteverify METHOD: POST

POST Parameter	Description
secret	Required. The shared key between your site and reCAPTCHA.
response	Required. The user response token provided by the reCAPTCHA client-side integration on your site.
remoteip	Optional. The user's IP address.

API Response

Figure 3.34b: Token Restrictions and API Request for the CAPTCHA response



API Response

The response is a JSON object:

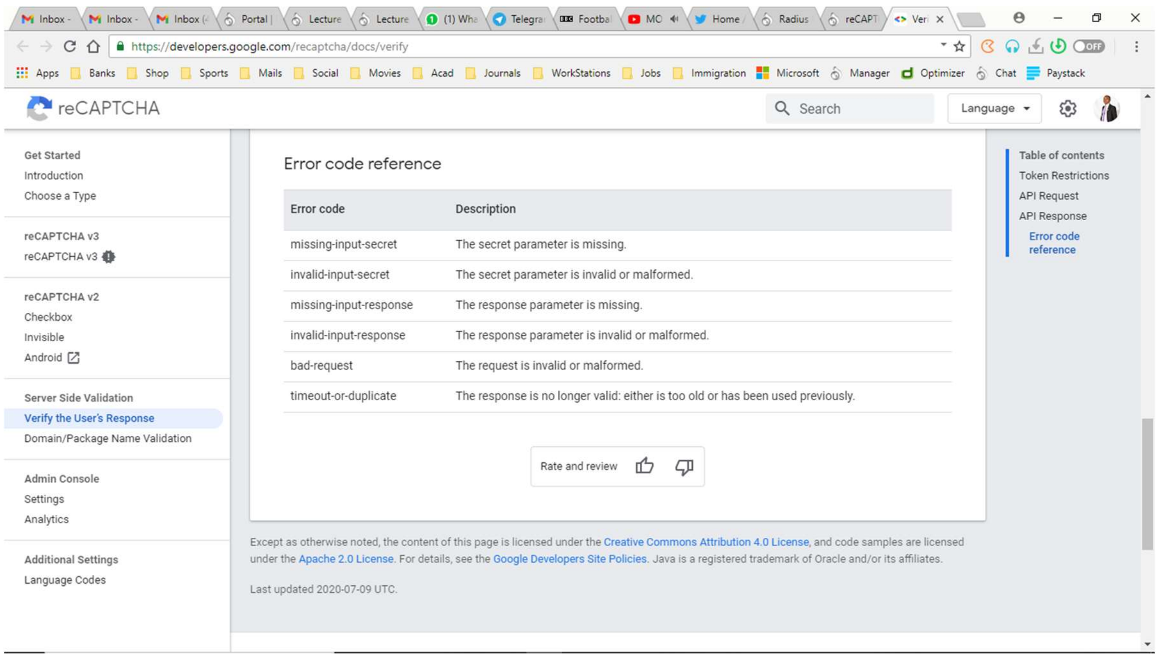
```
{
  "success": true|false,
  "challenge_ts": timestamp,  // timestamp of the challenge load (ISO format yyyy-MM-dd'T'HH:mm:ssZ
  "hostname": string,         // the hostname of the site where the reCAPTCHA was solved
  "error-codes": [...]        // optional
}
```

For reCAPTCHA Android:

```
{
  "success": true|false,
  "challenge_ts": timestamp,  // timestamp of the challenge load (ISO format yyyy-MM-dd'T'HH:mm:ssZ
  "apk_package_name": string, // the package name of the app where the reCAPTCHA was solved
  "error-codes": [...]        // optional
}
```

Error code reference

Figure 3.34c: API response capturing for the CAPTCHA response



Error code reference

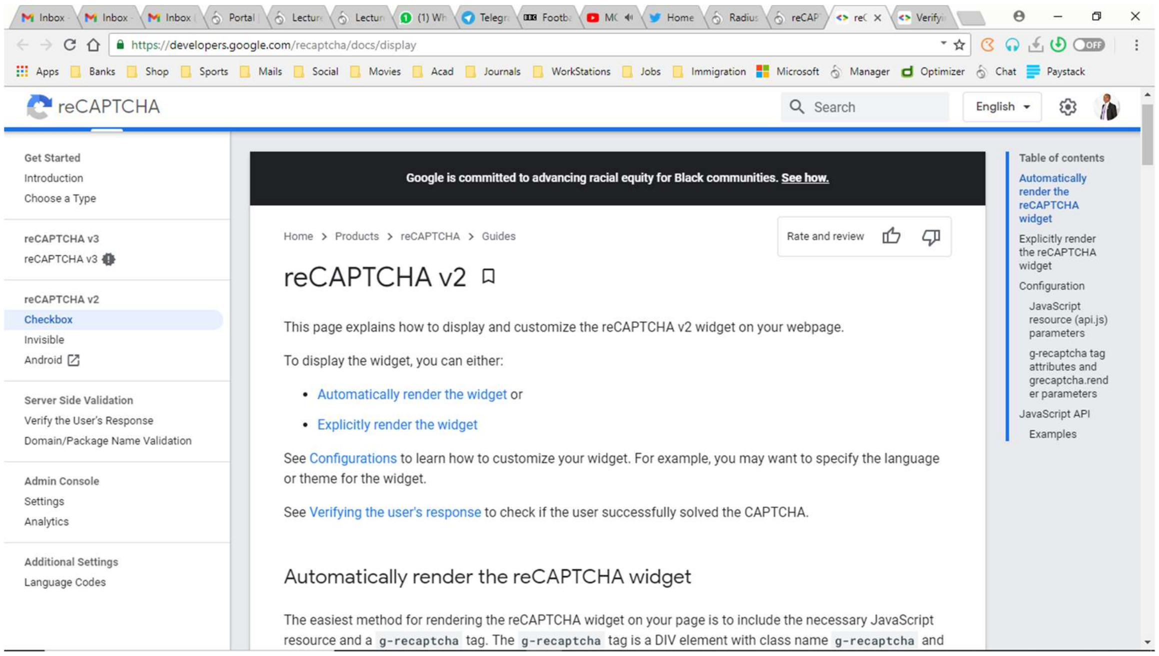
Error code	Description
missing-input-secret	The secret parameter is missing.
invalid-input-secret	The secret parameter is invalid or malformed.
missing-input-response	The response parameter is missing.
invalid-input-response	The response parameter is invalid or malformed.
bad-request	The request is invalid or malformed.
timeout-or-duplicate	The response is no longer valid: either is too old or has been used previously.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License, and code samples are licensed under the Apache 2.0 License. For details, see the Google Developers Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2020-07-09 UTC.

Figure 3.34d: Error code reference for the CAPTCHA response

Next, the code for the display and customization of the CAPTCHA widget is implemented on the application server. Figures 3.35a to 3.35f shows the steps in implementing the display and customization procedures.


https://developers.google.com/recaptcha/docs/display
reCAPTCHA
Search
English
Get Started
Introduction
Choose a Type
reCAPTCHA v3
reCAPTCHA v2
Checkbox
Invisible
Android
Server Side Validation
Verify the User's Response
Domain/Package Name Validation
Admin Console
Settings
Analytics
Additional Settings
Language Codes
Home > Products > reCAPTCHA > Guides
Rate and review
Table of contents
Automatically render the reCAPTCHA widget
Explicitly render the reCAPTCHA widget
Configuration
JavaScript resource (api.js) parameters
g-recaptcha tag attributes and grecaptcha.render parameters
JavaScript API
Examples

Google is committed to advancing racial equity for Black communities. See how.

reCAPTCHA v2

This page explains how to display and customize the reCAPTCHA v2 widget on your webpage.

To display the widget, you can either:

- Automatically render the widget or
- Explicitly render the widget

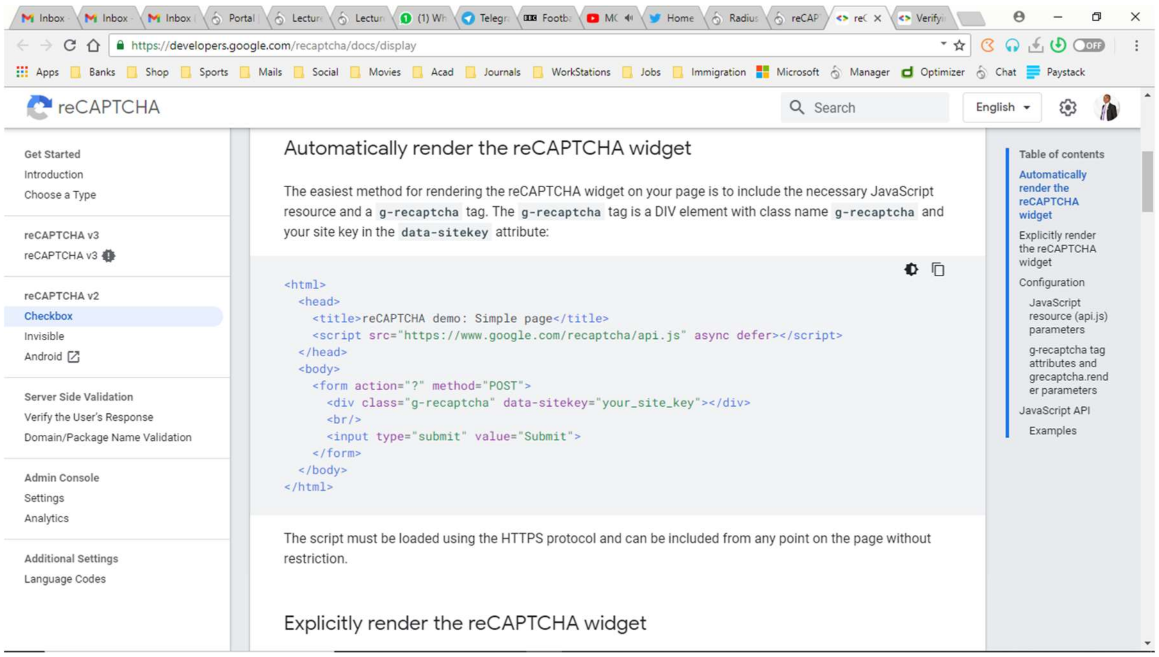
See Configurations to learn how to customize your widget. For example, you may want to specify the language or theme for the widget.

See Verifying the user's response to check if the user successfully solved the CAPTCHA.

Automatically render the reCAPTCHA widget

The easiest method for rendering the reCAPTCHA widget on your page is to include the necessary JavaScript resource and a `g-recaptcha` tag. The `g-recaptcha` tag is a DIV element with class name `g-recaptcha` and

Figure 3.35a: Displaying and Customizing the CAPTCHA on the application site


https://developers.google.com/recaptcha/docs/display
reCAPTCHA
Search
English
Get Started
Introduction
Choose a Type
reCAPTCHA v3
reCAPTCHA v2
Checkbox
Invisible
Android
Server Side Validation
Verify the User's Response
Domain/Package Name Validation
Admin Console
Settings
Analytics
Additional Settings
Language Codes
Table of contents
Automatically render the reCAPTCHA widget
Explicitly render the reCAPTCHA widget
Configuration
JavaScript resource (api.js) parameters
g-recaptcha tag attributes and grecaptcha.render parameters
JavaScript API
Examples

Automatically render the reCAPTCHA widget

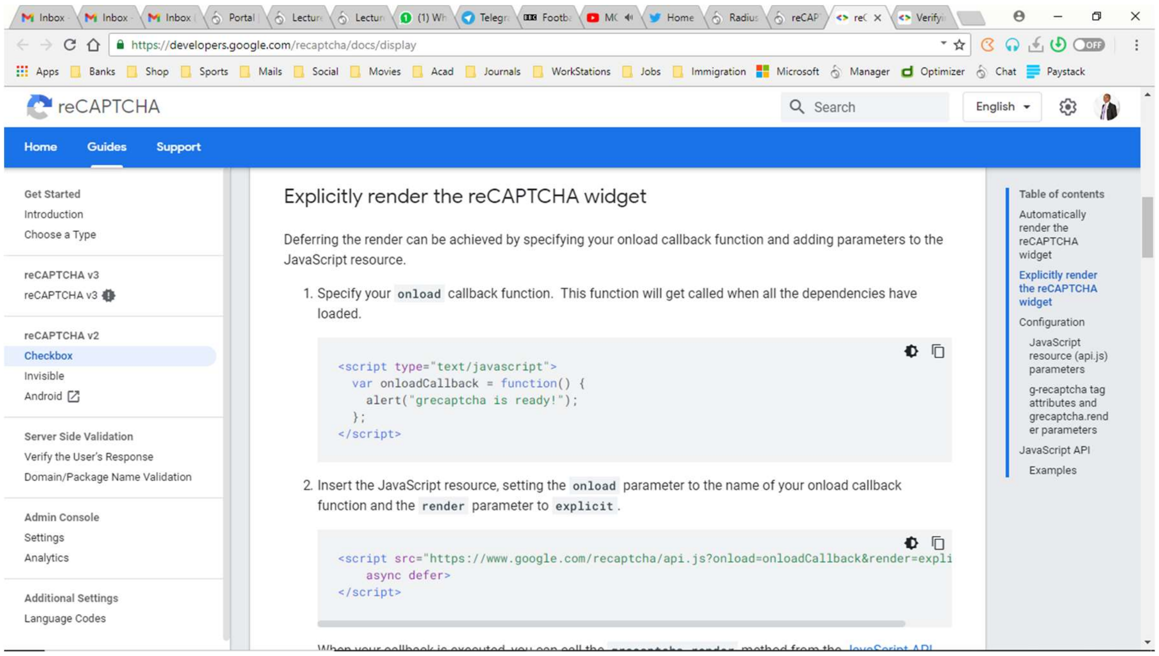
The easiest method for rendering the reCAPTCHA widget on your page is to include the necessary JavaScript resource and a `g-recaptcha` tag. The `g-recaptcha` tag is a DIV element with class name `g-recaptcha` and your site key in the `data-sitekey` attribute:

```
<html>
  <head>
    <title>reCAPTCHA demo: Simple page</title>
    <script src="https://www.google.com/recaptcha/api.js" async defer></script>
  </head>
  <body>
    <form action="?" method="POST">
      <div class="g-recaptcha" data-sitekey="your_site_key"></div>
      <br/>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

The script must be loaded using the HTTPS protocol and can be included from any point on the page without restriction.

Explicitly render the reCAPTCHA widget

Figure 3.35b: Automatically rendering the reCAPTCHA widget



Explicitly render the reCAPTCHA widget

Deferring the render can be achieved by specifying your onload callback function and adding parameters to the JavaScript resource.

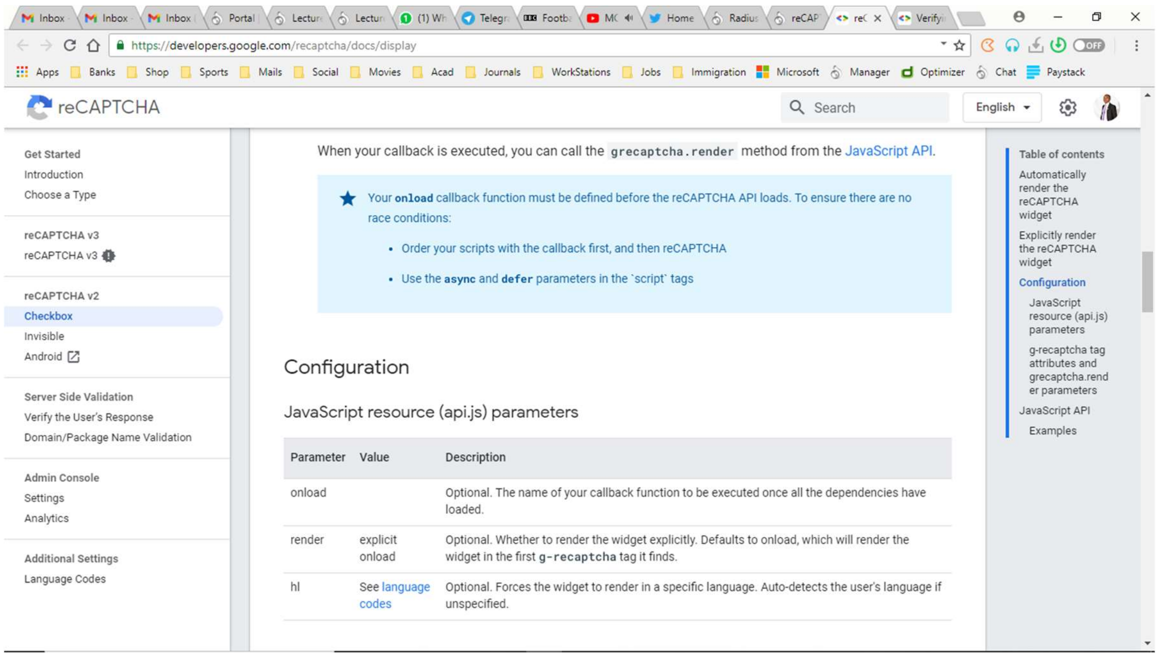
1. Specify your `onload` callback function. This function will get called when all the dependencies have loaded.

```
<script type="text/javascript">
  var onloadCallback = function() {
    alert("grecaptcha is ready!");
  };
</script>
```

2. Insert the JavaScript resource, setting the `onload` parameter to the name of your onload callback function and the `render` parameter to `explicit`.

```
<script src="https://www.google.com/recaptcha/api.js?onload=onloadCallback&render=expli
    async defer>
</script>
```

Figure 3.35c: Explicitly rendering the reCAPTCHA widget


https://developers.google.com/recaptcha/docs/display
reCAPTCHA
Search
English
Get Started
Introduction
Choose a Type
reCAPTCHA v3
reCAPTCHA v3
reCAPTCHA v2
Checkbox
Invisible
Android
Server Side Validation
Verify the User's Response
Domain/Package Name Validation
Admin Console
Settings
Analytics
Additional Settings
Language Codes
Table of contents
Automatically render the reCAPTCHA widget
Explicitly render the reCAPTCHA widget
Configuration
JavaScript resource (api.js) parameters
g-recaptcha tag attributes and grecaptcha.render parameters
JavaScript API
Examples

When your callback is executed, you can call the `grecaptcha.render` method from the JavaScript API.

Your **onload** callback function must be defined before the reCAPTCHA API loads. To ensure there are no race conditions:

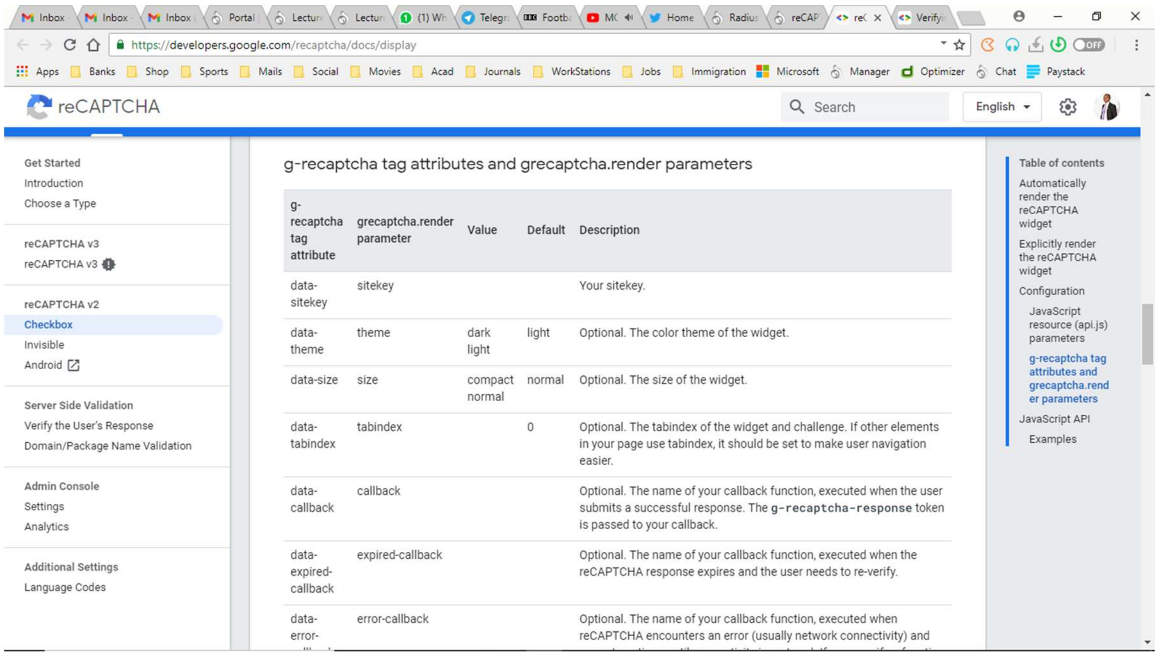
- Order your scripts with the callback first, and then reCAPTCHA
- Use the **async** and **defer** parameters in the `script` tags

Configuration

JavaScript resource (api.js) parameters

Parameter	Value	Description
onload		Optional. The name of your callback function to be executed once all the dependencies have loaded.
render	explicit onload	Optional. Whether to render the widget explicitly. Defaults to onload, which will render the widget in the first `g-recaptcha` tag it finds.
hl	See language codes	Optional. Forces the widget to render in a specific language. Auto-detects the user's language if unspecified.

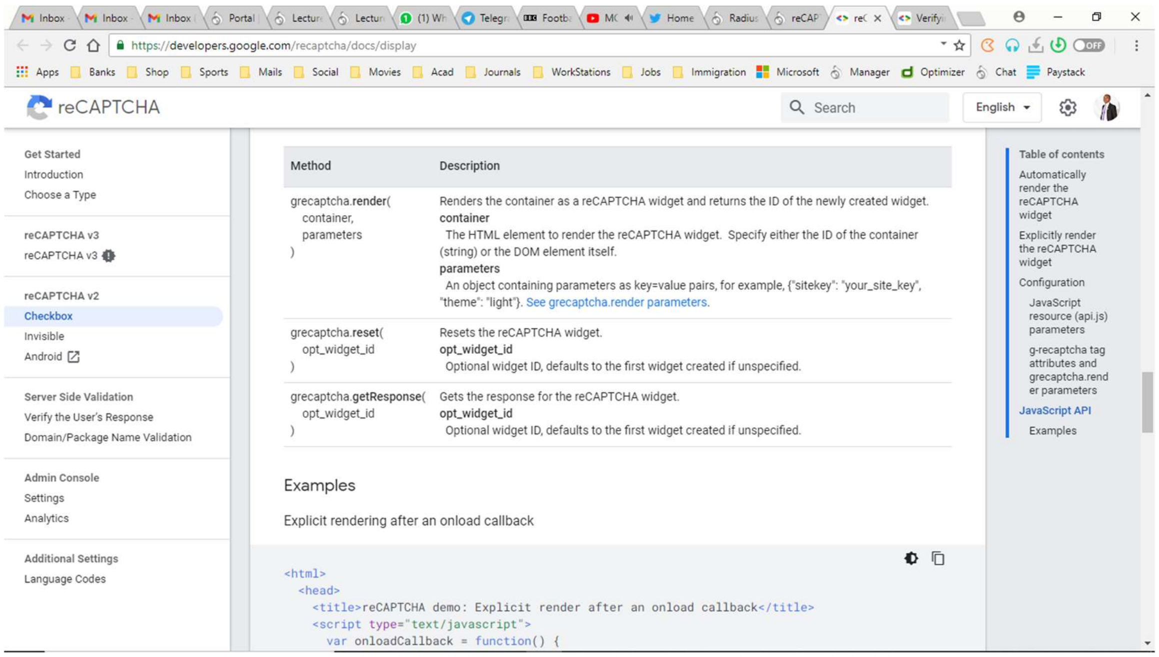
Figure 3.35d: Configuring the reCAPTCHA widget with javascript resource api


https://developers.google.com/recaptcha/docs/display
reCAPTCHA
Search
English
Get Started
Introduction
Choose a Type
reCAPTCHA v3
reCAPTCHA v2
Checkbox
Invisible
Android
Server Side Validation
Verify the User's Response
Domain/Package Name Validation
Admin Console
Settings
Analytics
Additional Settings
Language Codes
Table of contents
Automatically render the reCAPTCHA widget
Explicitly render the reCAPTCHA widget
Configuration
JavaScript resource (api.js) parameters
g-recaptcha tag attributes and grecaptcha.render parameters
JavaScript API
Examples

g-recaptcha tag attributes and grecaptcha.render parameters

g-recaptcha tag attribute	grecaptcha.render parameter	Value	Default	Description
data-sitekey	sitekey			Your sitekey.
data-theme	theme	dark light	light	Optional. The color theme of the widget.
data-size	size	compact normal	normal	Optional. The size of the widget.
data-tabindex	tabindex		0	Optional. The tabindex of the widget and challenge. If other elements in your page use tabindex, it should be set to make user navigation easier.
data-callback	callback			Optional. The name of your callback function, executed when the user submits a successful response. The `g-recaptcha-response` token is passed to your callback.
data-expired-callback	expired-callback			Optional. The name of your callback function, executed when the reCAPTCHA response expires and the user needs to re-verify.
data-error-	error-callback			Optional. The name of your callback function, executed when reCAPTCHA encounters an error (usually network connectivity) and

Figure 3.35e: Attributes associated with the g-reCAPTCHA widget


https://developers.google.com/recaptcha/docs/display
reCAPTCHA
Search
English
Get Started
Introduction
Choose a Type
reCAPTCHA v3
reCAPTCHA v2
Checkbox
Invisible
Android
Server Side Validation
Verify the User's Response
Domain/Package Name Validation
Admin Console
Settings
Analytics
Additional Settings
Language Codes
Table of contents
Automatically render the reCAPTCHA widget
Explicitly render the reCAPTCHA widget
Configuration
JavaScript resource (api.js) parameters
g-recaptcha tag attributes and grecaptcha.render parameters
JavaScript API
Examples

Method	Description
grecaptcha.render(container, parameters)	Renders the container as a reCAPTCHA widget and returns the ID of the newly created widget. **container** The HTML element to render the reCAPTCHA widget. Specify either the ID of the container (string) or the DOM element itself. **parameters** An object containing parameters as key=value pairs, for example, {"sitekey": "your_site_key", "theme": "light"}. See grecaptcha.render parameters.
grecaptcha.reset(opt_widget_id)	Resets the reCAPTCHA widget. **opt_widget_id** Optional widget ID, defaults to the first widget created if unspecified.
grecaptcha.getResponse(opt_widget_id)	Gets the response for the reCAPTCHA widget. **opt_widget_id** Optional widget ID, defaults to the first widget created if unspecified.

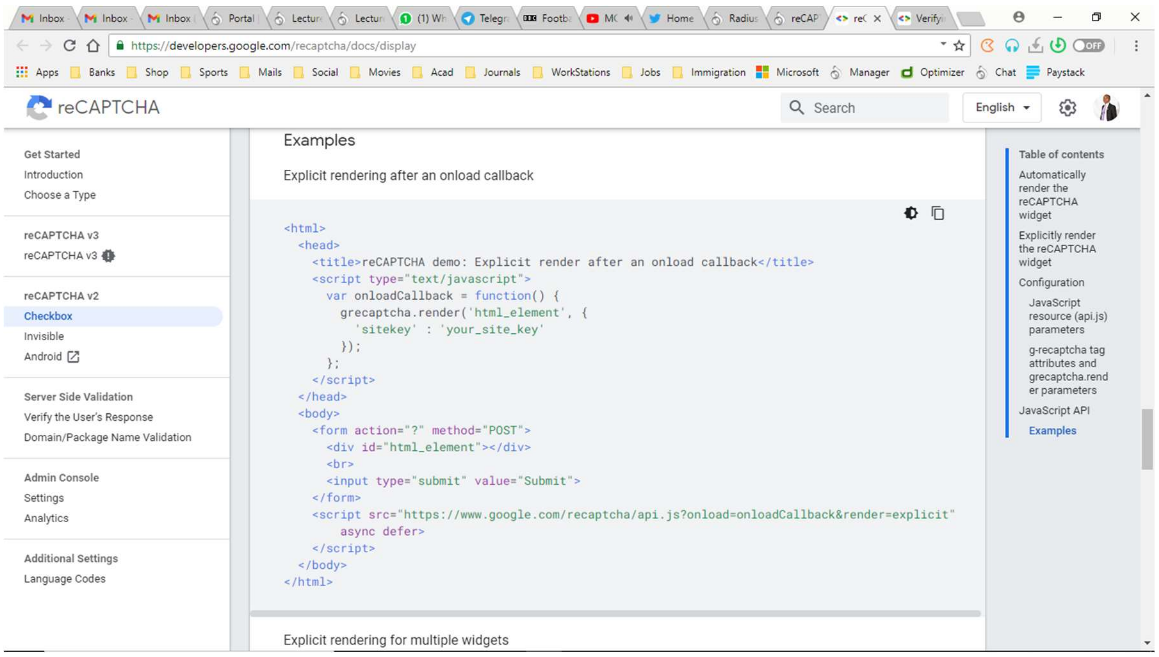
Examples

Explicit rendering after an onload callback

```
<html>
  <head>
    <title>reCAPTCHA demo: Explicit render after an onload callback</title>
    <script type="text/javascript">
      var onloadCallback = function() {
```

Figure 3.35f: Methods and description in rendering the reCAPTCHA widget

Going forward, implemented sample codes for explicitly rendering CAPTCHA widget of the application server after an onload callback. Figure 3.36a to 3.36c shows some of the sample codes that helps with this implementation.



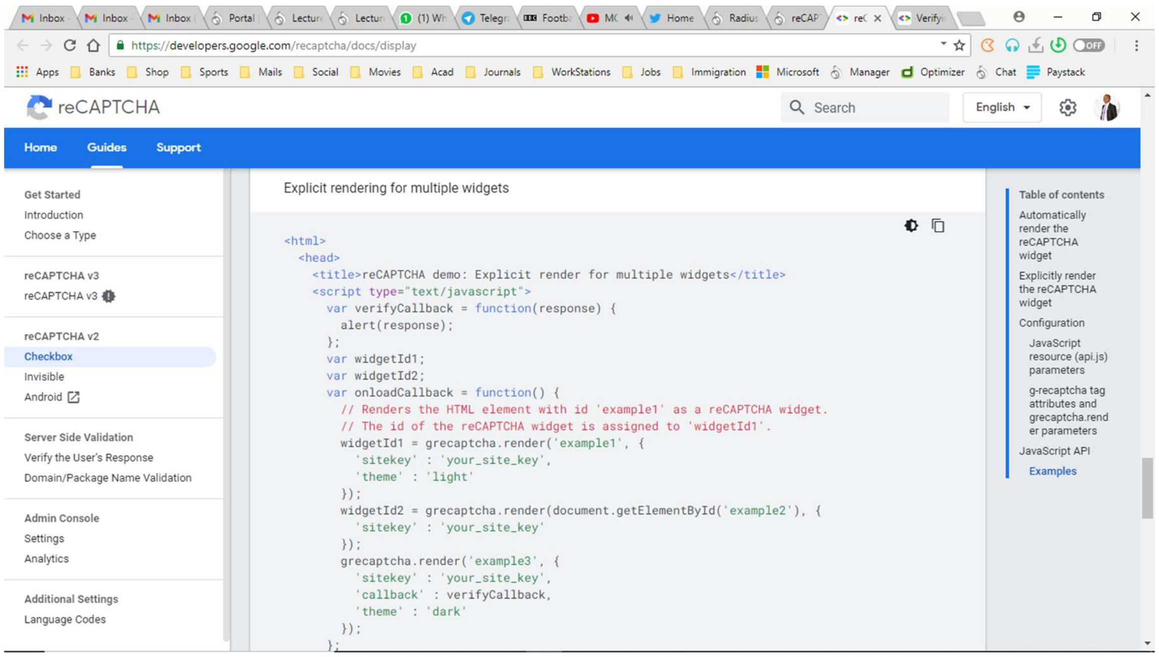
Examples

Explicit rendering after an onload callback

```
<html>
  <head>
    <title>reCAPTCHA demo: Explicit render after an onload callback</title>
    <script type="text/javascript">
      var onloadCallback = function() {
        grecaptcha.render('html_element', {
          'sitekey' : 'your_site_key'
        });
      };
    </script>
  </head>
  <body>
    <form action="?" method="POST">
      <div id="html_element"></div>
      <br>
      <input type="submit" value="Submit">
    </form>
    <script src="https://www.google.com/recaptcha/api.js?onload=onloadCallback&render=explicit"
        async defer>
    </script>
  </body>
</html>
```

Explicit rendering for multiple widgets

Figure 3.36a: Examples of rendering after an onload callback of the reCAPTCHA widget



Explicit rendering for multiple widgets

```
<html>
  <head>
    <title>reCAPTCHA demo: Explicit render for multiple widgets</title>
    <script type="text/javascript">
      var verifyCallback = function(response) {
        alert(response);
      };
      var widgetId1;
      var widgetId2;
      var onloadCallback = function() {
        // Renders the HTML element with id 'example1' as a reCAPTCHA widget.
        // The id of the reCAPTCHA widget is assigned to 'widgetId1'.
        widgetId1 = grecaptcha.render('example1', {
          'sitekey' : 'your_site_key',
          'theme' : 'light'
        });
        widgetId2 = grecaptcha.render(document.getElementById('example2'), {
          'sitekey' : 'your_site_key'
        });
        grecaptcha.render('example3', {
          'sitekey' : 'your_site_key',
          'callback' : verifyCallback,
          'theme' : 'dark'
        });
      };
```

Figure 3.36b: Examples of rendering for multiple reCAPTCHA widgets

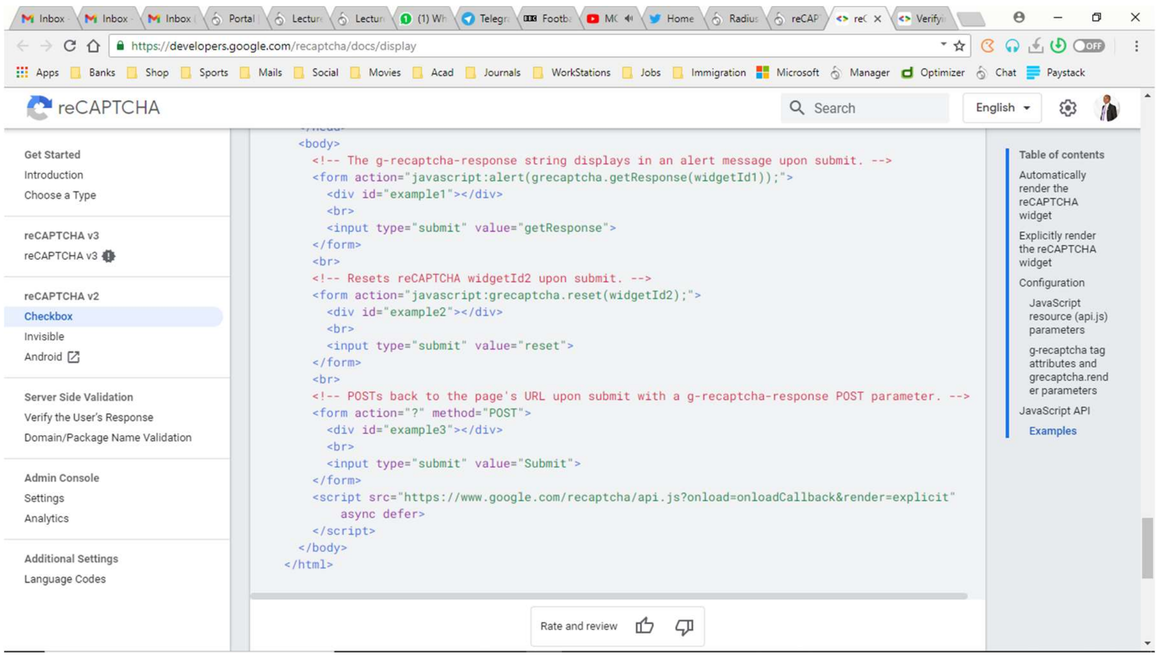
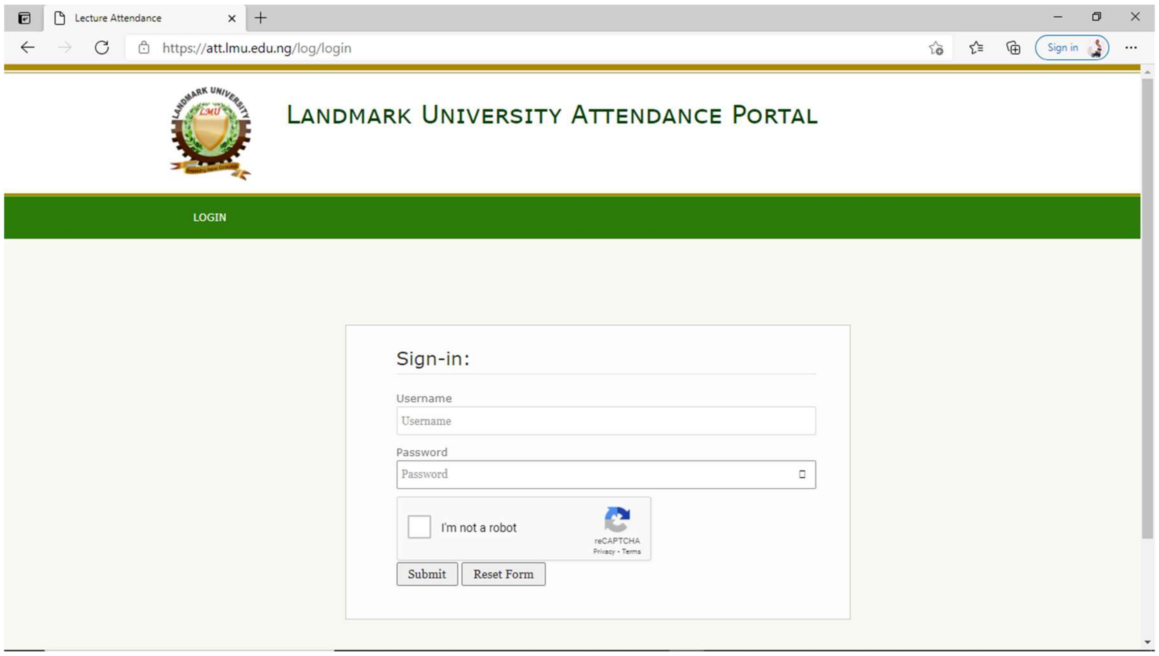

```
  <body>
    <!-- The g-recaptcha-response string displays in an alert message upon submit. -->
    <form action="javascript:alert(grecaptcha.getResponse(widgetId1));">
      <div id="example1"></div>
      <br>
      <input type="submit" value="getResponse">
    </form>
    <br>
    <!-- Resets reCAPTCHA widgetId2 upon submit. -->
    <form action="javascript:grecaptcha.reset(widgetId2);">
      <div id="example2"></div>
      <br>
      <input type="submit" value="reset">
    </form>
    <br>
    <!-- POSTs back to the page's URL upon submit with a g-recaptcha-response POST parameter. -->
    <form action="?" method="POST">
      <div id="example3"></div>
      <br>
      <input type="submit" value="Submit">
    </form>
    <script src="https://www.google.com/recaptcha/api.js?onload=onloadCallback&render=explicit"
        async defer>
    </script>
  </body>
</html>
```

Figure 3.36c: More examples of rendering for multiple reCAPTCHA widgets

Finally, the CAPTCHA IPS is incorporated into our Attendance portal application and that is depicted in Figure 3.37:


LANDMARK UNIVERSITY ATTENDANCE PORTAL

LOGIN

Sign-in:

Username

Password

I'm not a robot

Submit   Reset Form

Figure 3.37: The new face of the attendance portal having implemented the reCAPTCHA widget

Having implemented the CAPTCHA IPS, it is tested to show the workability of the Intrusion Prevention System and the outcome is displayed in Figures 3.38a and 3.38b:

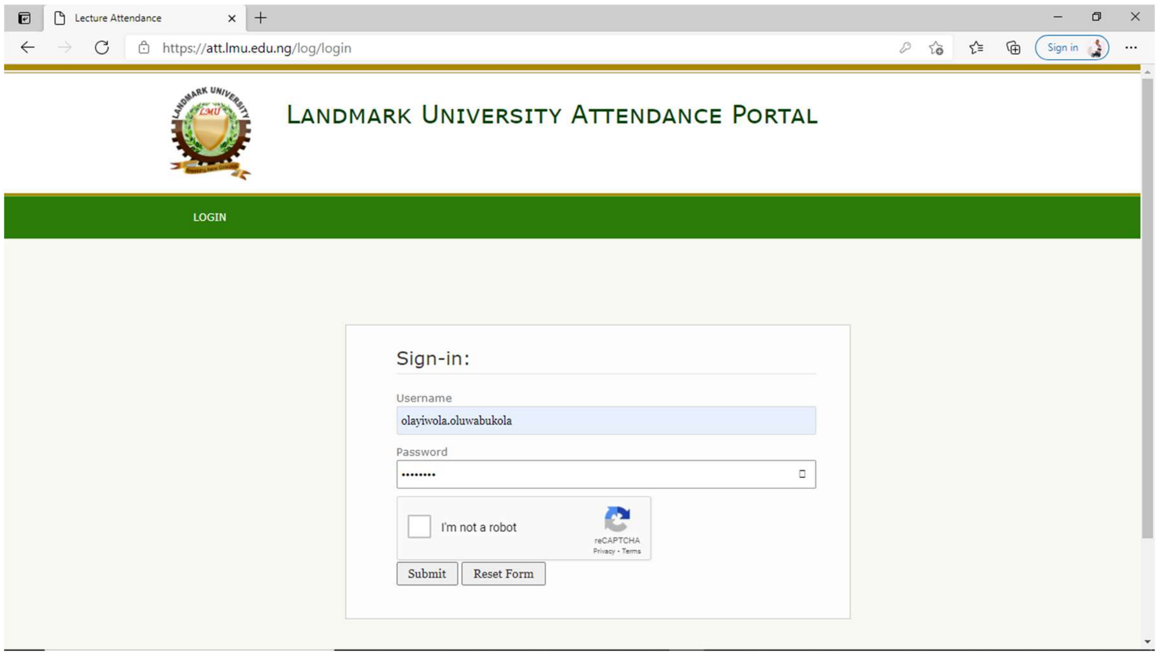
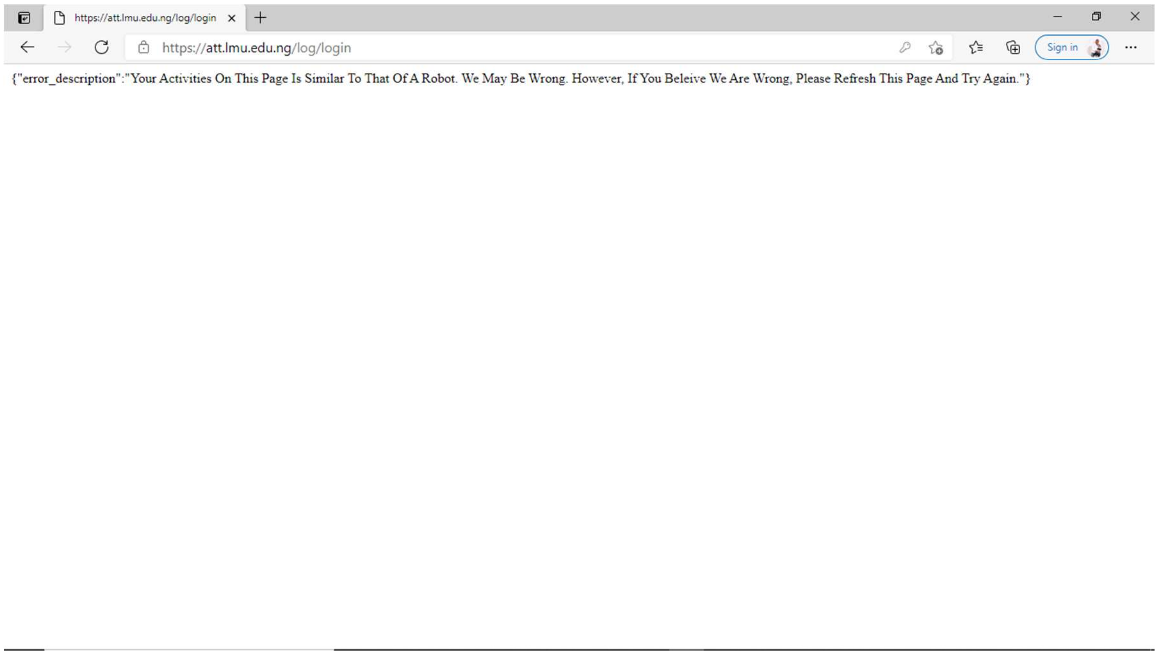


Figure 3.38a: A student trying to login to the platform without taking the CAPTCHA test

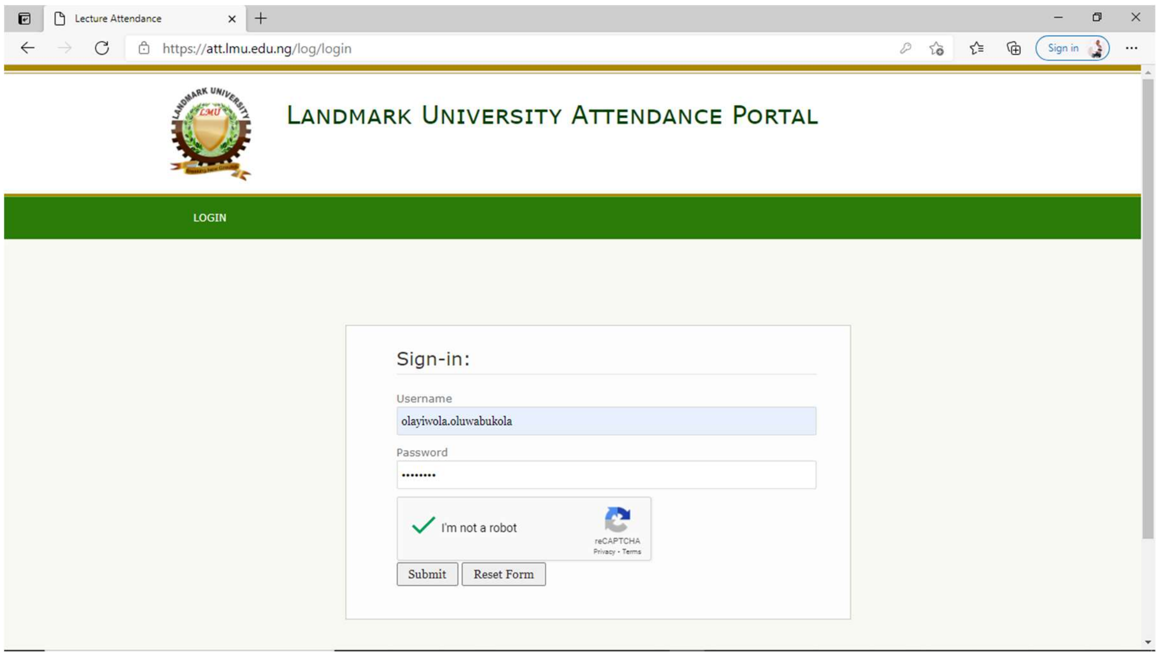

https://att.lmu.edu.ng/log/login
https://att.lmu.edu.ng/log/login
Sign in

```
{"error_description":"Your Activities On This Page Is Similar To That Of A Robot. We May Be Wrong. However, If You Beleive We Are Wrong, Please Refresh This Page And Try Again."}
```

Figure 3.38b: Error message returned without taking the CAPTCHA test

Next, a student successfully enters her credentials and passes the CAPTCHA test in Figure

3.39a and Figure 3.39b:


Lecture Attendance

https://att.lmu.edu.ng/log/login

LANDMARK UNIVERSITY ATTENDANCE PORTAL

LOGIN

Sign-in:

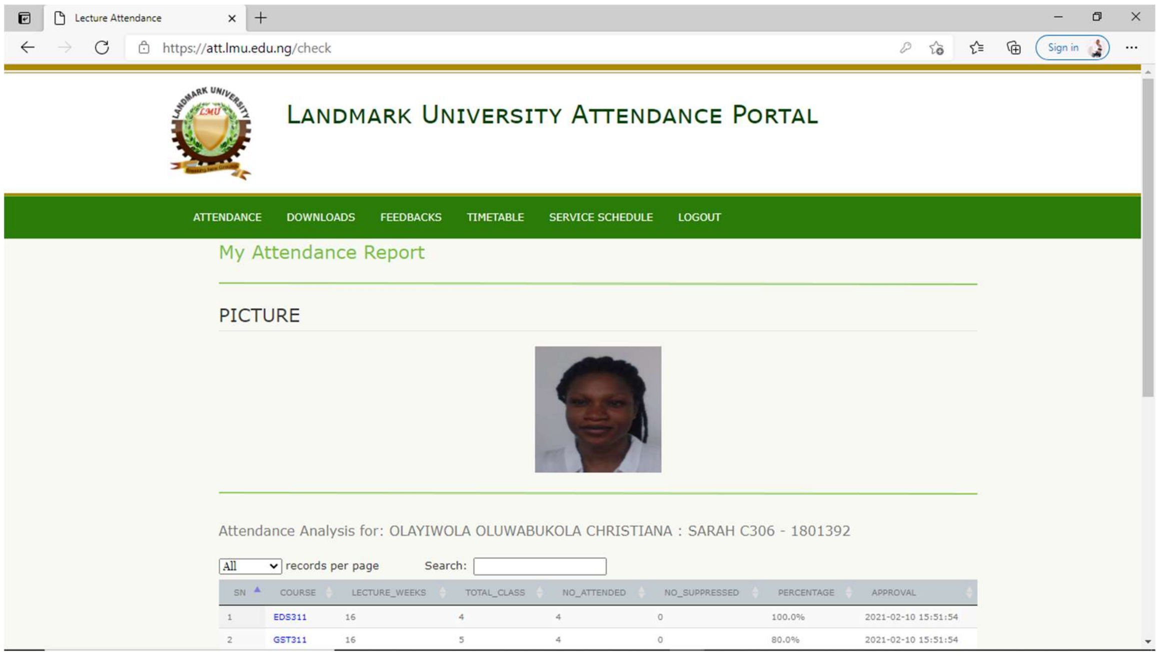
Username: olayiwola.oluwabukola

Password: ••••••••

I'm not a robot

Submit | Reset Form

Figure 3.39a: Correct parameters and successfully taking the CAPTCHA test


Lecture Attendance

https://att.lmu.edu.ng/check

LANDMARK UNIVERSITY ATTENDANCE PORTAL

ATTENDANCE | DOWNLOADS | FEEDBACKS | TIMETABLE | SERVICE SCHEDULE | LOGOUT

My Attendance Report

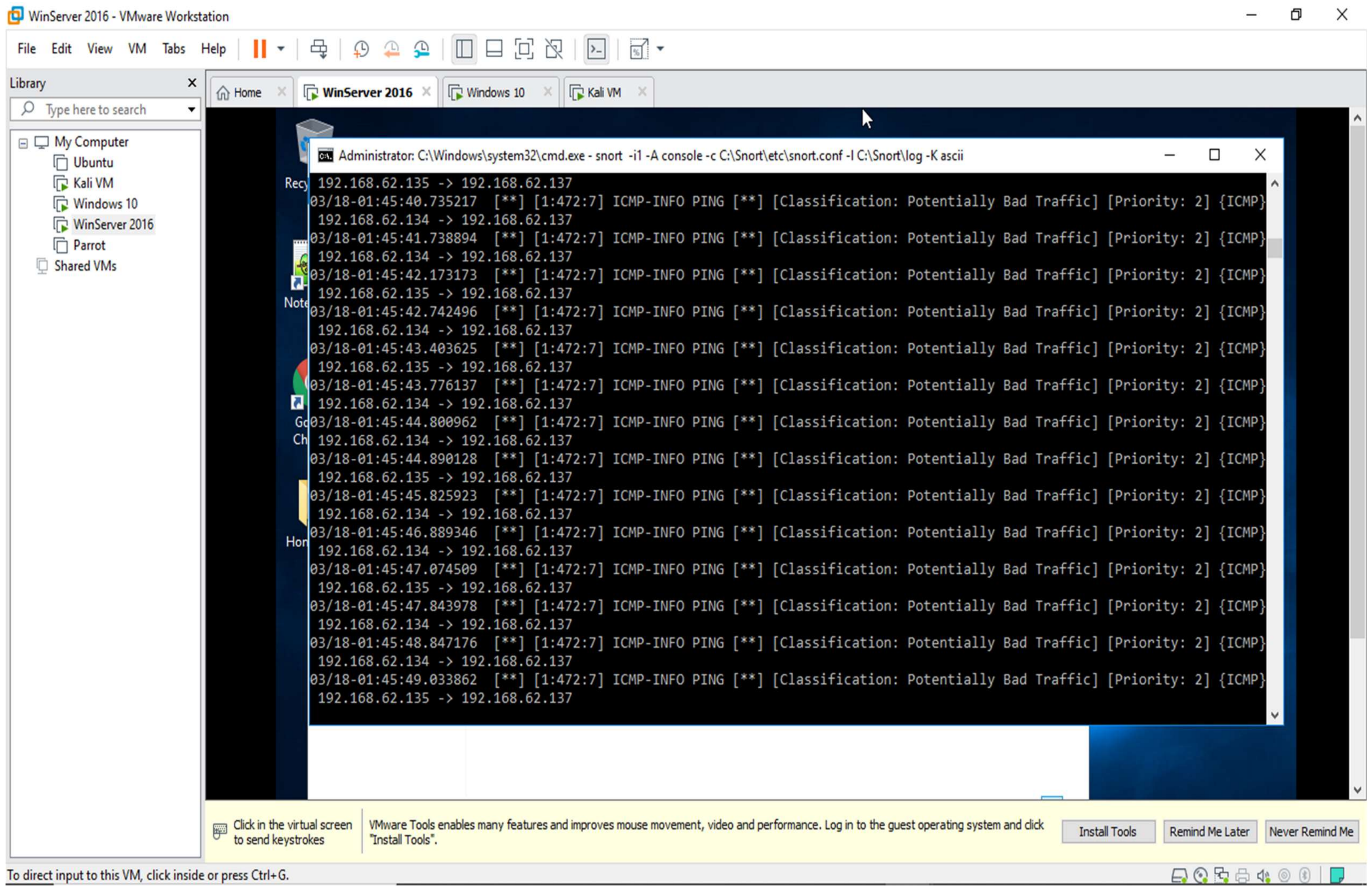
PICTURE

Attendance Analysis for: OLAYIWOLA OLUWABUKOLA CHRISTIANA : SARAH C306 - 1801392

All records per page   Search:

SN	COURSE	LECTURE_WEEKS	TOTAL_CLASS	NO_ATTENDED	NO_SUPPRESSED	PERCENTAGE	APPROVAL
1	EDS311	16	4	4	0	100.0%	2021-02-10 15:51:54
2	GST311	16	5	4	0	80.0%	2021-02-10 15:51:54

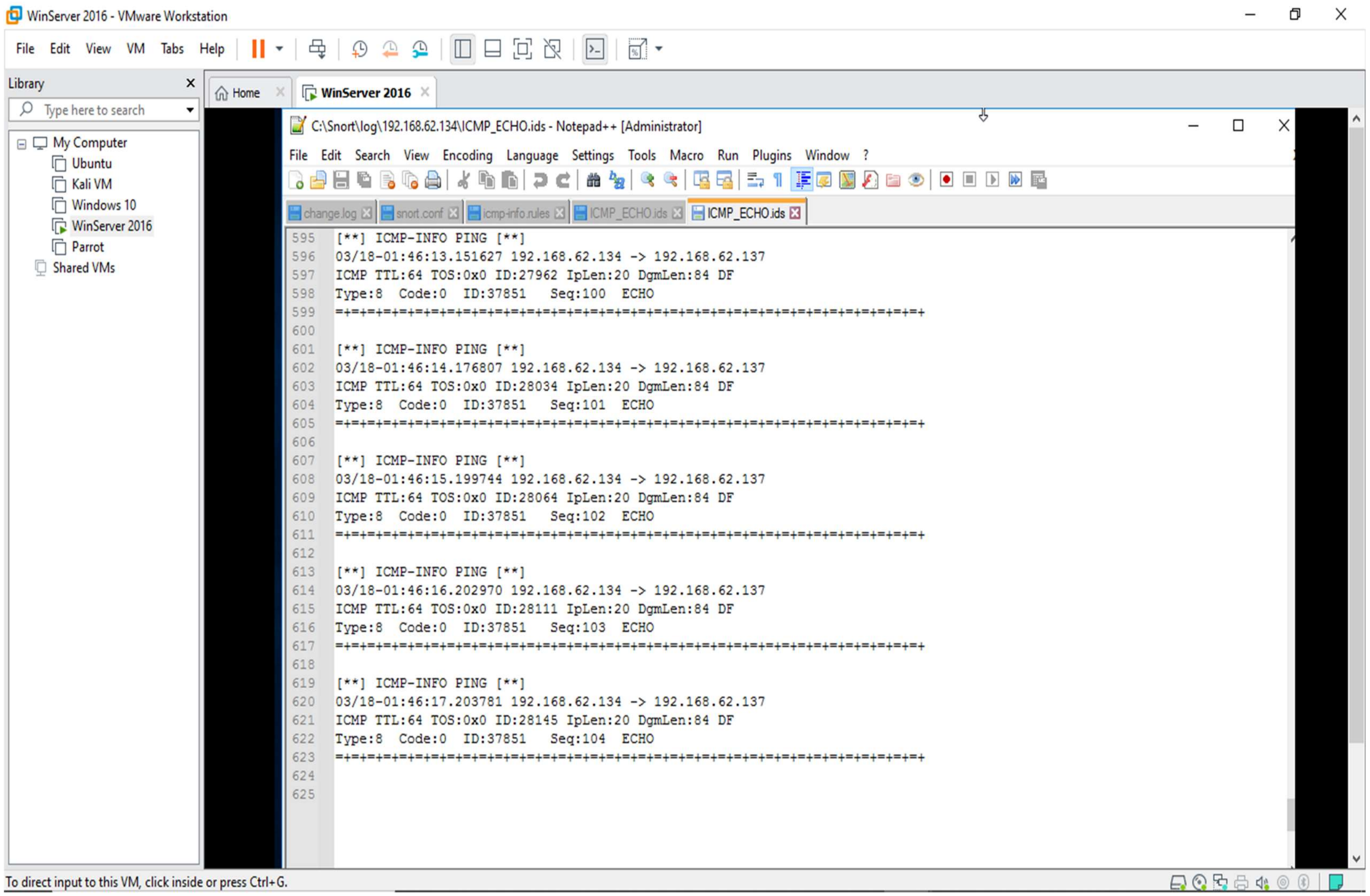
Figure 3.39b: Successful login after taking the CAPTCHA test


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
Windows 10
Kali VM
Administrator: C:\Windows\system32\cmd.exe - snort -i1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii
Click in the virtual screen to send keystrokes
Install Tools
Remind Me Later
Never Remind Me
To direct input to this VM, click inside or press Ctrl+G.

```
 192.168.62.135 -> 192.168.62.137
03/18-01:45:40.735217  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:41.738894  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:42.173173  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/18-01:45:42.742496  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:43.403625  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/18-01:45:43.776137  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:44.800962  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:44.890128  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/18-01:45:45.825923  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:46.889346  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:47.074509  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
03/18-01:45:47.843978  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:48.847176  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.134 -> 192.168.62.137
03/18-01:45:49.033862  [**] [1:472:7] ICMP-INFO PING [**] [Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
 192.168.62.135 -> 192.168.62.137
```

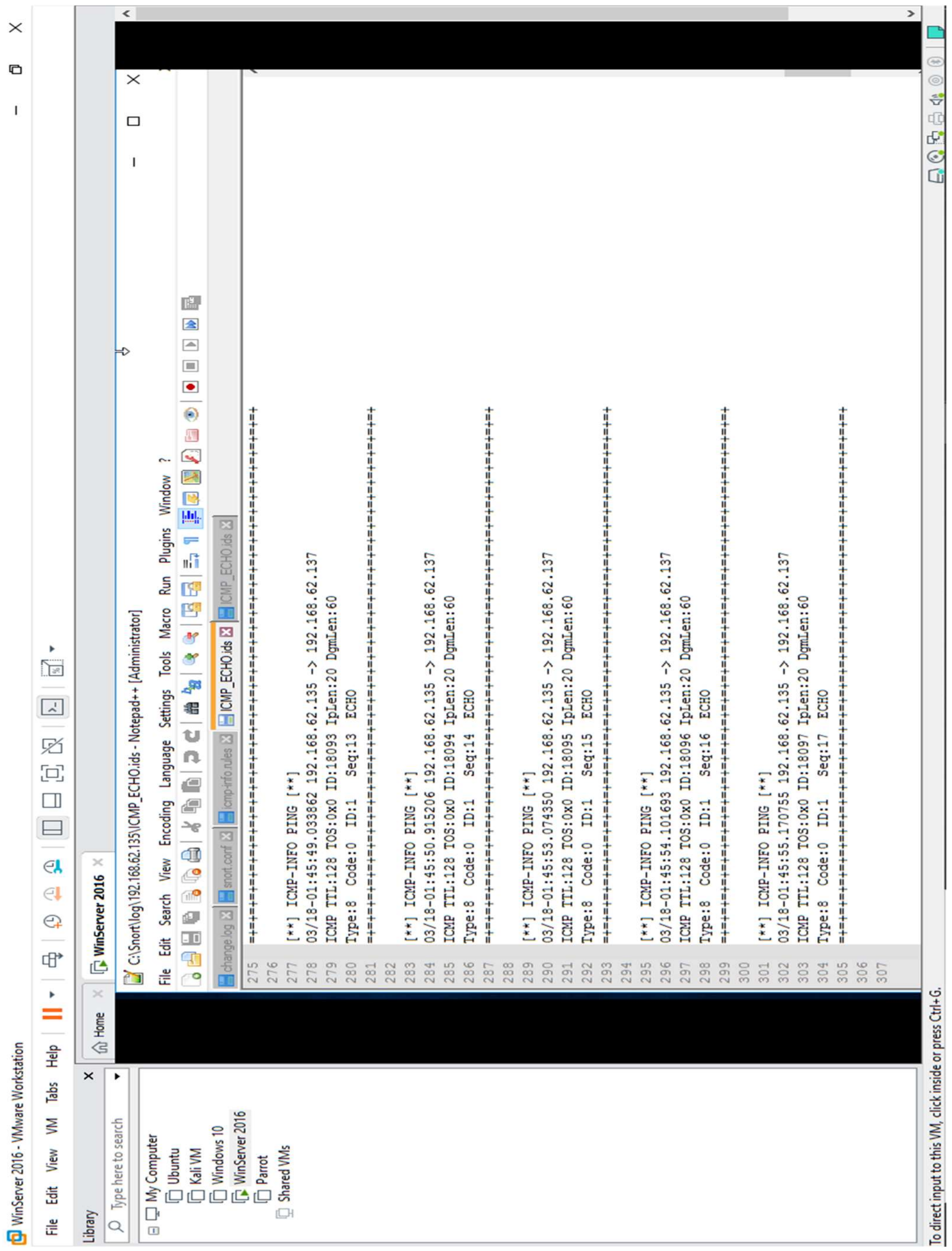
VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".

Figure 3.40: Traffic captured on our server machine using our deployed Honeypot


WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
C:\Snort\log\192.168.62.134\ICMP_ECHO.ids - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
snort.conf
icmp-info.rules
ICMP_ECHO.ids
ICMP_ECHO.ids
To direct input to this VM, click inside or press Ctrl+G.

```
[**] ICMP-INFO PING [**]
03/18-01:46:13.151627 192.168.62.134 -> 192.168.62.137
ICMP TTL:64 TOS:0x0 ID:27962 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:37851   Seq:100  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:46:14.176807 192.168.62.134 -> 192.168.62.137
ICMP TTL:64 TOS:0x0 ID:28034 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:37851   Seq:101  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:46:15.199744 192.168.62.134 -> 192.168.62.137
ICMP TTL:64 TOS:0x0 ID:28064 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:37851   Seq:102  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:46:16.202970 192.168.62.134 -> 192.168.62.137
ICMP TTL:64 TOS:0x0 ID:28111 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:37851   Seq:103  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:46:17.203781 192.168.62.134 -> 192.168.62.137
ICMP TTL:64 TOS:0x0 ID:28145 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:37851   Seq:104  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 3.41: Log file of the captured packets from the Kali Linux client machine



```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:45:49.033862 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:18093 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:13  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:45:50.915206 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:18094 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:14  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:45:53.074350 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:18095 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:15  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:45:54.101693 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:18096 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:16  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP-INFO PING [**]
03/18-01:45:55.170755 192.168.62.135 -> 192.168.62.137
ICMP TTL:128 TOS:0x0 ID:18097 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:17  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

WinServer 2016 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Kali VM
Windows 10
WinServer 2016
Parrot
Shared VMs
Home
WinServer 2016
C:\Snort\log\192.168.62.135\ICMP_ECHO.ids - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
snort.conf
icmp-info.rules
ICMP_ECHO.ids
ICMP_ECHO.ids
To direct input to this VM, click inside or press Ctrl+G.

Figure 3.42: Log file of the captured packets from the Windows client machine

Activity is captured as depicted in Figures 3.40, 3.41 and 3.42 respectively.

One of the bot/malicious program detected by our technique is found below while others can be found in Appendix D.

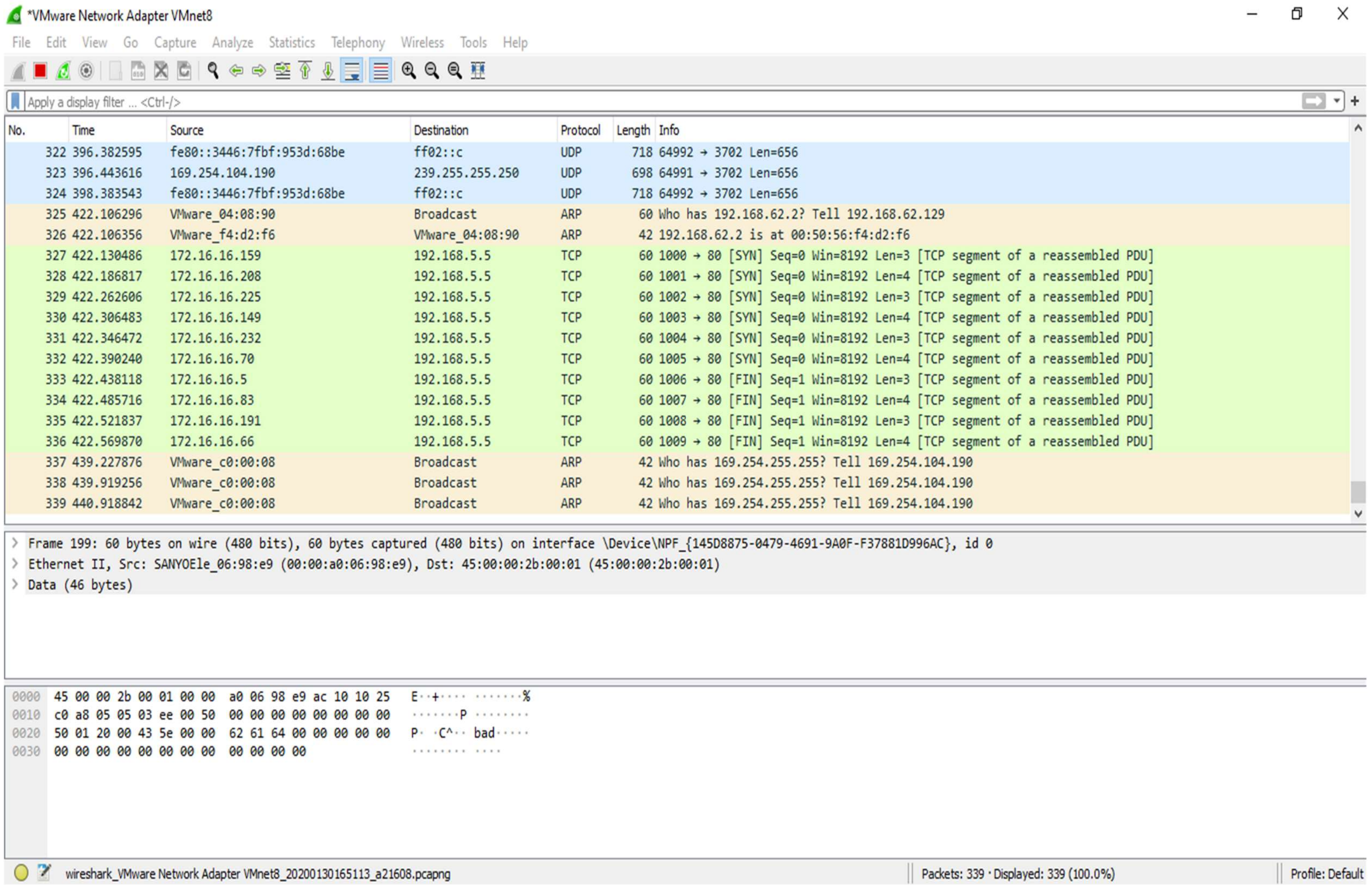
CHAPTER FOUR

RESULTS AND DISCUSSION

This chapter presents the results of the implementation performed and its evaluation results. Also, the chapter further provides a detailed discussion of the results and findings of the proposed framework. The result of the evaluation proves the justification for the performance of this study aligning with the aim and objectives of the study.

4.1. Results

Jess Rules, implementing Rete algorithm and Diffie-Hellman key exchange algorithms were implemented in the hybrid technique, thereafter, performance evaluation metrics were performed. Specifically, this section presents the results of the studies for the proposed framework. Application and comparison of the existing techniques and the proposed methods is performed, using Think-Aloud Activity, with Thematic Analysis, this study suggest that the proposed hybrid technique performs competitively compared with the other existing methods, the effectiveness, efficiency and accuracy established during the usability testing was found to be 93% accurate, outperforming the existing single and hybrid frameworks. Figure 3.40 shows the hybrid simulation for the implementation capturing a bot attack while Figure 4.1 shows how data packets received were analyzed using Wireshark.



*VMware Network Adapter VMnet8

```
No.  Time        Source                      Destination      Protocol Length Info
 322 396.382595  fe80::3446:7fbf:953d:68be   ff02::c          UDP      718 64992 → 3702 Len=656
 323 396.443616  169.254.104.190             239.255.255.250  UDP      698 64991 → 3702 Len=656
 324 398.383543  fe80::3446:7fbf:953d:68be   ff02::c          UDP      718 64992 → 3702 Len=656
 325 422.106296  VMware_04:08:90             Broadcast        ARP       60 Who has 192.168.62.2? Tell 192.168.62.129
 326 422.106356  VMware_f4:d2:f6             VMware_04:08:90  ARP       42 192.168.62.2 is at 00:50:56:f4:d2:f6
 327 422.130486  172.16.16.159               192.168.5.5      TCP       60 1000 → 80 [SYN] Seq=0 Win=8192 Len=3 [TCP segment of a reassembled PDU]
 328 422.186817  172.16.16.208               192.168.5.5      TCP       60 1001 → 80 [SYN] Seq=0 Win=8192 Len=4 [TCP segment of a reassembled PDU]
 329 422.262606  172.16.16.225               192.168.5.5      TCP       60 1002 → 80 [SYN] Seq=0 Win=8192 Len=3 [TCP segment of a reassembled PDU]
 330 422.306483  172.16.16.149               192.168.5.5      TCP       60 1003 → 80 [SYN] Seq=0 Win=8192 Len=4 [TCP segment of a reassembled PDU]
 331 422.346472  172.16.16.232               192.168.5.5      TCP       60 1004 → 80 [SYN] Seq=0 Win=8192 Len=3 [TCP segment of a reassembled PDU]
 332 422.390240  172.16.16.70                192.168.5.5      TCP       60 1005 → 80 [SYN] Seq=0 Win=8192 Len=4 [TCP segment of a reassembled PDU]
 333 422.438118  172.16.16.5                 192.168.5.5      TCP       60 1006 → 80 [FIN] Seq=1 Win=8192 Len=3 [TCP segment of a reassembled PDU]
 334 422.485716  172.16.16.83                192.168.5.5      TCP       60 1007 → 80 [FIN] Seq=1 Win=8192 Len=4 [TCP segment of a reassembled PDU]
 335 422.521837  172.16.16.191               192.168.5.5      TCP       60 1008 → 80 [FIN] Seq=1 Win=8192 Len=3 [TCP segment of a reassembled PDU]
 336 422.569870  172.16.16.66                192.168.5.5      TCP       60 1009 → 80 [FIN] Seq=1 Win=8192 Len=4 [TCP segment of a reassembled PDU]
 337 439.227876  VMware_c0:00:08             Broadcast        ARP       42 Who has 169.254.255.255? Tell 169.254.104.190
 338 439.919256  VMware_c0:00:08             Broadcast        ARP       42 Who has 169.254.255.255? Tell 169.254.104.190
 339 440.918842  VMware_c0:00:08             Broadcast        ARP       42 Who has 169.254.255.255? Tell 169.254.104.190
```

```
> Frame 199: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{145D8875-0479-4691-9A0F-F37881D996AC}, id 0
> Ethernet II, Src: SANYOEle_06:98:e9 (00:00:a0:06:98:e9), Dst: 45:00:00:2b:00:01 (45:00:00:2b:00:01)
> Data (46 bytes)
```

```
0000  45 00 00 2b 00 01 00 00  a0 06 98 e9 ac 10 10 25   E··+···· ·······%
0010  c0 a8 05 05 03 ee 00 50  00 00 00 00 00 00 00 00   ·······P ········
0020  50 01 20 00 43 5e 00 00  62 61 64 00 00 00 00 00   P· ·C^·· bad·····
0030  00 00 00 00 00 00 00 00  00 00 00 00               ········ ····
```

wireshark_VMware Network Adapter VMnet8_20200130165113_a21608.pcapng | Packets: 339 · Displayed: 339 (100.0%) | Profile: Default

Figure 4.1: Default Wireshark Interface

4.2 Simulation Response Time Results

In validating the response time required for the hybrid technique, certain parameters were put in place as shown in the Tables 4.1 and 4.2:

Table 4.1: Simulation parameters for computing the response time

SN	METRIC	DESCRIPTION	EXECUTION TIME
1	Top-k (k)	Computes the number of times where the correct label is among the top k labels predicted (ranked by predicted scores)	120
2	Number of QoS attributes (*q*)	Measures service quality along two distinct dimensions: **reliability** and **responsiveness**	2
3	Number of Trials	Progression on trails	15,43,72,120,145
4	Priority Weight (*w*)	A function of the number of QoS attributes	1/q (corresponding to [0.5,0.5])
5	Number of successful trial runs (*t*)	Indicating how many times the simulation experiment was run	145

Data was statistically analyzed using Linear Regression in the statistical package called SPSS.

Table 4.2: Linear Regression analysis of the response time using SPSS

#Trials	Range	Min (s)	Max (s)	Mean (s)	Std. Deviation
15	79	312.00	391.00	336.87	19.8
43	87	312.00	399.00	340.27	23.98
72	126	312.00	438.00	342.8	27.86
120	94	312.00	406.00	344.63	22.83
145	78	328.00	406.00	349.43	19.4

The performance in terms of execution time of our Hybrid technique in producing a robust and secure web application was found to scale linearly with increase in number of service alternative (the more trail experiment carried out, the more the R^2 value tends towards full relativity.) as shown in Figure 4.2.

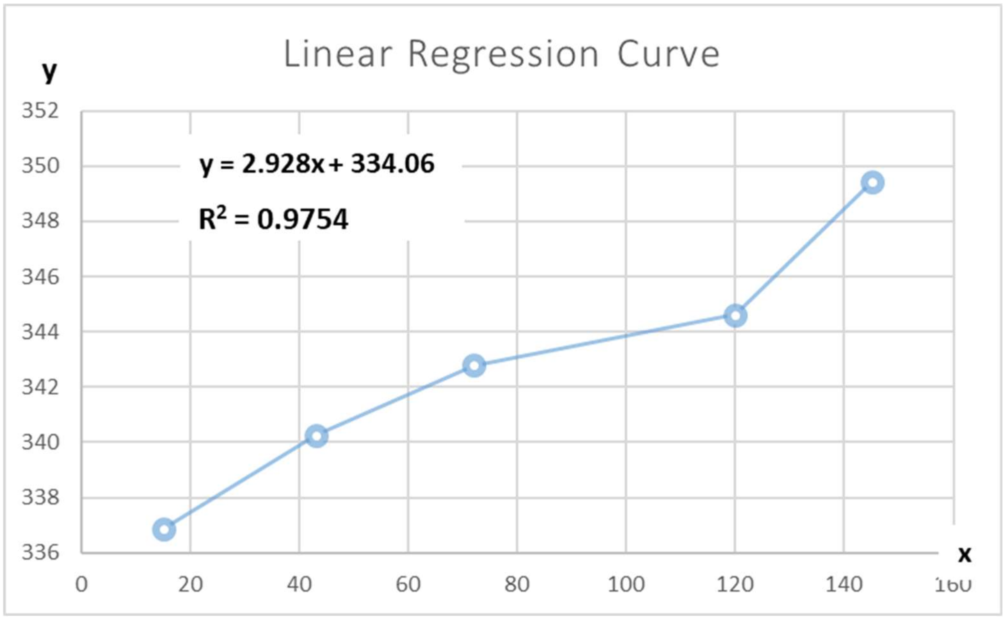


Figure 4.2: Linear Regression Curve

The adjusted R^2 value from the regression analysis is $\boldsymbol{R^2}$=**0.98** while the p-value (p=0.002) is less than the alpha value ($p < 0.05$).

Therefore our hybridized system is timely efficient and linearly scalable

4.3 System Usability Evaluation Using ISO 9241

The system usability evaluation was carried out in measuring user satisfaction, effectiveness and efficiency. Effectiveness succinctly expresses precision and completeness with which users can reach stated goals, while efficiency expresses the effort and resources required in order to reach goal attainment. User's satisfaction defines as user's attitude towards using a system.

In carrying out the usability evaluation, questionnaires were designed for users. The users in this case were crackers in the cyber-world i.e. those who are familiar with IDS/IPS and indeed HIPs. The designed questionnaire had five sections, Background information of respondents, User satisfaction, Effectiveness, Efficiency and Learnability of the system. The questions were administered via a five-point rating scale where 1 = Strongly Disagree, 2 = Disagree, 3 = Undecided, 4 = Agree and 5 = Strongly Agree. A total of 29 questions were administered and all responses were recorded and analyzed. The result of the one hundred and forty-five (145) respondents to the functionality of the system are analyzed in Table 4.3 and other details of the Questionnaire are given in Appendix E.

Table 4.3: Descriptive Statistics of Respondents

SN	VARIABLES	DESCRIPTION	RESPONDENTS
1	Gender	Male: **101** Female: **44**	145
2	Age Group	Below 20: **16** 21-30yrs: **81** 31-40yrs: **23** 41 and Above**: 15**	145
3	Cybersecurity Experience	Below 5yrs: **67** Above 5yrs: **78**	145

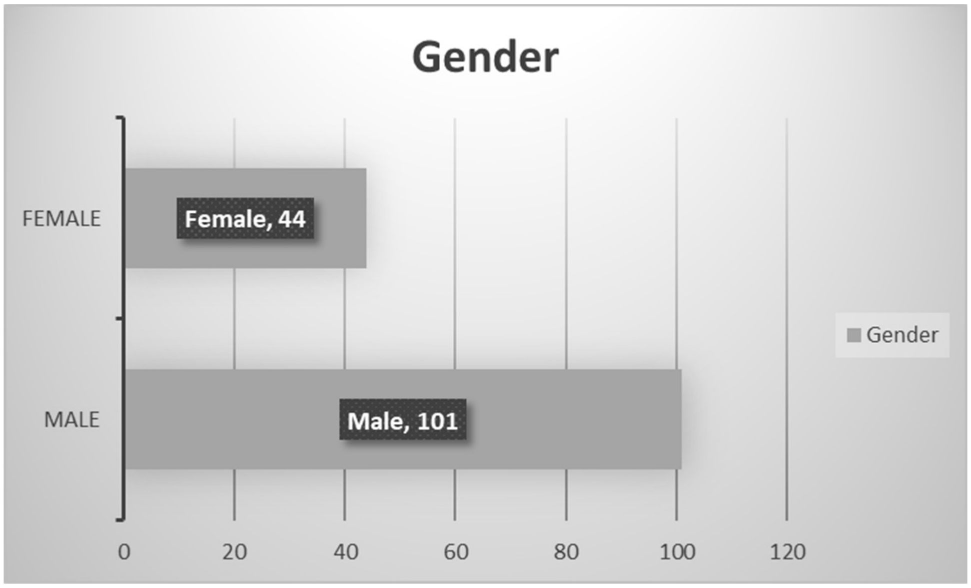


Figure 4.3: Gender description of respondents

From the 145 respondents, 101 of them were male while 44 of them were female as depicted in Figure 4.3.

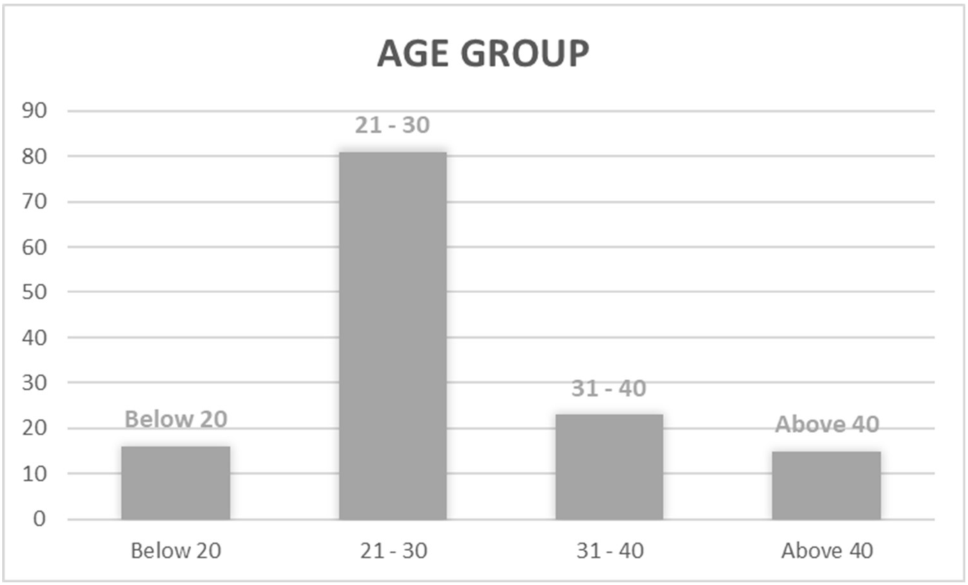


Figure 4.4: Age-group description of respondents

From the 145 respondents, 16 of them were below 20years, 81 of the respondents representing about 59% of the entire population were between ages 21 and 30years, 23 of the respondents were between 31 and 40 years while 15 of the respondents were above 40 years of age as depicted in Figure 4.4.

Also, our experiment looked at the cybersecurity experience of the respondents and it was found that 67 of the respondents had below 5years experience while 78 respondents, representing 58% had over 5years of cybersecurity experience.

The experience of the respondents becomes an important metric in the evaluation because it was important to situate the familiarity of our respondents on pertinent security issues with web application security, as this helps to further evaluate their understanding of the proposed solution. Figure 4.5 shows the distribution of their experience range.

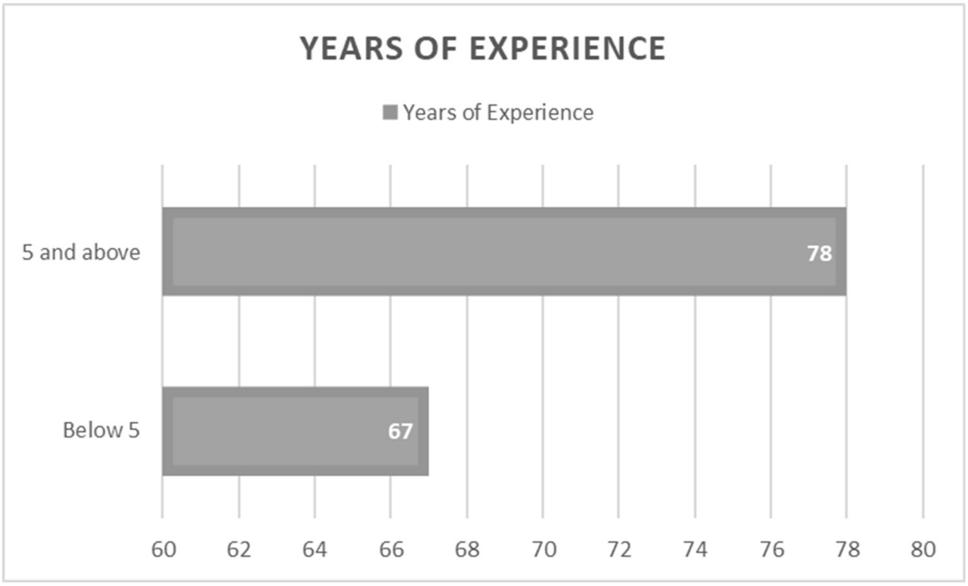


Figure 4.5: Years of Experience of respondents

4.3.1 Effectiveness of the proposed Model

Measures of Effectiveness (MOE) corresponds to the measures put in place to discover the accomplishment of mission objectives and the achievement of desired results. It helped in quantifying the results to be obtained by the system and may be expressed as probabilities that the system will perform as required.

Some questions were put to the user of the system to find out their basis for the performance of the system as it relates to system effectiveness. Table 4.4 and Figure 4.6 gives a representation of user responses.

Table 4.4: Measure of Effectiveness of Hybrid System

CATEGORY Q_{i-n}	**% RESPONSE**
Strongly Disagree	3
Disagree	6
Undecided	12
Agree	25
Strongly Agree	55

where Qi…Qn represents question categories

This MOE helped in assessing behavioural changes in the system, its capability, or operational environment which is tied to measuring the attainment of the achievement of the set out objective.

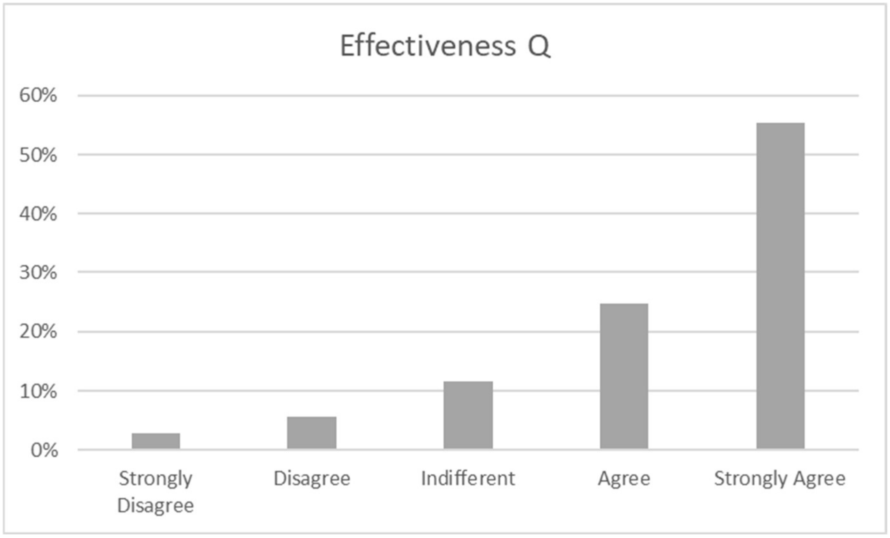


Figure 4.6: System Effectiveness of the proposed Framework

4.3.2 Efficiency of the proposed Framework

A system has to be evaluated for effectiveness and efficiency for the highest utility to the system user. In simpler terms, the effectiveness is a measure of the goodness of the output, while the efficiency focuses on measuring the system productivity, that is, the measure of the output against the input.

Questions were put to the user of the system to find out their basis for the productivity of the system as it relates to system efficiency. Table 4.5 and Figure 4.7 gives a representation of user responses.

Table 4.5: Measure of Efficiency of Hybrid System

CATEGORY Q_{i-n}	**% RESPONSE**
Strongly Disagree	3.26

Disagree	3.72
Undecided	16.28
Agree	22.33
Strongly Agree	54.42

where Qi…Qn represents question categories

The measure of the efficiency defines the ratio of useful output to total input, usually expressed with the mathematical formula r=P/C.

P indicates the amount of valid output ("product") produced per the amount C ("cost") of utilized resources.

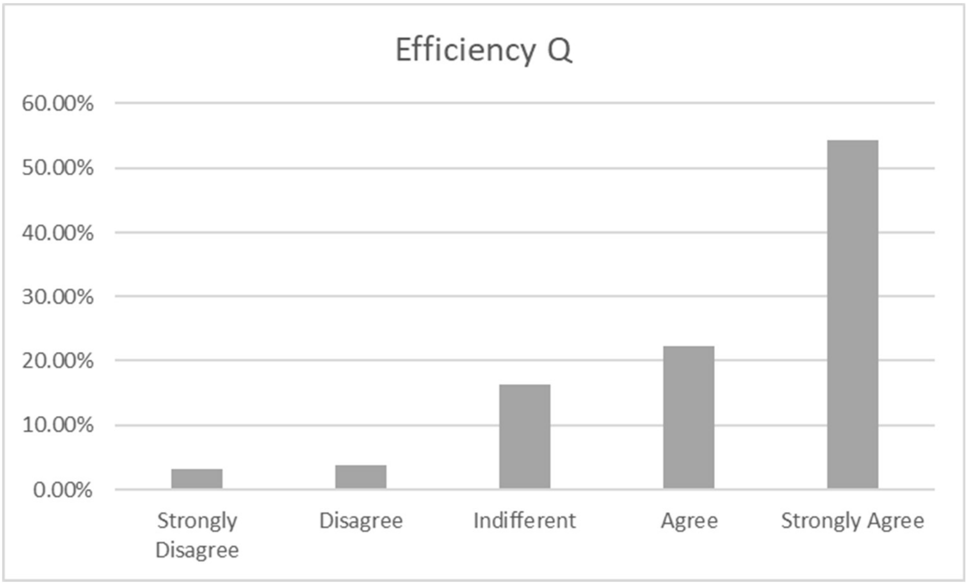


Figure 4.7: System Efficiency of the proposed model

4.3.3 User Satisfaction of the proposed Model

Establishing a form of connection with the system user is a valid way to measure user satisfaction. One of the most common methods of measuring user satisfaction was employed where respondents recorded their feedback via multiple-choice questions, rating questions, open-ended questions.

Questions were put to the user of the system to find out the measure of their satisfaction with system usage. Table 4.6 and Figure 4.8 gives a representation of user responses.

Table 4.6: Measure of User Satisfaction of Hybrid System

CATEGORY Q_{i-n}	**% RESPONSE**
Strongly Disagree	3.72
Disagree	8.84
Undecided	14.42
Agree	29.30
Strongly Agree	43.72

where Qi…Qn represents question categories

There is an agitation that User Satisfaction has a theoretical support for attributing attitudes (i.e., satisfaction) and behaviour in psychology. Inadvertently, evidence points to increasing employment of User Satisfaction questionnaires as a measure of system effectiveness.

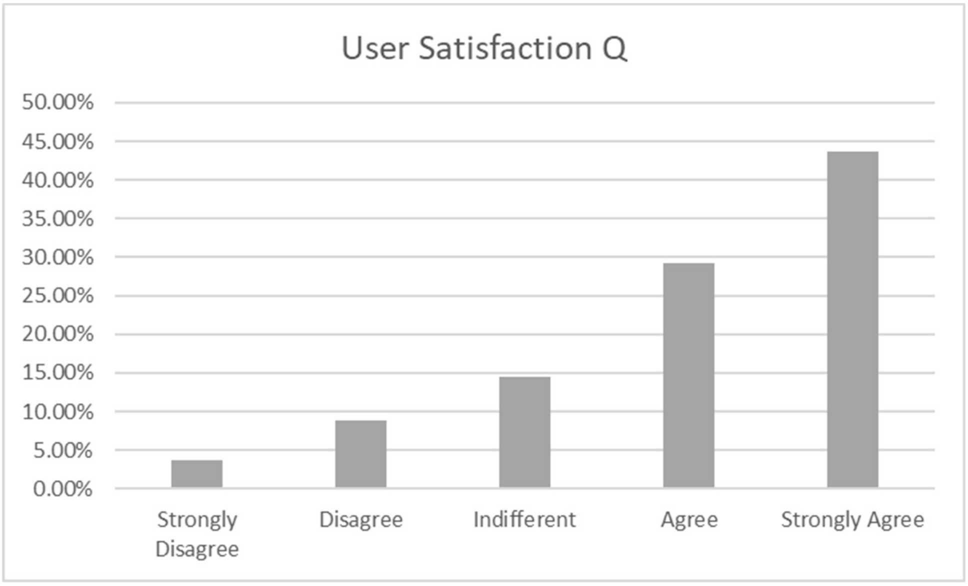


Figure 4.8: User Satisfaction of the proposed model

4.3.4 Learnability of the proposed Model

Learnability defines the ease with which the proposed framework in the developed application can be used and understood by users. The better the learnability of our proposed framework, the less training needed and the lesser the response time it will take for a person to use it.

Hence, questions were tailored to find out the basis for the time of completion of the system usage as it relates to system learnability. Table 4.7 and Figure 4.9 gives a representation of user responses.

Table 4.7: Measure of Learnability of Hybrid System

CATEGORY Q_{i-n}	**% RESPONSE**
Strongly Disagree	58.14
Disagree	69.77
Undecided	65.11
Agree	79.07
Strongly Agree	69.77

where Qi…Qn represents question categories

Principles behind the learnability of the system are concerned with the interactive system features, which helps unlearned users to learn quickly and also allows steady progression to perfection.

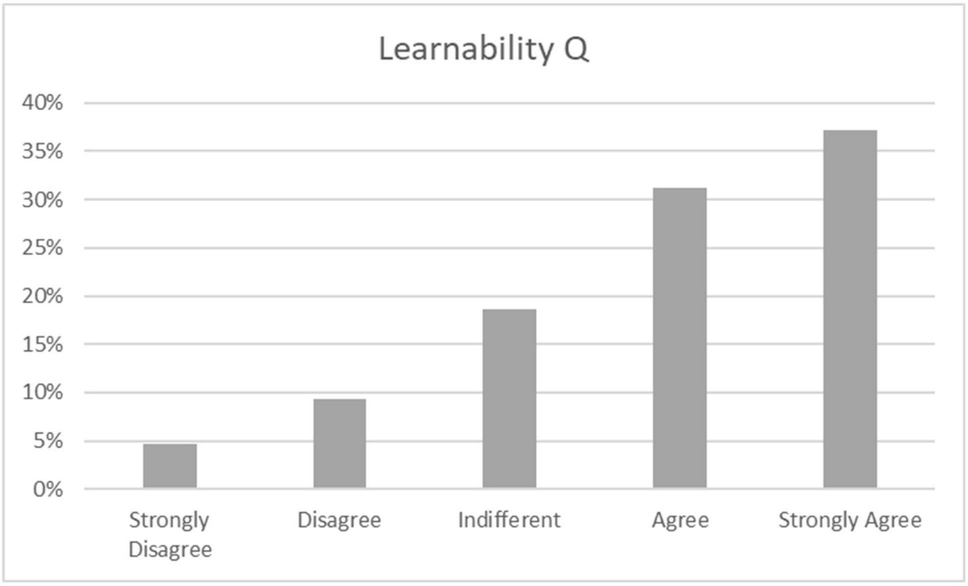


Figure 4.9: Learnability of the proposed Model

Table 4.8 below describes the responses of the participants in the survey for each of the question categories and subcategories. The range of the categories stretches from Effectiveness to Efficiency, to User Satisfaction and then Learnability.

Table 4.8: Usability Evaluation Report Summary

		Strongly Disagree	**Disagree**	**Indifferent**	**Agree**	**Strongly Agree**
EFFECTIVENESS	Q1	4.65%	9.30%	6.98%	9.30%	69.77%
	Q2	2.33%	6.98%	9.30%	39.53%	41.86%
	Q3	0	0	27.91%	30.23%	41.86%
	Q4	2.33%	4.65%	2.33%	20.93%	69.77%
	Q5	4.65%	6.98%	11.63%	23.26%	53.49%
EFFICIENCY	Q1	0	0	4.65%	2.33%	93.02%
	Q2	0	4.65%	6.98%	34.88%	53.49%
	Q3	2.33%	4.65%	23.26%	41.86%	27.91%
	Q4	0	0	18.60%	9.30%	72.09%
	Q5	13.95%	9.30%	27.91%	23.26%	25.58%

SATISFACTION	Q1	0	18.60%	6.98%	9.30%	65.12%
	Q2	2.33%	9.30%	6.98%	51.16%	30.23%
	Q3	4.65%	4.65%	25.58%	41.86%	23.26%
	Q4	9.30%	6.98%	16.28%	32.56%	34.88%
	Q5	2.33%	4.65%	16.28%	11.63%	65.12%
LEARNABILITY	Q1	9.30%	23.26%	13.95%	27.91%	30.23%
	Q2	4.65%	9.30%	16.28%	27.91%	41.86%
	Q3	4.65%	2.33%	27.91%	34.88%	30.23%
	Q4	0	9.30%	11.63%	44.19%	34.88%
	Q5	4.65%	2.33%	23.26%	20.93%	48.84%

4.3.5 Think-Aloud Activity and Thematic Analysis Evaluation

In carrying out the usability evaluation, a clear theme with the first five (5) users was established, otherwise, we sent a mail to another group of students who have volunteered for the scheme and have had experience working on HIPs in the cyber-world as both pentesters, crackers and so on. in order to hear them voice out their opinion, see and observe themes amongst them as they use the hybrid design. The methods put into place were essentially ***Think-Aloud activity*** and ***Thematic Analysis***.

Table 4.9: Themes creation Report

SN	Think-Aloud	Themes
1	Picture Selection	CAPTCHA Puzzle Issues
2	Merging IDS	IDS/IPS
3	Frontend security	Internet connectivity
4	Backend security	Server (Honeypot) Issues
5	User authentication	Link-correlation
6	Hybridization	Very much a possibility
7	User details incorporation in CAPTCHA	Feasible but not encouraged

The Think-Aloud Activity helped in building a strategy that converts thoughts to words for absolute comprehension and in-depth knowledge. This activity was conducted for the following reasons:

i. To understand what they know about the IDS and IPS schemes
ii. To see what they envisage as an advantage for the hybrid scheme
iii. To see if they understand the design of the hybrid scheme
iv. To see if they have adequate knowledge about the hybrid scheme as they iterate through
v. To see if there are ways they could have a better understanding of the design
vi. To point out the most important aspect of the hybrid scheme
vii. To see how the design fits into what they already know about IDS and IPS.

On the other hand, Thematic Analysis helps with the identification of patterns or themes within qualitative data. Hence this method showcased ways of identification of common trends or patterns amongst the participants at each and every stage of the usability test of the hybrid design. The analysis revolves around the following reasons:

i. Examining collated data to identify significant themes.
ii. Collating data relevant to each participant's pattern.
iii. Work with the data
iv. Review the viability of each participant's theme.

Having explicitly looked at some of the performance metrics of the hybrid system utilizing the Diffie-Hellman key exchange algorithm, Jess Rules implementing the Rete algorithm and then the Hidden Markov Models, the summary of the evaluation report is presented in Table 4.10.

Table 4.10: Test results from Think-Aloud Activity

SN	CATEGORY	OBSERVATIONS
1	Picture Selection	**Multiple CAPTCHA solving**: Oftentimes, CAPTCHA keeps asking for re-authentication even after correctly selecting the images concerned. **Blurry Images**: Oftentimes, the images appear very blurry which makes it quite difficult for even human users to decipher. **CAPTCHA Inconsistency**: On some interfaces, the CAPTCHA appears as a single-click solution while on others, it requires a multi-click solution.
2	Merging IDS/IPS	**IDS**: It is not enough to detect intrusions, prevention should also be considered. **IPS**: Preventing an intrusion will not give UI/UX developers updated knowledge on current strategies engaged by hackers and system crackers.
3	Frontend Security	**Online Validation**: Validation should be done not just offline but online in order to utilize the knowledge base of the most recent attack strategies. Locally validating will absolve latest strategies. **User Authentication**: Both the user details authentication and CAPTCHA validation can go on side-by-side. One does not really have to wait until the completion of the other. Side-by-side authentication can suffice.
4	Backend Security	**Honeypot:** There should be a clear distinction between malicious traffic and pure network traffic. The Snort Honeypot setup should be able to filter such.
5	Link Correlation	**Authentication Sequence**: CAPTCHA is solved first, Honeypot analyzes the incoming traffic before the user details are then authenticated.
6	Hybridization	**Possibility:** From implementation, it is evident it is possible to hybridize an IDS and IPS to further strengthen the web application security. **Novel Idea**: While it is a good idea, more work still needs to be done to make the idea robust and efficient
7	User details CAPTCHA	**Integration:** While it is a brilliant idea, it may be a window towards further granting bots access to human intelligence as every solved CAPTCHA goes into a knowledge base **Security:** Security can be further breached when human details are integrated into CAPTCHAs.

Table 4.11: Test results from Thematic Analysis

SN	THEMES	PHASE OF HYBRID	EXAMPLE
1	Enthusiasm	Application Login	Interested in the development
2	Unpreparedness on scheme details	Hybrid Scheme Registration	Having no prior knowledge of the details of the hybrid scheme. Users didn't know what to expect.
3	Soliloquy on CAPTCHA use	Solving CAPTCHAs	Solving multiple CAPTCHAs.
4	Validation issues	Navigation	CAPTCHA validation is done online and not locally.
5	Server Authentication	Honeypot Validation	All traffic are being treated as malicious

As a result of the user feedback in Table 4.11, the following processes were addressed in the iteration on the initial design:

i. Making the CAPTCHA less dramatic. Just simple and easy to comprehend.

ii. The ability to decipher which HIP is active at every point in time.

iii. Making the CAPTCHA locally solved and verifiable

iv. Dynamic linking on user authentication and validation of IDS/IPS

4.4 Validation of Results

In this study, we have performed some experiments incorporating a hybrid IDS/IPS using Puzzle CAPTCHA and Honeypot, carried out as an hybrid security layer using HMM and Diffie-Hellman Key exchange algorithm for implementation, the output of the accuracy for the performance metrics of the hybrid approach is tabulated in Table 4.14 and in the process, outperforming some of the other approaches with 93% accuracy.

Table 4.12: Accuracy Metrics for the Hybridized Technique of the Study

SN	Hybridized IDS/IPS Approach	Accuracy (%)
1	DBSCAN	95
2	K-Means, K-NN, NAÏVE BAYES	98.43
3	K-means, SVM and Fuzzy NN	97.31
4	Hidden Markov Model (HMM) and Decision Trees	77.8
5	Clustering, KNN	91.7
6	K-means, Modified Optimum Path Forest (MOPF)	85.92
7	Tree-based subspace Clustering (TCLUS)	98
8	Hidden Markov Model, Diffie-Hellman (Proposed Model)	93

Previous hybridized models have produced varying levels of accuracy as highlighted in Table 4.12, some out-performing our model while others under-performed, when compared to our presented hybrid model.

4.5 Heuristic Evaluation of Hybrid Model

Heuristic Evaluation was chosen to examine the hybridized design under the following steps:

i. Nielsen's Norman Group rules on usability guidelines for Heuristic Evaluation was selected as appropriate heuristics to evaluate the hybridized design because of the range of evaluators considered (Hackers, Crackers and Researcher). It also allows users to be observed individually in a Think-Aloud and Thematic Analysis session and have the results compiled.

ii. In addition to the few available researchers, a major stakeholder (Mr. Folarin Olufemi – a Network Administrator in Landmark University) was recruited to perform an evaluation of the attendance portal activities with a focus on the hybridized design.

iii. A combination of the researchers and the stakeholder worked through the portal processes to see if heuristic criteria were met.

iv. Where a violation of heuristic was found, it was noted and the specific heuristic categories were identified. Should an error be discovered that did not match a heuristic criteria, it was worthy of note.

v. Upon completion, the hackers, researcher and stakeholder compared notes to form a comprehensive list.

Each item was critically reviewed and a severity rating marked for each that indicates the results of the discussion.

The schema for the severity rating is noted below.

a. 0 - don't agree that this is a usability problem

b. 1 - cosmetic problem

c. 2 - minor usability problem

d. 3 - major usability problem; important to fix

e. 4 - usability catastrophe; imperative to fix

4.5.1 Results of Heuristic Evaluation

The complete list of violations identified by all evaluators (hackers, the researcher and stakeholder) can be found in Tables 4.13, 4.14 and 4.15.

System users (in this case, Hackers and Crackers) rated the severity of some of the violations identified in the Themes created and the result of their ratings was grouped in table 4.13

Table 4.13: List of severity ratings by the Hackers and Crackers

SN	HEURISTIC CATEGORY	VIOLATION	SEVERITY
1	Picture Selection	CAPTCHA Puzzle Issues	2
2	Merging IDS	IDS/IPS	0
3	Frontend security	Internet connectivity	3
4	Backend security	Server (Honeypot) Issues	1
5	User authentication	Link-correlation	1
6	Hybridization	Very much a possibility	0
7	User details incorporation in CAPTCHA	Feasible but not encouraged	4

Furthermore, stakeholders (in this case, Network and Database Administrators) rated the severity of some of the violations identified in the Themes created and the result of their ratings was grouped in table 4.14

Table 4.14: List of severity ratings by the Stakeholder (Network Administrators)

SN	HEURISTIC CATEGORY	VIOLATION	SEVERITY
1	Picture Selection	CAPTCHA Puzzle Issues	0
2	Merging IDS	IDS/IPS	0
3	Frontend security	Internet connectivity	4
4	Backend security	Server (Honeypot) Issues	0
5	User authentication	Link-correlation	1
6	Hybridization	Very much a possibility	0
7	User details incorporation in CAPTCHA	Feasible but not encouraged	2

Finally, an overall (in this case, both the system users and stakeholders) rated the severity of some of the violations identified in the Themes created and the result of their ratings was grouped in table 4.15

Table 4.15: Overall list of severity ratings by both the Researcher and the Stakeholder

SN	HEURISTIC CATEGORY	VIOLATION	SEVERITY
1	Picture Selection	CAPTCHA Puzzle Issues	1
2	Merging IDS	IDS/IPS	0
3	Frontend security	Internet connectivity	3
4	Backend security	Server (Honeypot) Issues	1
5	User authentication	Link-correlation	1
6	Hybridization	Very much a possibility	0
7	User details incorporation in CAPTCHA	Feasible but not encouraged	3

4.6 Discussion on Results

In line with the Research questions as depicted in Table 4.16, hypothesis and objectives which was set out from inception has been experimentally actualized and answers to our research questions have been provided determining which hypothesis to adhere to, the null or the alternative.

Table 4.16: Research questions answered

SN	RESEARCH QUESTION	EXPERIMENTAL RESULT
1	Is it possible to improve Web Application security by hybridizing protocols to fend-off intrusions from bots?	There have been several hybridized IDS/IPS, and the hybrid of CAPTCHA and Honeypot was successfully implemented in a Web Application.
2	Would the proposed Web Application security solution outperform the existing web security solutions highlighted in literature?	The resulting solution appeared to be very secure and robust, and in the process produced an improved overall performance in web application security

Kaspersky in an article (Kaspersky Lab, 2015) implied cyber surveillance through data exfiltration is a major fear of every business or establishment. In this experiment, the transfer of the intended data to the attacker occurs after host infection bypass and more often than not, uses encrypted traffic. Botnets essentially facilitate Distributed Denial of Service (DDoS) attacks, irrespective of the fact that the attackers are hired or owned (Symantec, 2016). Considering this, botnet infections keeps increasing year-in-year-out, making high-speed organizational networks, For example a university network, the primary targets.

Hence, in detecting botnets, communication in data can be based on some features within the network, features such as the communication protocol: HTTP, IRC, P2P or methods utilizing social network Thomas and Nicol, (2010) and also, the possibility of evasion techniques, For example Fast-Flux (Nazario and Holz, 2008).

The hybrid model developed in this study outperformed some of the already existing models in terms of accuracy with HMM + Diffie-Hellman achieving 93% as shown in table 4.17.

Table 4.17: Comparative Table Showing Performance Measures of other Techniques

SN	Authour	Technique(s)	Perf (%)
1	Amiri et. al., (2011)	Modified mutual information-based feature selection algorithm (MMIFS)	88
2	Zhang et. al., (2012)	Weighed Symmetrical Uncertainty_Area Under Roc(WSU_AUC), Selection Robust Stable Features(SRSF)	87
3	Fahad, Tari, Khalil, Habib, and Alnuweiri, (2013)	Local Optimization Approach (LOA)	83

4	Fahad, Tari, Khalil, Almalawi, and Zomaya, (2014)	Global Optimization Algorithm(GOA)	92
5	Liu et. al, (2015)	Class-Oriented Feature Selection (COFS)	91
6	Bhuyan, Bhattacharyya, and Kalita (2016)	Mutual Information and Generalized Entropy FS (MIGE-FS)	85
7	Bostani and Sheikhan, (2017)	NSGA-II and classifier is GHSOM with probabilistic relabeling	80
8	Zhu, Liang, Chen, and Ming, (2017)	I-NSGA-III+GHSOM	85
9	Zhang et. al., (2012)	Weighed Symmetrical Uncertainty_Area Under Roc(WSU_AUC), Selection Robust Stable Features(SRSF)	87
10	Fahad, Tari, Khalil, Habib, and Alnuweiri, (2013)	Local Optimization Approach (LOA)	83
11	**Proposed Hybrid Model** (Current Work)	**HMM + Diffie-Hellman**	**93**

With HMM and Diffie-Hellman algorithms, a few experiments were carried out, where optimization is needed to obtain accurate results. This optimization-based method has been proven as very efficient for classification and can be a tool for Web Administrators or Web Developers to diagnose attacks and trace activities engaged by users over a network.

CHAPTER FIVE
SUMMARY, CONCLUSION AND RECOMMENDATION

5.1. Summary

This study has succinctly looked at the concept of hybridization or layering on Intrusion Detection Systems (IDS) and have combined both the CAPTCHA and the Honeypot techniques, layering them over each other. The hybrid Intrusion Detection System utilizes a combination of Hidden Markov Model (HMM) and a Diffie Helman key exchange algorithms to work extensively on a manually generated dataset from user input on a locally developed server (https://att.lmu.edu.ng). The Hidden Markov Model (HMM) was solely channeled into identification of possible attacks while the Deffie Hellman key exchange has concentrated on confirming only the true attacks. This system performs the identification in a much simpler way by reducing the complexity of working on an overhead server as compared to a conventional local server. It achieved a good accuracy. Thus, IDS hybridization and layering achieves a more efficient security as depicted in the developed system. This work shows that as much as Honeypots and CAPTCHAs are both strong independent Intrusion Detection Systems and as well achieve the security aim they are targeted at, yet, there have been recorded breaches overtime when these independent security strategy are implemented. Hence, it becomes pertinent to push the ideology of layering and hybridization of more than one Intrusion Detection System in alleviating modern security problems on web platforms.

5.2 Conclusion

This study has been able to present the achievement of all four objectives set out from the beginning as well as providing answers to the two (2) research questions created at the beginning of the study.

This study introduces a hybrid method to resolve the inherent problems in high data breaches in web applications. The hybridized IDS layer method guarantees positive responses of the relevant data. An optimized solution incorporating HMM and Diffie-Hellman algorithms were developed using a locally generated dataset. The results are unique, using Thematic and Think Aloud Analysis classifiers. This study proves that the conventional techniques of focusing on one particular Intrusion Detection System per time in solving web form challenges with bots and spam is not strong enough to repel multiple forms of attacks. It only solves one kind of attack mechanism. This has largely been overlooked in the previous literatures. Focus is expected to shift to a new and novel method for defending web forms and data protection that will help in effectively securing databases from structured and coordinated attacks. It is about time hybridization methods that are novel such as is presented in this study, are developed to provide a defense mechanism against coordinated attacks by hackers.

The developed method can become a solution to new problems and challenges present in web forms, application management and other data breach application management. The use and introduction of the hybridized approach for Intrusion Detection model is confined to all web induced interactive applications.

5.3 Contribution to Knowledge

This research work contributes to knowledge in the following ways:

i. Introduction of SSH protocol, Diffie-Hellman key-exchange algorithm, Hidden Markov Models and Jess rules into the integration of an adaptive CAPTCHA and Honeypot was found to be successful and 93% accurate.

ii. It also developed an in-house application on a locally designed server (https://att.lmu.edu.ng), which provides users within the community (both students and staff) with a pattern (or themes) in its usage over the server as they interact with web forms on the server.

5.4 Recommendation

A developed hybridized framework should focus on solving more than one attack type at every point in time. Attackers are not relenting when it comes to the security of web applications. When they launch an attack and it is not successful, they try other forms of attack just to see how penetration can be possible. Hence the need for Intrusion Prevention/Detection Systems to become more versatile and exercise additional security measures. Hybridization seems to be the way forward and the earlier it gains wide acceptability, the better for everyone.

Results from the hybridized solution produced an accuracy level that surpassed majority of the existing solutions and in the process, users of the system noticed little difference compared to the usage of other methods which validates that our method scales linearly in usability and response time.

This hybrid solution is therefore recommended to web application developers for effective defense against bot attacks and human-solvers of CAPTCHA while also further strengthening the security of web applications and transactions over web-forms.

5.5 Future Work

Security in web forms need to look beyond a singular attack and defense mechanism, hence, future works will seek to develop frameworks that efficiently identifies several other forms of hybridization in Intrusion Detection Systems thereby forming hybrid techniques in further strengthening the security of web applications.

Development of frameworks that effectively identifies the strengths, weaknesses as well as compatibility of individual Intrusion Detection Systems, layers them and validates the conceived and developed hybrid system with various datasets.

5.6 Limitations of Work

In carrying out this study, there were a few limitations encountered some of which are listed below:

i. Selection: Users of the system were carefully selected. They were those who are familiar with the latest IDS/IPS security technologies and have been users of the security systems for at least two (2) years.

ii. Instrumentation: Users of the system were tutored on the modalities of operation of the system for familiarity purposes.

iii. Platform: In this study, not all platforms of implementation are considered, the protocol was only tested on Web platform ruling out mobile app implementation.

iv. Requirements: Method requires the use of Internet Service

v. Predictability: You never really can predict the human mind

REFERENCES

Abdullah, W. K. (2016). A Survey of Current Research on CAPTCHA. I*nternational Journal of Computer Science and Engineering Survey*, *7*(3), 1–21.

Abinaya, R., Janani, S., and Devi, P. N. (2015). Anti-Phishing Image Captcha Validation Scheme using Visual Cryptography. *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET)*, *1*(2), 86–90.

Al-Fannah, N. M. (2018). Making Defeating CAPTCHAs Harder for Bots. *Proceedings of Computing Conference 2017*, *2018 - Janua*, 775–782. https://doi.org/10.1109/SAI.2017.8252183

Aljarah, I., and Ludwig, S. A. (2013). MapReduce Intrusion Detection System based on a Particle Swarm Optimization Clustering algorithm. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, (June), 955–962. https://doi.org/10.1109/CEC.2013.6557670

Almazyad, A. S., Ahmad, Y., and Kouchay, S. A. (2011). Multi-Modal CAPTCHA: A User Verification Scheme. *IEEE Symposium on Security and Privacy*, *11*.

Amiri, F., Yousefi, M. R., Lucas, C., Shakery, A., and Yazdani, N. (2011). Mutual in-formation-based feature selection for intrusion detection systems. Journal of Net-work and Computer Applications, 34(4), 1184-1199.

Anil, J., Naveli, G. S., and Bhukya, S. (2018). *Image Based CAPTCHA Generation System*. International Journal of Pure and Applied Mathematics *118*(24), 1–9.

Ezenwoke, A., and Igbekele, E. (2019). Cloud Computing Research in Nigeria: A Bibliometric and Content Analysis. *Asian Journal of Scientific Research*, *12*(1), 41–53.

Ba, Y. (2017). Understanding Cybercrime and Developing a Monitoring Device. (Bachelor's Thesis). Turku University of Applied Sciences, Turku, Finland, 2017.

Banday, M. T., and Shah, N. A. (2009). Image Flip CAPTCHA. *The ISC Int'l Journal of Information Security*, *8*, 13.

Banday, M. T., and Shah, N. A. (2011). *A Study of CAPTCHAs for Securing Web Services*. International Journal of Secure Digital Information Age, Vol. 1. No. 2, December 2009

Baykara, M., and Das, R. (2018). A novel honeypot based security approach for real-time intrusion detection and prevention systems. *Journal of Information Security and Applications*, *41*(August), 103–116. https://doi.org/10.1016/j.jisa.2018.06.004

Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2012). An effective unsupervised network anomaly detection method. *ACM International Conference Proceeding Series*, (March 2016), 533–539.

Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2016). A multi-step outlier-based anomaly detection approach to network-wide traffic. *Information Sciences*, *348*, 243–271.

Bohara, A., Thakore, U., and Sanders, W. H. (2016). Intrusion detection in enterprise systems *by combining and clustering diverse monitor data*. Symposium and Bootcamp on the Science of Security (April), 7–16.

Bostani, H., and Sheikhan, M. (2017). Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept. Pattern Recognition, 62, 56-72.

Bursztein, E., Bethard, S., Fabry, C., Mitchell, J. C., and Jurafsky, D. (2010). How Good

Are Humans At Solving Captchas. *IEEE Symposium on Security and Privacy*. Retrieved from http://www.stanford.edu/~jurafsky/burszstein_2010_captcha.pdf - March 15, 2021

Butrimas, V. (2018). Threat Intelligence Report: Cyberattacks Against Ukrainian Ics. *Report, Threat Intelligence*. NATO Cooperative Cyber Defence Centre of Excellence, Tallinn , Estonia. https://www.sentryo.net/wpcontent/uploads/2017/09/EBOOK_CYBERATTACKS-AGAINST-UKRAINIAN-ICS.pdf - May 18, 2021

Casas, P., Mazel, J., and Owezarski, P. (2012a). Knowledge-independent traffic monitoring: Unsupervised detection of network attacks. *IEEE Network*, *26*(1), 13–21.

Casas, P., Mazel, J., and Owezarski, P. (2012b). Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, *35*(7), 772–783.

Chandrasekhar, A. M., and Raghuveer, K. (2013, January). Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers. In Computer Communication and Informatics (ICCCI), 2013 IEEE International Conference on page 1-7.

Chellapilla, K., and Simard, P. Y. (2005). Using machine learning to break visual human interaction proofs (HIPs). *Advances in Neural Information Processing Systems*. Microsoft Research One Microsoft Way Redmond, WA 98052

Chen, J., Luo, X., Guo, Y., Zhang, Y., and Gong, D. (2017). A Survey on Breaking Technique of Text-Based CAPTCHA. *Security and Communication Networks*, *2017*, 1–15.

Chen, J., Luo, X., Hu, J., Ye, D., and Gong, D. (2018). An Attack on Hollow CAPTCHA Using Accurate Filling and Nonredundant Merging. *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)*, *35*(sup1), 106–118.

Chen, J., Luo, X., Liu, Y., Wang, J., and Ma, Y. (2019). Selective Learning Confusion Class for Text-Based CAPTCHA Recognition. *IEEE Access*, *7*, 22246–22259.

Chen, M., Challita, U., Saad, W., Yin, C., and Debbah, M. (2019). Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial. *IEEE Communications Surveys and Tutorials*, *21*(4), 3039–3071.

Chibuko R. I. B. (2015). *The Legal Aspects of Cybercrime in Nigeria: An Analysis with the UK Provisions (PhD Dissertation)*. University of Stirling.

Choudhury, M. M. (2009). *A study of the significant factors affecting trust in electronic commerce*. 1–364. Retrieved from http://ethesis.dur.ac.uk/2533/

Costa, K. A., Pereira, L. A., Nakamura, R. Y., Pereira, C. R., Papa, J. P., and Falcão, A. X. (2015). A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. Information Sciences, 294, 95-108.

Divyashree, N. (2018). Secured Conversion and Generation of Cognitive CAPTCHA Implementing Honeypot Technique. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *20*(3), 24–26.

Elbasiony, R. M., Sallam, E. A., Eltobely, T. E., and Fahmy, M. M. (2013). A hybrid network intrusion detection framework based on random forests and weighted k-means. *Ain Shams Engineering Journal*, *4*(4), 753–762.

https://doi.org/10.1016/j.asej.2013.01.003

Fahad, A., Tari, Z., Khalil, I., Habib, I., and Alnuweiri, H. (2013). Toward an efficient and scalable feature selection approach for internet traffic classification. Computer Networks, 57(9), 2040-2057.

Fahad, A., Tari, Z., Khalil, I., Almalawi, A., and Zomaya, A. Y. (2014). An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion. *Future Generation Computer Systems*, *36*, 156–169. https://doi.org/10.1016/j.future.2013.09.015

Gogoi, P., Bhattacharyya, D. K., Borah, B., and Kalita, J. K. (2014). MLH-IDS: A multi-level hybrid intrusion detection method. *Computer Journal*, *57*(4), 602–623. https://doi.org/10.1093/comjnl/bxt044

Hernandez-Castro, C. J., R-Moreno, M. D., and Barrero, D. F. (2015). Using JPEG to Measure Image Continuity and Break Capy and Other Puzzle CAPTCHAs. *IEEE Internet Computing*, *19*(6), 46–53. https://doi.org/10.1109/MIC.2015.127

Hernández-Castro, C. J., R-Moreno, M. D., Barrero, D. F., and Gibson, S. (2017). Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis. *Computers and Security*, Elsevier, Volume 70, September 2017, 744-756

Higgins, A. (2018). *Adaptive Containerised Honeypots for Cyber-Incident Monitoring (PhD Dissertation)*. Computer Engineering Department, University of Dublin, Trinity College.

Hindle, A., Godfrey, M. W., and Holt, R. C. (2008). Reverse Engineering CAPTCHAs. *University of Waterloo, Ontario*. Retrieved from http://yurichev.com/non-wikifiles/RE_for_beginners-ru.pdf

Hosseinpour, F., Amoli, P., Farahnakian, F., Plosila, J., and Hämäläinen, T. (2014). Artificial immune system based intrusion detection: Innate immunity using an unsupervised learning approach. *International Journal of Digital Content Technology and Its Applications*, *8*(5), 1.

Jha, M., and Acharya, R. (2016). An immune inspired unsupervised intrusion detection system for detection of novel attacks. In Intelligence and Security Informatics (ISI), 2016 IEEE, 292-297

Jatti, S. A. V., and Kishor Sontif, V. J. K. (2019). Intrusion detection systems. International Journal of Recent Technology and Engineering, 8(2 Special Issue 11), 3976–3983. https://doi.org/10.35940/ijrte.B1540.0982S1119

Kaspersky Lab. (2015). *Damage Control: The Cost of Security Breaches*. Retrieved from: https://media.kaspersky.com/pdf/it-risks-survey-report-cost-of-security-breaches.pdf - March 17, 2021

Kaur, E. N. (2018). *Introduction of Cyber Crime and Its Type*. *5*(08), 2014–2018. Retrieved from https://www.academia.edu/37288317/INTRODUCTION_OF_CYBER_CRIME_AND_ITS_TYPE - March 15, 2021

Khu-smith, V. (2003). *Enhancing the security of electronic commerce transactions*. (June). Retrieved from http://digirep.rhul.ac.uk/items/e42a99f4-78e1-69a3-87fd-6ac743f828ca/1/ - April 4, 2021

Kluever, K. A. (2008). Video CAPTCHAs : usability vs . security. *IEEE Workshop on Image Processing*, 1–4.

Li, Y., Wang, J. L., Tian, Z. H., Lu, T. B., and Young, C. (2009). Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. *Computers and Security*, *28*(6), 466–475. https://doi.org/10.1016/j.cose.2009.01.001

Lin, W. C., Ke, S. W., and Tsai, C. F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, *78*(1), 13–21. https://doi.org/10.1016/j.knosys.2015.01.009

Liu, Z., Wang, R., Tao, M., and Cai, X. (2015). A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion. Neurocomputing, 168, 365-381.

Ma, W., Qin, J., Xiang, X., Tan, Y., Luo, Y., and Neal, N. (2019). Adaptive Median Filtering Algorithm Based on Divide and Conquer and Its Application in CAPTCHA Recognition. *58*(3), 665–677.

Maguire, M., and Delahunt, B. (2017). Doing a Thematic Analysis: A Practical, Step-by-Step. *The All Ireland Journal of Teaching and Learning in Higher Education*, *8*(3), 3351.

Manzoor, S. M., and Soumya, K. R. (2014). Securing Websites through Multi-CAPTCHA. *International Journal of Computing and Technology (IJCAT)*, *1*(2), 59–62.

Menken, M. (2002). JESS tutorial. *VrijeUniversiteit, Amsterdam, TheNetherlands*, 1–57.

Miao, J., and Niu, L. (2016). A Survey on Feature Selection. *Procedia Computer Science*, *91*(Itqm), 919–926. https://doi.org/10.1016/j.procs.2016.07.111

Moradi, M., and Keyvanpour, M. (2015). CAPTCHA and its Alternatives: A Review. *Security and Communication Networks*. International Conference on Cross-Cultural Design, 2016 130-138.

Mphago, B. (2017). Deception in Web Application Honeypots: Case of Glastopf. *International Journal of Cyber-Security and Digital Forensics*, *6*(4), 179–185.

Nassar, N., and Miller, G. (2013). *Method for Two Dimensional Honeypot in a Web Application*. Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on Collaborative Computing

Nazario, J., and Holz, T. (2008). As the net churns: Fast-flux botnet observations. *3rd International Conference on Malicious and Unwanted Software, MALWARE 2008*, 24–31. https://doi.org/10.1109/MALWARE.2008.4690854

Nguyen, V. D. (2014). Contributions to Text-based CAPTCHA Security. (PhD Dissertation) *University of Wollongong*, 190.

Nisioti, A.; Mylonas, A.; Yoo, P.D.; and Katos, V. (2018) From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Commun. Surv. Tutor.*, 20, 3369–3388.

Nkwetta, A. J. (2018). *HONEY-SYSTEM : DESIGN , IMPLEMENTATION AND ATTACK ANALYSIS . College of Technology,University of Buea.* 2018.

Om, H., and Kundu, A. (2012). A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. In Recent Advances in Information Technology (RAIT), IEEE 1st International Conference on (pp. 131-136).

Omid, R., Gary, S., Large, J., and David, B. (2017). *In-Depth Study of CAPTCHA*. (April), 0–21. (Term Paper) University of Pennysylvania

Pope, C., and Kaur, K. (2005). Is it Human or Computer? Defending eCommerce with CAPTCHA. *IEEE Internet Computing*, (April), 43–49.

Powers, D. M. W. (2020). *Evaluation: from precision, recall and F-measure to ROC,* *informedness, markedness and correlation*. (January 2008). Retrieved from http://arxiv.org/abs/2010.16061 - May 16, 2021

Premanand, V., Meiappane, A., and Arulalan, V. A. (2015). Survey on Captcha and its Techniques for BOT Protection. *International Journal of Computer Applications*, *109*(5), 1–4. https://doi.org/10.5120/19181-0661

Raj, S. B. E., Devassy, D., and Jagannivas, J. (2011). A new architecture for the generation of picture based CAPTCHA. *ICECT 2011 - 2011 3rd International Conference on Advanced Computing, Networking and Security, 568-574*.

Rufus, R., Nick, W., Shelton, J., and Esterline, A. (2016). An autonomic computing system based on a rule-based policy engine and Artificial Immune Systems. *CEUR Workshop Proceedings*, *1584*(April), 105–108.

Rushikesh, K. (2019). Study on Honeypot Based Secure Network System. *International Journal of Advanced Research in Computer Science*, *10*(3), 71–72. https://doi.org/10.26483/ijarcs.v10i3.6420

Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Lu, W., Hakimian, P. (2011). Detecting P2P botnets through network behaviour analysis and machine learning. *2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011*, 174–180.

Sahu, N., and Richhariya, V. (2012). *Honeypot : A Survey*. International Journal of Computer Science And Technology, *8491*, 858–864.

Selvaraj, R., Kuthadi, V. M., and Marwala, T. (2016). *EIDPS : An Efficient Approach to Protect the Network and Intrusion Prevention*, Elsevier 35–47.

Shirali-Shahreza, M., and Shirali-Shahreza, S. (2008). Advanced collage CAPTCHA.

Proceedings - International Conference on Information Technology: New Generations, ITNG 2008, (July), 1234–1235.

Shukla A. K., Singh P. and Vardhan M., (2019) A New Hybrid Feature Subset Selection Framework Based on Binary Genetic Algorithm and Information Theory, International Journal for Computational Intelligence and Applications. 18(3), 1950020.

Smith, S. W. (2001). *WebALPS : A Survey of E-Commerce Privacy and Security Applications*. ACM SIGecom Exchanges 2.3 *1*(212).

Song, J., Takakura, H., Okabe, Y., and Nakao, K. (2013). Toward a more practical unsupervised anomaly detection system. *Information Sciences*, *231*, 4–14. https://doi.org/10.1016/j.ins.2011.08.011

Souley, B., and Abubakar, H. (2018). A Captcha - Based Intrusion Detection Model. *International Journal of Software Engineering and Applications*, *9*(1), 29–40. https://doi.org/10.5121/ijsea.2018.9103

Subramanyam, M., and Priya, V. (2018). *A Study of Captcha Techniques and Development of SUPER Captcha for A Study of Captcha Techniques and Development of SUPER Captcha for Secured Web Transactions*.

Symantec. (2016). Internet security threat report. *Network Security*, *21*(2), 1–3. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S1353485805001947

Thomas, K., and Nicol, D. M. (2010). The Koobface botnet and the rise of social malware? *Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware 2010*, (November 2010), 63–70. https://doi.org/10.1109/MALWARE.2010.5665793

Vahdani A. P., and Amoli, V. (2015). Unsupervised network intrusion detection systems for zero-day fast-spreading network attacks and botnets. *Jyväskylä Studies in Computing*, *10*(231), 1–13.

Vasilomanolakis, E., Karuppayah, S., Muhlhauser, M., and Fischer, M. (2015). Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys*, *47*(4). https://doi.org/10.1145/2716260

vonHan, L., Blum, M., and Langford, J. (2013). Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI. In *Communication of the ACM* (Vol. 3).

Winter-Hjelm C, Kleming M. H., Bakken R. H., (2009) An in-teractive 3D CAPTCHA with semantic information.Proceedings of the Norwegian AI Society, NAIS'2009, ACM, 2009; 157–160.

Woo, S. S., Rey, M. Del, and Kim, J. (2014). 3DOC : 3D Object CAPTCHA. *WWW'14 Companion*, 397–398.

Yamamoto, T., Tygar, J. D., and Nishigaki, M. (2010). CAPTCHA using strangeness in machine translation. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, (April), 430–437. https://doi.org/10.1109/AINA.2010.55

Ylonen, T. (2006). *The Secure Shell (SSH) Connection Protocol*. Information Technology Promotion Agency, Japan.

Retrieved https://www.ipa.go.jp/security/rfc/RFC4254EN.html - March 18, 2021

Yu, N., and Darling, K. (2019). A Low-Cost Approach to Crack Python CAPTCHAs Using AI-Based Chosen-Plaintext Attack. *Applied Sciences*, *9*(10).

https://doi.org/10.3390/app9102010

Zhang, H., Lu, G., Qassrawi, M. T., Zhang, Y., and Yu, X. (2012). Feature selection for optimizing traffic classification. Computer Communications, 35(12), 1457-1471.

Zhang, J., Hei, X., and Wang, Z. (2019). Typer vs . CAPTCHA : Private information based human cheating. *Beijing Electronics Science and Technology Institute*, 1–17.

Zhou, M., Huang, H., and Wang, Q. (2012). A graph-based clustering algorithm for anomaly intrusion detection. *ICCSE 2012 - Proceedings of 2012 7th International Conference on Computer Science and Education*, (Iccse), 1311–1314. https://doi.org/10.1109/ICCSE.2012.6295306

APPENDICES

Appendix A

Some major IDS Datasets

Citation	**Year**	**Type**	**Attack**	**Publicly Available**	**Description**	**# of Features**
UNB ISCX (M. Chen, Challita, Saad, Yin, and Debbah, 2019)	2012	Real life	ALL	Yes	Analysis of real packet traces in order to create profiles for traffic-generating agents in real life	19
ISOT Botnet (Saad et al., 2011)	2010	Benchmark	ALL	Yes	Series of combined multiple existing packets	
CAIDA (The CAIDA, 2016)	2008-2016	Benchmark	ALL	Yes	Randomized inactive backbone traffic without payload	Not Applicable
MAWI	2006-2016	Real life	N/A	Yes	Upstream ISP having a daily trace at the transit link	Not Applicable
LBNL	2005	Benchmark	DoS	Yes	Several activity hours from multiple internal hosts, randomized without payload	Not Applicable
UNIBS	2009	Real life	ALL	Yes	Transmission Control Protocol (99%) and User Datagram Protocol traffic	Not Applicable
DARPA	2000	Benchmark	DoS	Yes	Two distinct scenarios, LLDOS 1.0 and LLDOS 2.0.2	Not Applicable
KDD99	1999	Benchmark	ALL	Yes	Most circulated dataset	41
NSL-KDD	1999	Benchmark	ALL	Yes	KDD99-variant (url for reasons)	41
TUIBS	N/A	Real life	ALL	Not Applicable	Not Applicable	50,24
METROSEC	N/A	Real life	N/A	No	Not Applicable	Not Applicable
DEFCON	N/A	Benchmark	DoS	Not Applicable	CTF traffic	Not Applicable

Appendix B

Honeypot configuration file

```
<?php
/////////////////////////////////////////////////////
//Google Hack Honeypot v1.1
//Configuration File
//http://ghh.sourceforge.net - many thanks to SourceForge
/////////////////////////////////////////////////////

/////////////////////////////////////////////////////
//Begin Global Configuration Section
/////////////////////////////////////////////////////

/////////////////////////////////////////////////////
//Logging (CSV or MySQL?)
/////////////////////////////////////////////////////
//CSV, MySQL, or xml-rpc?
$LogType = 'xmlrpc'; //Enter 'CSV', 'xmlrpc', or 'MySQL', then complete the relevant
configuration section below

        //CSV Config
        $Filename = ''; //yourORIGINALfilename.txt (this better be original!!!!!) This is
where logs are being written to.

        //MySQL Config
        $Owner = ''; //There may be many people logging in the remote database, so who
are you? This will determine which logs are yours.
        $Server = ''; //MySQL Server (IP, IP:port, IP:port/path/to/socket)
        $DBUser = ''; //MySQL Username
        $DBPass = ''; //DB Password
        $DBName = ''; //Default ghh (name of the database)

        //XML-rpc Config
        $XMLhost = ''; //the hostname for the site that has xmlrpc example ghh.sf.net
        $XMLport = ''; //the port that xmlrpc is running on this is most likely port 80, if
you use something other then 80 we will try to connect with https.
        $XMLresource = ''; //the "path" to the xmlrpc server such as
'/ghh/xmlrpc/server.php'
        $XMLident = ''; //the string that identfies this host to the xml server
        $XMLmagic = ''; //the magic string that goes along with the host like a password
        $XMLrpc = 'xml.inc'; //the file to include that has xmlrpc (name it something
other then xmlrpc.inc or .php)
        $XMLhttps = false;

        $XMLproxy = false;  //if you are behind a proxy set this to true
```

```
	$XMLproxyHost = ""; //the host that you need to go through for the proxy
	$XMLproxyPort = 0; // the port for your proxy

/////////////////////////////////////////////////////
//End Global Configuration Section
/////////////////////////////////////////////////////

/////////////////////////////////////////////////////
//Begin Housekeeping Section
/////////////////////////////////////////////////////
$Signature = array();
$DateTime = date("m-d-Y h:i:s A");
$Attack = "";
$HoneypotName = "";
$Log = "";
error_reporting(0);
$downloadedFile = null;
/////////////////////////////////////////////////////
//End Housekeeping Section
/////////////////////////////////////////////////////
/////////////////////////////////////////////////////
//End Housekeeping Section
/////////////////////////////////////////////////////

/////////////////////////////////////////////////////
//Begin Basic Security Section (This makes the configuration file a honeypot itself to
prevent fingerprinting, no transparent links to this file please.)
/////////////////////////////////////////////////////
//Checks for $RegisterGlobals so $Honeypot cannot be bypassed

if(ini_get("register_globals")==1)
{
	if (strstr($_SERVER["REQUEST_URI"], "config.php"))
		unset($Honeypot);
}

if(!isset($Honeypot)){
	//Set Config honeypot's name
	$HoneypotName = "CONFIG.PHP";
	//Attack Acquisition Section
	$Attack = getAttacker();
	//Determine Standard Signatures
	$Signature = standardSigs($Attack, "none");
	//Build Log
	writeLog($Owner, $HoneypotName, $DateTime, $Attack, $Signature, $LogType,
$Filename, $DBName, $DBUser, $DBPass, $Server);
```

```
        exit;
}
////////////////////////////////////////////////////
//End Basic Security Section
////////////////////////////////////////////////////
////////////////////////////////////////////////////
//Begin Functions Section
//Contains core functions of GHH which are shared by all honeypot files.
//Function list: getAttacker(),standardSigs(),sanitize(), writelog(), buildLog()
////////////////////////////////////////////////////


function getFullHeaders() {
      return var_export($_SERVER, true) . var_export($_GET, true) .
var_export($_POST, true);
}
//downloadHTTPfile($host, $port, $resorce)
//this downloads the first 500k of a file and then puts the base64 of that in the global
varable downloadedFile
function downloadHTTPfile($host, $port, $resorce) {

        $connection = fsockopen($host, $port);
        $resorce = preg_replace($resorce, '//', '/?(.*)/');
        //set the headers to look like wget
        $request  = "GET $resorce HTTP/1.1\r\n";
        $request .= "Host: $host\r\n";
        $request .= "user-agent: Wget/1.10.2\r\n";
        $request .= "accept: */*\r\n";
        $request .= "content-length: 0\r\n";
        $request .= "Connection: Close\r\n\r\n";

        //did we fail to connect
        if (!$connection)
                return ;

        fwrite($connection, $request);
        $buffer = "";
        while (!feof($connection) andand strlen($buffer) < 1024 * 500)
        {
                        $in =  fgets($connection, 4096);
                        $buffer .= $in;
        }
        preg_replace($buffer, '//', '/'.chr(13).chr(10).chr(0).chr(10).'/');
        if (strlen($buffer) > 0)
                $GLOBALS['downloadedFile'] = base64_encode($buffer);
}
```

```
//getProxy() Detects a proxy. If the real IP is available, it's logged.
function getProxy() {
	$proxy = array();

	if(isset($_SERVER['HTTP_CLIENT_IP']))
		$proxy = array_merge($proxy, explode(',',
$_SERVER['HTTP_CLIENT_IP']));

	if(isset($_SERVER['HTTP_X_FORWARDED_FOR']))
		$proxy = array_merge($proxy, explode(',',
$_SERVER['HTTP_X_FORWARDED_FOR']));

	if (!count($proxy) > 0) {
		if (isset($_SERVER["HTTP_PROXY_CONNECTION"]) ||
isset($_SERVER["HTTP_VIA"])) {
			return "::Proxy Detected";
		}
		return "";
	}
	$proxy = implode('::', $proxy);
	return '::' . $proxy;
}

//Sanitize returns a version of the string passed that does not have any characters that
could cause problems with sql or html.
function sanitizeHtmlandSql($string) {
	if (strtolower($LogType) != "xmlrpc") {
		$ornament[0] = '/\and/';
		$ornament [1] = '/</';
		$ornament [2] = "/>/";
		$ornament [3] = '/"/';
		$ornament [4] = "/'/";
		$ornament [5] = "/%/";
		$ornament [6] = '/\(/';
		$ornament [7] = '/\)/';
		$ornament [8] = '/\+/';
		$ornament [9] = '/-/';
		$substitute[0] = 'and#26;';
		$substitute[1] = 'andlt;';
		$substitute[2] = 'andgt;';
		$substitute[3] = 'andquot;';
		$substitute[4] = 'and#39;';
		$substitute[5] = 'and#37;';
		$substitute[6] = 'and#40;';
		$substitute[7] = 'and#41;';
		$substitute[8] = 'and#43;';
```

```
                $substitute[9] = 'and#2d;';
        } else {
                $ornament [0] = '/ /';
                $substitute[0] = ' ';
        }
        $clean = substr(preg_replace($ornament, $substitute, $string),0,3000);
        return $clean;
}


//sanitize() returns $_SERVER['REQUEST_URI'] stripped of any illegal chars that may
corrupt the log when parsed into HTML.  500 character limit per field.
function sanitize($string, $maxlen=250) {
        if (strtolower($LogType) != "xmlrpc") {
                $ornament [0] = '/\and/';
                $ornament [1] = '/</';
                $ornament [2] = "/>/";
                $ornament [3] = '/\n/';
                $ornament [4] = '/"/';
                $ornament [5] = "/'/";
                $ornament [6] = "/%/";
                $ornament [7] = '/\(/';
                $ornament [8] = '/\)/';
                $ornament [9] = '/\+/';
                $ornament [10] = '/-/';
                $ornament [11] = '/,/';
                $substitute[0] = 'and#26;';
                $substitute[1] = 'andlt;';
                $substitute[2] = 'andgt;';
                $substitute[3] = '';
                $substitute[4] = 'andquot;';
                $substitute[5] = 'and#39;';
                $substitute[6] = 'and#37;';
                $substitute[7] = 'and#40;';
                $substitute[8] = 'and#41;';
                $substitute[9] = 'and#43;';
                $substitute[10] = 'and#45;';
                $substitute[11] = 'and#44;';
        } else {
                $ornament[0] = '/ /';
                $substitute[0] = ' ';
        }
        $clean = substr(preg_replace($ornament, $substitute, $string),0,$maxlen);
        return $clean;
}
```

```
//displayLog() returns nothing.  Writes results of captured honeypot attack to disk or
database
function displayLog($Accessor, $Honey, $TimeStamp, $Hack, $Imprint, $LogType,
$Filename, $DBN, $DBUsername, DBPassword, $Server) {
$SigLog = '';

        if(strtolower($LogType) == "mysql") {
                foreach ($Imprint as $string)
                        $SigLog .= $string . ';';
                //Host and user details are extracted from configuration file
                $link = mysql_connect($Server, $DBUsername, $DBPassword);
                if (!$link) {
                        die();
                }
                //The name of the DB is extracted from config file.
                $db = mysql_select_db($DBN);

                $query = "INSERT INTO logs ( `Accessor`, `Tripped`, `TimeOfHack`,
`Host`, `RequestURI`, `Referrer`, `Accepts`, `AcceptsCharset`, `AcceptLanguage`,
`Connection`, `keepalive`, `UserAgent`, `Imprints`, `Headers`)
VALUES ('" . $Accessor . "', '" . $Honey . "', NOW( ), '" . $Hack['IP'] . "', '" .
$Hack['request'] . "' , '" . $Hack['referer'] . "', '" . $Hack['accept'] . "', '" . $Hack['charset'] .
"', '" . $Hack['language'] . "', '" . $Hack['connection'] . "', '" . $Hack['keep_alive'] . "', '" .
$Hack['agent'] . "', '" .$SigLog . "', '" . $Hack['headers'] . "');";

                $result = mysql_query($query, $link);
                mysql_close($link);

        }else if (strtolower($LogType) == "xmlrpc") {
                include($GLOBALS['XMLrpc']);
                //make a connection to the xmlrpc server
                $server = new xmlrpc_client($GLOBALS['XMLresource'],
$GLOBALS['XMLhost'], $GLOBALS['XMLport']);
                //add xmlrpc debugging set to 1
                $server->setDebug(0);

                if ($GLOBALS['XMLproxy'])
                        $server->setProxy($GLOBALS['XMLproxyHost'],
$GLOBALS['XMLproxyPort']);

                //add ident and magic to the array
                $Attack['Ident'] = $GLOBALS['XMLident'];
                $Attack['Magic'] = $GLOBALS['XMLmagic'];

                //if we downloaded a file lets send it to our cental logging server
                $Attack['downloadedFile'] = $GLOBALS['downloadedFile'];
```

```
			//add the last few vars to the array
			foreach ($Signature as $string)
				$SigLog .= $string . ';';

			$Attack['SigLog'] = $SigLog;
			$Attack['Name'] = $HoneypotName;

			//convert our array and make a xmlrpc message to send out
			$XMLattack =new xmlrpcmsg('ghh.log',
array(php_xmlrpc_encode($Attack)));
			//send the message
			if (!$GLOBALS['XMLhttps']){
				$responce = $server->send($XMLattack, 0, "http");
			} else {
				$server->setSSLVerifyPeer(false);
				$responce = $server->send($XMLattack, 0, "https");
			}


		}
		else { //Type is CSV
			$Log = "";
			$Log = $Honey . "," . $TimeStamp . "," . $Hack['IP'] . "," .
$Hack['request'] . "," . $Hack['referer'] . "," . $Hack['accept'] . "," . $Hack['charset'] . "," .
$Hack['encoding'] . "," . $Hack['language'] . "," . $Hack['connection'] . "," .
$Hack['keep_alive'] . "," . $Hack['agent'] . ",";

		}
/////////////////////////////////////////////////////
//End Functions Section
/////////////////////////////////////////////////////
/////////////////////////////////////////////////////
//End of config.php
/////////////////////////////////////////////////////
?>
```

Sample Honeypot PHPShell

```
<?php
//////////////////////////////////////////////////////
//Google Hack Honeypot v1.1
//Template File
//http://ghh.sourceforge.net - many thanks to SourceForge
//////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////
//Begin Configuration Section
////////////////////////////////////////////////////

//Enter the path to the GHH global configuration file
//(a smart move would be to change the config.php filename.)
$ConfigFile = '';

//Enter the URL of the page that links to this honeypot. This will help detect false
positives
//where a user finds your transparent link.
//(I.E http://yourdomain.com/forums/index.php, Wherever you put your transparent link
to the honeypot.)
$SafeReferer = '';

//The Honeypot will appear to run under this username.
$Username = '';

////////////////////////////////////////////////////
//End Configuration Section
////////////////////////////////////////////////////

////////////////////////////////////////////////////
//Housekeeping Section
//Include config, disable the header protection, init variables, stealth the errors.
////////////////////////////////////////////////////
error_reporting(0);
$Honeypot = true;
include($ConfigFile);
////////////////////////////////////////////////////
//End housekeeping section
////////////////////////////////////////////////////

//Attack Acquisition Section
$Hack = getHacker();

//Determine Standard Signatures
$Imprint = standardSigs($Attack, $SafeReferer);

////////////////////////////////////////////////////
//Begin Custom Honeypot Section
//GHH Honeypot by Brian Engert, Ryan McGeehan for GHDB Signature #365
(intitle:"PHP Shell *" "Enable stderr" filetype:php)
////////////////////////////////////////////////////
$Honey = "PHPSHELL";
```

```
//Beginning Shell Emulation
$output= '';
if(isset($_POST['command']))
{
	//they sent us a command so let's look for ; then emulate it
	$command = $_POST['command'];
	$commands = explode(";", $command);
	//echo "command = $command <br>";//debug code
	foreach ($commands as $cmd)
	{
		//now I have each command with it's paramaters in a seperate string in the
commands array
		$space = strpos($cmd, " ");	//if this space is inside of quotes we want to
keep looking
		if (strcmp($space,"") == 0)
		 $space = strlen($cmd);//we don't have a space so make the "space" the
end of the string

		$myCommand = substr($cmd, 0, $space);
		$paramaters = substr($cmd, $space+1, strlen($cmd));
		$output .= runCommand($myCommand, $paramaters, $Username)."\n";\
	}
}

//get the url up to the first ? so we have some proper links, prevent proxied attacks
$ourfile = $_SERVER['REQUEST_URI'];
$question = strpos($ourfile, '?');
if (strcmp($question,"") == 0)
	$question =strlen($ourfile);
$ourfile = substr($ourfile, 0, $question);


//Trick PHP Shell page
echo <<< Heredoc
<html>
<head>
<title>PHP Shell 1.7</title>
</head>
<body>
<h1>PHP Shell 1.7</h1>


<form name="myform" action="{$ourfile}" method="post">
<p>Current working directory: <b>
<a href="{$ourfile}?work_dir=/">Root</a>/</b></p>
```

```
<p>Choose new working directory:
<script language="JavaScript" type="text/javascript">
document.forms[0].command.focus();
</script>

<hr>
<i>Copyright andcopy; 2000andndash;2002, <a
href="mailto:gimpster@gimpster.com">Martin Geisler</a>. Get the latest
version at <a href="http://www.gimpster.com">www.gimpster.com</a>.</i>

</body>
</html>

Heredoc;

//View Commands Hacker is running
if(isset($_POST['command']))
        $Imprints[] = $_POST['command'];

//Find our PHP shell target in the referer site
if (strstr($Hack['referer'], "Shell")){
         $Imprints[] = "Target in URL";
}

//Finds if exact GHDB signature was used
if (strstr ($Hack['referer'],
"intitle%3A%22PHP+Shell+*%22+%22Enable+stderr%22+filetype%3Aphp")){
         $Imprints[] = "GHDB Imprint!";
}

//"Execute" commands like id, uname, wget. See associated functions below
function runCommand($command, $paramaters, $Username) {

  if(strcmp($command, 'id') == 0 || strcmp("/usr/bin/id", $command) == 0){
        $output = id($paramaters);
  }else if(strcmp($command, 'uname') == 0 || strcmp("/bin/uname", $command) == 0){
        return uname($paramaters);
  }else if(strcmp($command, 'wget') == 0 || strcmp("/usr/bin/wget", $command) == 0){
        $output = wget($paramaters);
  }else if (strcmp($command, 'w') == 0 || strcmp("/usr/bin/w", $command) == 0){
    $output = w($paramaters, $Username);
  }else if(strcmp($command, 'whoami') == 0 || strcmp("/usr/bin/whoami", $command) ==
0){
        $output = whoami($paramaters, $Username);
  }else if(strcmp($command, 'pwd') == 0 || strcmp("/bin/pwd", $command) == 0){
        if (strstr($paramaters, '-'))
```

```
                    $output = "-bash: pwd: $paramaters: invalid option\npwd: usage: pwd [-
PL]";
          else
             $output = "/home/$Username/htdocs";
  }else if(strcmp("ps", $command) == 0 || strcmp("/bin/ps", $command) == 0){
    $output = "  PID TTY        TIME COMMAND\n16919 pts/0     0:00 bash";
  }else if(strcmp("cat", $command) == 0){
          $output = cat($paramaters, $Username);
  }else if(strcmp("ls", $command) == 0 || strcmp("/bin/ls", $command) == 0){
          $output = ls($paramaters);
  }else if(strcmp("ping", $command) == 0 || strcmp("/bin/ping", $command) == 0){
       $output = ping($paramaters);
  }else if(strcmp("/bin/echo", $command) == 0 || strcmp("echo", $command) == 0){
       $output = descapeQuotes($paramaters);
  }else if(strcmp("/bin/bash", $command) == 0 || strcmp("bash", $command) == 0){
       $output = "";
  }else if(strcmp("uptime", $command) == 0 || strcmp("/usr/bin/uptime", $command) ==
0){
          $output = uptime($paramaters);
  }else{
    $output = ""; //the real phpshell does not give bash errors
  }
  return $output;
}
function descapeQuotes($string) {
   $string = preg_replace('/[^\\\\]([\'"])/', '', $string); //remove non escaped ' and "
   $string = preg_replace('/\\\\(\')/', '\'', $string); //replace escaped ' and " with just the char
that's being escaped
   $string = preg_replace('/\\\\(\")/', '"', $string); //replace escaped ' and " with just the char
that's being escaped
   return $string;
}
function whoami($paramaters, $Username)
{
          if (strstr($paramaters, '--help'))
                    $output = <<<whoamidump
Usage: whoami [OPTION]...
Print the user name associated with the current effective user id.
Same as id -un.

     --help     display this help and exit
     --version  output version information and exit

Report bugs to <bug-coreutils@gnu.org>.
whoamidump;
          else if (strstr($paramaters, '--version'))
		$output = <<<moreversions
whoami (GNU coreutils) 5.2.1
Written by Richard Mlynarik.

Copyright (C) 2004 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
moreversions;
	else if (strcmp($paramaters, '') == 0)
		$output = $Username;
	else
		$output = "Try `whoami --help' for more information.";
	return $output;
}
function uptime($paramaters)
{
	$time = date("g:ia");
	$load1 = rand(35,60)/100;
	$load2 = rand(35,60)/100;
	$load3 = rand(35,60)/100;
	$uptime = date("z") + rand(1,20);  //today plus 50 days

	if (strstr($paramaters, '-V'))
		$output = "procps version 3.2.1";
	else if (strcmp($paramaters, '') == 0)
		$output = $time." up ".$uptime." days, 15:24, 1 user, load averages: 
".$load1.", ".$load2.", ".$load3."";
	else
		$output = "usage: uptime [-V]\n    -V    display version";
	return $output;
}
function cat($paramaters, $Username)
{
	if (strstr($paramaters, '--help'))
		$output = "Usage: cat [OPTION] [FILE]...
Report bugs to <bug-coreutils@gnu.org>.";
	else if (strstr($paramaters, '--version'))
		$output = "cat (coreutils) 5.2.1\nWritten by Torbjorn Granlund and 
Richard M. Stallman.\n\nCopyright (C) 2004 Free Software Foundation, Inc.\nThis is 
free software; see the source for copying conditions.  There is NO\nwarranty; not even 
for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.";
	else if (strstr($paramaters, '>') || strstr($paramaters, '<'))
		$output = "";
	else if (strstr($paramaters, '/etc/passwd'))
		$output = "root:x:0:0:root:/root:/bin/bash
```

```
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
Debian-exim:x:102:102::/var/spool/exim4:/bin/false
identd:x:100:65534::/var/run/identd:/bin/false
sshd:x:101:65534::/var/run/sshd:/bin/false
bind:x:103:104::/var/cache/bind:/bin/false
postfix:x:104:105::/var/spool/postfix:/bin/false
mysql:x:105:107:MySQL Server,,,:/var/lib/mysql:/bin/false
$Username:x:1000:1000:,,,:/home/$Username:/bin/bas";

        else if (strstr($paramaters, '/etc/hosts'))
                $output = "127.0.0.1       localhost.localdomain   localhost

<?php

define('PHPSHELL_VERSION', '1.7');

?>

<html>
<head>
<title>PHP Shell <?php echo PHPSHELL_VERSION ?></title>
</head>
<body>
<h1>PHP Shell <?php echo PHPSHELL_VERSION ?></h1>

<form name=\"myform\" action=\"<?php echo \$PHP_SELF ?>\" method=\"post\">
<p>Current working directory: <b>
<?php

\$work_dir_splitted = explode('/', substr(\$work_dir, 1));
```

```
echo '<a href=\"' . \$PHP_SELF . '?work_dir=/\">Root</a>/';

if (!empty(\$work_dir_splitted[0])) {
  \$path = '';
  for (\$i = 0; \$i < count(\$work_dir_splitted); \$i++) {
    \$path .= '/' . \$work_dir_splitted[\$i];
    printf('<a href=\"%s?work_dir=%s\">%s</a>/',
           \$PHP_SELF, urlencode(\$path), \$work_dir_splitted[\$i]);
  }
}

?></b></p>
<p>Choose new working directory:
<select name=\"work_dir\" onChange=\"this.form.submit()\">
<?php

\$dir_handle = opendir(\$work_dir);

while (\$dir = readdir(\$dir_handle)) {
  if (is_dir(\$dir)) {
    if (\$dir == '.') {
      echo \"<option value=\\\"\$work_dir\\\" selected>Current Directory</option>\n\";
    } elseif (\$dir == '..') {

             directory is the root directory (/). */
      if (strlen(\$work_dir) == 1) {

      } elseif (strrpos(\$work_dir, '/') == 0) {

      echo \"<option value=\\\"/\\\">Parent Directory</option>\n\";
      } else {


phpshellz;
          else if (strcmp($paramaters, '') == 0)
                    $output = "";
          else
                    $output = "cat: ".$paramaters.": No such file or directory";
          return $output;
}
function ls($paramaters)
{
          $pattern = "/\/([a-zA-Z-._]*)$/";
          preg_match($pattern, $_SERVER['REQUEST_URI'], $matches);
          //this gives me the file that should be in ls :-D
```

```
        if (strstr($paramaters, '--help'))
                $output = <<<lsmandump

lsmandump;
        else if (strstr($paramaters, '-la') || strstr($paramaters, '-al') || (strstr($paramaters, '-
a') andand strstr($paramaters, '-l')))
                $output = "total 16
drwxr-xr-x  2 $Username $Username    80 2005-11-30 10:37 .
drwxr-xr-x  3 $Username $Username    80 2005-11-30 10:33 ..
-rw-r--r--  1 $Username $Username 12976 2005-11-30 10:34 index.php
-rw-r--r--  1 $Username $Username 12976 2005-11-30 10:34 ".sanitize($matches[1])."
-rw-r--r--  1 $Username $Username 12976 2005-11-30 10:34 config.php
";
        else if (strstr($paramaters, '-l'))
                $output = "total 16
-rw-r--r--  1 $Username $Username 12976 2005-11-30 10:34 index.php
-rw-r--r--  1 $Username $Username 12976 2005-11-30 10:34 ".sanitize($matches[1])."
-rw-r--r--  1 $Username $Username 12976 2005-11-30 10:34 config.php
";
        else if (strcmp($paramaters, '') == 0)
                $output = "index.php  ".sanitize($matches[1])."       config.php";
        else
        $output = "ls: unrecognised option `$paramaters'\nTry `ls --help' for more
information.";

        return $output;
}
function ping($paramaters)
{
        $paramater = explode(" ", $paramaters);
      foreach ($paramater as $param)
      {
        if (strstr($param, '.'))
           $domainip = $param;//this is either our domain or ip of what we are going to
"Ping"
      }
      if (isset($domainip)){
        $ip = gethostbyname($domainip);
        $ttl = rand(60,120);
        $timea = rand(100,130);//if it goes below 100 it's still 3 digits so this makes it ezer
        $timeb = rand(100,130);
        $timec = rand(100,130);
        $timed = rand(100,130);
        sleep(($timeb + $timec + $timed)/1000 + 2);
        //I should do this for as meny as I'm told to do with -t but amm don't want to right
now
```

```
         $output = "PING $domainip ($ip) 56(84) bytes of data.
64 bytes from $domainip ($ip): icmp_seq=1 ttl=$ttl time=$timea ms
64 bytes from $domainip ($ip): icmp_seq=2 ttl=$ttl time=$timeb ms
64 bytes from $domainip ($ip): icmp_seq=3 ttl=$ttl time=$timec ms
64 bytes from $domainip ($ip): icmp_seq=4 ttl=$ttl time=$timed ms

--- $domainip ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = $timea/$timeb/$timed/$timec ms
";

      }else {
         $output = <<<pingpong
Usage: ping [-LRUbdfnqrvVaA] [-c count] [-i interval] [-w deadline]
         [-p pattern] [-s packetsize] [-t ttl] [-I interface or address]
         [-M mtu discovery hint] [-S sndbuf]
         [ -T timestamp option ] [ -Q tos ] [hop1 ...] destination
pingpong;
      }
      return $output;
}
function w($paramaters, $Username)
{
         $time = date("g:ia");
   $load1 = rand(35,60)/100;
   $load2 = rand(35,60)/100;
   $load3 = rand(35,60)/100;
   $uptime = date("z") + 50;  //today plus 50 days
         if (strstr($paramaters, '-V'))
                  $output = "procps version 3.2.1";
         else if (strstr($paramaters, '-h')){
                  if (strstr($paramaters, 'f')){
                           if (strstr($paramaters, 's')){
                                    $output = "$Username     pts/0      0.00s -bash";
                           }else{
                                    $output = "$Username     pts/0     15:58    0.00s  0.04s
0.01s -bash";
                           }
                  }else if (strstr($paramaters, 's')){
                           $output = "$Username     pts/0    -               0.00s -bash";
                  }
         }else if (strstr($paramaters, '-s')){
                  if (strstr($paramaters, 'f')){
                           $output = " $time up 1 day, 21:05,  1 user,  load average: $load1,
$load2, $load3\nUSER     TTY        IDLE WHAT\n$Username     pts/0      0.00s -bash";
                  }else
```

```
                        $output = " $time up $uptime day, 21:07,  1 user,  load average:
$load1, $load2, $load3\nUSER     TTY      FROM              IDLE WHAT\n$Username
pts/0    -                0.00s -bash";
        }else if (strstr($paramaters, '-f')){
                $output = " $time up $uptime day, 21:10,  1 user,  load average: $load1,
$load2, $load3\nUSER     TTY        LOGIN@   IDLE   JCPU   PCPU
WHAT\n$Username     pts/0     15:58    0.00s  0.04s  0.01s -bash";
        }else if (strcmp($paramaters, '') == 0)
                $output = " $time up $uptime day, 20:51,  1 user,  load average: $load1,
$load2, $load3\nUSER     TTY      FROM            LOGIN@   IDLE   JCPU   PCPU
WHAT\n$Username     pts/0    -              15:58    0.00s  0.03s  0.01s w";
        else
                $output = "  16:34:20 up 1 day, 21:27,  1 user,  load average: $load1,
$load2, $load3\nUSER     TTY      FROM            LOGIN@   IDLE   JCPU   PCPU
WHAT";
        return $output;
}
function id($paramaters)
{
        $output = "uid=0(root) gid=0(root) groups=0(root)";
        if (strstr($paramaters, '--help'))
        {
                $output = <<<idhelp

Report bugs to <bug-coreutils@gnu.org>.
idhelp;
        }
        return $output;
}
function uname($paramaters)
{
   $date = date("D M j G:i:s");
        $output = "";
        if (strstr($paramaters, '-')){
      //the next array of if's are correct order I checked the combos and you only need 1 -
at least for snrvmo
                if (strstr($paramaters, '--help'))
                {
                        $output = <<<HEREDOC
debian:/home/lart# uname --hes
uname: unrecognised option `--hes'
Try `uname --help' for more information.
HEREDOC;
                        return $output;
                }
                if (strstr($paramaters, 's') || strstr($paramaters, '--kernel-name'))
```

```
                        $output .= "Linux ";
                if (strstr($paramaters, 'n') || strstr($paramaters, '--nodename'))
                        $output .= "debian ";
                if (strstr($paramaters, 'r') || strstr($paramaters, '--kernel-release'))
                        $output .= "2.6.8-2-k7 ";
                if (strstr($paramaters, 'v') || strstr($paramaters, '--kernel-version'))
                        $output .= "#1 $date ";
                if (strstr($paramaters, 'm') || strstr($paramaters, '--machine'))
                        $output .= "i686 ";
                if (strstr($paramaters, 'o') || strstr($paramaters, '--operating-system'))
                        $output .= "GNU/Linux";
                if (strstr($paramaters, 'a') || strstr($paramaters, '--all'))
                        $output .= "Linux debian 2.6.8-2-k7 #1 $date i686 GNU/Linux";
                if (strlen($output) == 0)
                {
                        $output = <<<badparam
uname: unrecognised option `$paramaters'
Try `uname --help' for more information.
badparam;
                }
        }else
                $output = "Linux";
        return $output;
}
function wget($paramaters)
{
        $time = date("g:ia");
   $size = rand(5120,1048576);
        $speed = rand(30,500);
   $datetime = date("m-d-Y.G-i-s");

   if (strstr($paramaters, '--help')){
                $output = "GNU Wget 1.9.1, a non-interactive network retriever.
Usage: wget [OPTION]... [URL]...

Copyright (C) 2003 Free Software Foundation, Inc.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

Originally written by Hrvoje Niksic <hniksic@xemacs.org>.";

        }else if (strstr($paramaters, 'http://')){
                if (ereg("http:\/\/.*\/.*\/",$paramaters)){
             //file is inside of a folder
```

```
                //echo "file is in a folder<br>";//go debug code
            $pattern = "/http:\/\/([a-zA-Z\.-]*)(\/.*)\/(.*)/";
            preg_match($pattern, $paramaters, $matches);
            $domain =  sanitize($matches[1]);
            $folder = sanitize($matches[2]);
            $file =  sanitize($matches[3]);
        } else {
                //echo "file is not in a folder<br>";//go debug code
            $pattern = "/http:\/\/(.*)\/(.*)/";
            preg_match($pattern, $paramaters, $matches);
            $domain =  sanitize($matches[1]);
            $folder = "";
            $file =  sanitize($matches[2]);
                }
                //downloads a file and sends it with our xmlrpc
                downloadHTTPfile($domain, 80, $folder . "/" . $file);

                $ip = gethostbyname($domain);

                $url =  "http://" . $domain . $folder . "/" . $file;
                $cleanSysURL = sanitize_system_string($url, 5, 256);

                if ($file == '')
                {
                        $file = "index.html";
                }

                $output = "--$time--  " . $url . "\n=> `" . $file . "'\nResolving " .$domain .
"... " . $ip ."\nConnecting to " .$domain . "[" . $ip ."]:80... connected.\nHTTP request
sent, awaiting response... 200 OK\nLength: $size
[text/html]\n\n100%[====================================>] $size        --.--
K/s\n\n14:36:50 ($speed K/s) - `" . $file . "' saved [$size/$size]";
        }else if (strstr($paramaters, 'ftp://') andand strstr($paramaters, '@') ){
                $pattern = "/ftp:\/\/(.*):(.*)@/i";
                preg_match($pattern, $paramaters, $matches);
                $username = sanitize($matches[1]);
                $password = sanitize($matches[2]);
                $domain =  sanitize($matches[3]);
                $ip = gethostbyname($domain);
                $folder = sanitize($matches[4]);
                $file =  sanitize($matches[5]);
                $url = 'ftp://'.$username.':'.$password.'@'.$domain.$folder.'/'.$file;
                $output = "--$time--  ".$url."\n          => `".$file."'\nResolving
".$domain."... ".$ip."\nConnecting to ".$domain."[".$ip."]:21... connected.\nLogging in
as ".$username." ... Logged in!\n==> SYST ... done.    ==> PWD ... done.\n==> TYPE I
... done.  ==> CWD ".$folder." ... done.\n==> PASV ... done.    ==> RETR ".$file." ...
```

```
done.\nLength: 729,821
(unauthouritative)\n\n100%[======================================>]
729,821      155.12K/s    ETA 00:00\n\n15:44:07 (146.77 KB/s) - `".$file."' saved
[729821]";
          }else if (strstr($paramaters, 'ftp://')){
                    if (ereg("ftp:\/\/.*\/.*\/",$paramaters)) {
                              $pattern = "/ftp:\/\/([a-zA-Z\.-]*)(\/.*)\/(.*)/";
                              preg_match($pattern, $paramaters, $matches);
                              $domain =  sanitize($matches[1]);
                              $folder = sanitize($matches[2]);
                              $file =  sanitize($matches[3]);
                    } else {
                              $pattern = "/(ftp|http):\/\/([a-zA-Z\.-]*)\/(.*)/";
                              preg_match($pattern, $paramaters, $matches);
                              $domain =  sanitize($matches[2]);
                              $folder = "/";
                              $file =  sanitize($matches[3]);
                    }
       if (strcmp($folder,"") == 0)
       {
          $folder = "/";
       }
       if (strcmp($file, ""))
          $file = ".listing";
                    $username = "anonymous";
                    $ip = gethostbyname($domain);

                    $url = 'ftp://'.$domain.$folder.'/'.$file;
                    $output = "--$time--  ".$url."\n           => `".$file."'\nResolving
".$domain."... ".$ip."\nConnecting to ".$domain."[".$ip."]:21... connected.\nLogging in
as ".$username." ... Logged in!\n==> SYST ... done.    ==> PWD ... done.\n==> TYPE I
... done.  ==> CWD ".$folder." ... done.\n==> PASV ... done.    ==> RETR ".$file." ...
done.\nLength: 729,821
(unauthouritative)\n\n100%[======================================>]
729,821      155.12K/s    ETA 00:00\n\n15:44:07 (146.77 KB/s) - `".$file."' saved
[729821]";
          }else if (preg_match("/\.[a-zA-Z]{2,4}\/.*/",$paramaters)) {
                    $params = explode(" ", $paramaters);
                    foreach ($params as $parma)
                    {
                              if (preg_match("/\.[a-zA-Z]{2,4}\/.*/",$parma))
                                        $url = "http://" .$parma;
                    }
                    $urlstuff = parse_url($url);

                    $domain =  $urlstuff['host'];
```

```
                $lastSlash =  strrpos($urlstuff['path'], '/');
                $folder = substr($urlstuff['path'], 0, $lastSlash);
                $file =  substr($urlstuff['path'], $lastSlash+1, strlen($urlstuff['path'])-1);


                downloadHTTPfile($domain, 80, $folder . "/" . $file);
                $ip = gethostbyname($domain);

                $url =  "http://" . $domain . $folder . "/" . $file;
                $cleanSysURL = sanitize_system_string($url, 5, 256);

                if ($file == '')
                {
                        $file = "index.html";
                }
                $output = "--$time--  " . $url . "\n=> `" . $file . "'\nResolving " .$domain .
"... " . $ip ."\nConnecting to " .$domain . "[" . $ip ."]:80... connected.\nHTTP request
sent, awaiting response... 200 OK\nLength: $size
[text/html]\n\n100%[====================================>] $size        --.--
K/s\n\n14:36:50 ($speed K/s) - `" . $file . "' saved [$size/$size]";

        }else{//either I got a weird/bad url or they didn't give me http/ftp
                $output = "wget: missing URL\nUsage: wget [OPTION]...
[URL]...\n\nTry `wget --help' for more options.";
        }

        return $output;
}


?>
```

Appendix C

Server Program: Implementing the Diffie-Hellman Key Exchange Algorithm in Java

```
import java.net.*;
import java.io.*;

public class GreetingServer {
   public static void main(String[] args) throws IOException
   {
      try {
         int port = 8088;

         // Server Key
         int b = 3;

         // Client p, g, and key
         double clientP, clientG, clientA, B, Bdash;
         String Bstr;

         // Established the Connection
         ServerSocket serverSocket = new ServerSocket(port);
         System.out.println("Waiting for client on port " + serverSocket.getLocalPort() +
"...");
         Socket server = serverSocket.accept();
         System.out.println("Just connected to " + server.getRemoteSocketAddress());

         // Server's Private Key
         System.out.println("From Server : Private Key = " + b);

         // Accepts the data from client
         DataInputStream in = new DataInputStream(server.getInputStream());

         clientP = Integer.parseInt(in.readUTF()); // to accept p
         System.out.println("From Client : P = " + clientP);

         clientG = Integer.parseInt(in.readUTF()); // to accept g
         System.out.println("From Client : G = " + clientG);

         clientA = Double.parseDouble(in.readUTF()); // to accept A
         System.out.println("From Client : Public Key = " + clientA);

         B = ((Math.pow(clientG, b)) % clientP); // calculation of B
         Bstr = Double.toString(B);

         // Sends data to client
```

```
            // Value of B
            OutputStream outToclient = server.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToclient);

            out.writeUTF(Bstr); // Sending B

            Bdash = ((Math.pow(clientA, b)) % clientP); // calculation of Bdash

            System.out.println("Secret Key to perform Symmetric Encryption = "
                        + Bdash);
            server.close();
         }

         catch (SocketTimeoutException s) {
            System.out.println("Socket timed out!");
         }
         catch (IOException e) {
         }
      }
   }
```

Client Program: Implementing the Diffie-Hellman Key Exchange Algorithm in Java

```
import java.net.*;
import java.io.*;

public class GreetingClient {
   public static void main(String[] args)
   {
      try {
         String pstr, gstr, Astr;
         String serverName = "localhost";
         int port = 8088;

         // Declare p, g, and Key of client
         int p = 23;
         int g = 9;
         int a = 4;
         double Adash, serverB;

         // Established the connection
         System.out.println("Connecting to " + serverName
                     + " on port " + port);
         Socket client = new Socket(serverName, port);
         System.out.println("Just connected to "
                     + client.getRemoteSocketAddress());

         // Sends the data to client
         OutputStream outToServer = client.getOutputStream();
         DataOutputStream out = new DataOutputStream(outToServer);

         pstr = Integer.toString(p);
         out.writeUTF(pstr); // Sending p

         gstr = Integer.toString(g);
         out.writeUTF(gstr); // Sending g

         double A = ((Math.pow(g, a)) % p); // calculation of A
         Astr = Double.toString(A);
         out.writeUTF(Astr); // Sending A

         // Client's Private Key
         System.out.println("From Client : Private Key = " + a);

         // Accepts the data
         DataInputStream in = new DataInputStream(client.getInputStream());

         serverB = Double.parseDouble(in.readUTF());
         System.out.println("From Server : Public Key = " + serverB);

         Adash = ((Math.pow(serverB, a)) % p); // calculation of Adash

         System.out.println("Secret Key to perform Symmetric Encryption = "
                     + Adash);
         client.close();
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Appendix D

BOT programs already captured by our Hybridized secure IDS

First BOT program

```
from selenium import webdriver
from selenium.webdriver.common.proxy import Proxy, ProxyType
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
import schedule
from datetime import date, timedelta, time, datetime
from time import sleep

from secret import friends, service_date, check


class AttBot():
   def __init__(self, username, pw, service_type):
      print("Opening browser...")
      self.driver = webdriver.Chrome()
      self.wait = WebDriverWait(self.driver, 10)
      self.username = username
      self.pw = pw
      self.service_type = service_type
      self.error = False

   def login(self, count=0):
      print("Accessing login page...")
      self.driver.get("https://att.lmu.edu.ng/log/login")
      sleep(2)

      if count < 5:
         try:
            print("Trying to login "+self.username+"...")
            uname = self.driver.find_element_by_xpath("//*[@id='name']")
            uname.send_keys(self.username)

            password = self.driver.find_element_by_xpath(
               "//*[@id='content-inner']/div/div/form/fieldset/div[2]/input")
            password.send_keys(self.pw)

            submit = self.driver.find_element_by_xpath(
               "//*[@id='content-inner']/div/div/form/fieldset/input[1]")
            submit.click()
            try:
                login_error = self.driver.find_element_by_xpath(
                    "//*[@id='content-inner']/div/div/form/fieldset/div[1]/div")
                if login_error.text == "Wrong Username or Password!":
                    print("Wrong Username or Password!")
                    self.error = True
            except:
                print("Logged in successfully")
                count = 0
                self.error = False
        except:
            print("Couldn't login "+self.username+" trying again...")
            self.error = True
            self.login(count+1)
    else:
        print("We tried to login "+self.username+" 5times but it failed")
        self.error = True

def book(self, count=0):
    print("Automating "+self.username+" chapel service...")
    self.driver.get("https://att.lmu.edu.ng/check/serveChoice")
    sleep(2)
    if count < 5:
        # Get element with tag name 'tbody'
        tbody = self.driver.find_element_by_tag_name('tbody')

        # Get all the elements available with tag name 'tr'
        tr = tbody.find_elements_by_tag_name('tr')
        confirm = False
        for e in tr:
            tds = e.find_elements_by_tag_name('td')
            for td in tds:
                if service_date in td.text:
                    confirm = True

        select_error = False
        selects_check = self.driver.find_elements_by_xpath(
            "//*[@id='page']/form/div/div[2]/select")
        for select_check in selects_check:
            if "You have missed Roll Call last week!" in select_check.text:
                print("You have missed Roll Call last week!")
                select_error = True
            elif "Chapel Service Closed for the Semester" in select_check.text:
                print("Chapel Service Closed for the Semester")
                select_error = True

        if not confirm:
```

```
        if not select_error:
            try:
                print(
                    "Booking your prefered service, make sure you didn't miss roll call")

                if self.service_type == "first":
                    service_choice = self.driver.find_element_by_xpath(
                        '//*[@id="page"]/form/div/div[2]/select/option[1]')
                    if "" in service_choice.text:
                        print("First service has ended")
                    else:
                        service_choice.click()
                else:
                    service_choice = self.driver.find_element_by_xpath(
                        '//*[@id="page"]/form/div/div[2]/select/option[2]')
                    service_choice.click()

                agree = self.driver.find_element_by_xpath(
                    '//*[@id="confirm_remove_original"]')
                agree.click()

                # Wait for the alert to be displayed
                self.wait.until(expected_conditions.alert_is_present())

                # Store the alert in a variable for reuse
                alert = self.driver.switch_to.alert

                # Press the Cancel button
                alert.accept()

                self.driver.find_element_by_css_selector(
                    "input[type='submit']").click()
            except:
                print("Couldn't book service for " +
                      self.username+", something went wrong")
                print("Trying again...")
                self.book(count+1)
            else:
                print("Automatation was successful for " + self.username +
                      " check it booked the prefered service")
    else:
        print("You have booked chapel service already")
else:
    print("We tried to book service " +
          self.username+" 5times but it failed")
    def logout(self):
        self.driver.get("https://att.lmu.edu.ng/log/logout")
        sleep(2)


# def main_task():
# if check():
for friend in friends:
    bot = AttBot(friend["username"], friend["pw"], friend["service_type"])
    bot.login()
    if bot.error == False:
        bot.book()
        bot.logout()
    bot.driver.close()
# else:
#     print("Can't not book chapel service now")

# schedule.every().tuesday.at("18:00").do(main_task)
# schedule.every().wednesday.do(main_task)
# schedule.every().thursday.do(main_task)

# while True:
#     schedule.run_pending()
#     sleep(1)
```

Second BOT program

```
const cron = require('node-cron');
const httpx = require('axios');
var qs = require('querystring');

let cookieJar;

var d = new Date();
d.setDate(d.getDate() + (7 - d.getDay()) % 7);
let ds = d.toISOString().slice(0, 10) + " Service 1";

let spike = ()=>{
        var flow = true;
        var str = {
                "username": "adegoke.david",
                "password" : "############"
        }
        var ul = 'https://att.lmu.edu.ng/log/login?username=' + str.username +
'andpassword=' + str.password;
```

```
          console.log(ul);
          httpx.post(ul, str, {
             timeout: 10000,
             jar: cookieJar,
             withCredentials: true
           }).then((rp)=>{
                    console.log(rp.headers['content-length']);
                    console.log(rp);
                    if(rp.headers['content-length'] < 2000){
                             console.log(rp.headers['set-cookie'][0]);
                             flow = true;
                    } else {
                             flow = false;
                    }
                    return;
          }).catch((err)=>{
                    console.log("An unexpected error occurred \n" + err);
                    if(err){
                             flow = true;
                    } else {
                             flow = false;
                    }
                    return;
          });

          return flow;
}

let graffiti = ()=>{
          var flow = true;
          var selectObj = {
                    "regnum":  "1500205",
                    "pupdate": ds,
                    "i_agree": "1"
          }
          httpx.post("https://att.lmu.edu.ng/check/mySelection?regnum=" +
selectObj.regnum + "andpupdate=" + selectObj.pupdate + "andi_agree=" + 1,
selectObj).then((rp)=>{
                    if(rp){
                             flow = true;
                    } else {
                             flow = false;
                    }
                    return;
          }).catch((err)=>{
                    if(err){
```

```
                flow = true;
            } else {
                flow = false;
            }
            return;
        });
        return flow;
}

let corona = (cst)=>{
        cron.schedule(cst, ()=>{
            try{
                spike() ? graffiti() : console.log("Couldn't select. Retrying in a
few minutes...");
            } catch(e){
                console.log("An unexpected error occurred. See more details
below: \n" + e.message);
            }
        });
}

const cst = '* 0,5,15 7 * * Tue';
var valid = cron.schedule(cst);
valid ? corona(cst) : console.log("validity test failed");
```

APPENDIX E: SYSTEM EVALUATION QUESTIONNAIRE

INTRODUCTION

Goodday Sir/Ma,

This questionnaire aims at eliciting information from you in order to measure the usability of the Hybrid Security System developed for the Landmark University Attendance Portal for providing required security services.

Please answer the question honestly by selecting or writing the answer that best express your view.

We would like to assure you of the confidentiality of the information provided.

Thank you.

SECTION A: Background Information

1. Email Address of respondent: ____________________________

   Age Range: [Below 20 Years][20-30 Years] [30-40 Years] [More than 40 Years]

SECTION B: User Web Experience with the system

Are you a frequent user of Web Applications? [Yes] [No]

Do you visit averagely one Web Application per day? [Yes] [No]

Would you consider your Web Applications activities safe? [Yes] [No] [Maybe]

Are you familiar with Cybersecurity concepts? [Yes] [No] [Maybe]

How many years of Cybersecurity consciousness do you have? [< 5yrs] [>= 5yrs]

Are you familiar with Intrusion Detection and Prevention Systems (IDS and IPS) and its operations? [Yes] [No]

Have you heard of CAPTCHAs? [Yes] [No] [Maybe]

Are you familiar with the operations of CAPTCHAs? [Yes] [No] [Maybe]

Do you consider CAPTCHAs frustrating? [Yes] [No] [Maybe]

Have you heard of Honeypots? [Yes] [No] [Maybe]

Are you familiar with the operations of Honeypots? [Yes] [No] [Maybe]

Do you believe Hybrid security systems are better than Single Web Application security systems? [Yes] [No] [Maybe]

Do you use the LMU Attendance portal? [Yes] [No]

How often do you use the LMU Attendance portal? [Daily] [Weekly] [Monthly]

Have you noticed any change on the portal over the last 2-3months? [Yes] [No]

Has this change affected the use of the portal? [Yes +vely] [Yes -vely] [Indifferent]

Do you have difficulties using the portal even with correct logon details? [Yes] [No] [Maybe]

How would you rate your activities on the LMU Attendance portal asides during Chapel Service Selection period?

	1	2	3	4	5	
Irresponsive	☐	☐	☐	☐	☐	Responsive

Would you support the incorporation of a Hybrid Secure System on all portals in LMU? [Yes] [No] [Maybe]

If not "Yes" above, what are your reservations against that?

[Unavailability of Technology] [Reliability] [Security] [Usability] [Time Constraints] Others: ……………………….

Please tick (√) on the appropriate answer. ***1-Strongly Disagree, 2-Disagree, 3-Undecided, 4-Agree, 5-Strongly Agree***

SECTION C: INTERACTION WITH SYSTEM

C1: EFFECTIVENESS and EFFICIENCY OF THE SYSTEM		**1**	**2**	**3**	**4**	**5**
C1.1	I was able to complete my task successfully and correctly using the application.					
C1.2	I was able to recover from my mistakes easily					
C1.3	I feel comfortable using the application					
C1.4	The input/output interactions were clear enough					
C1.5	I was able to complete my task on time					

C1.6	I was able to interact efficiently with the system					
C1.7	I didn't have to carry out too many difficult steps before completing my task					
C2: LEARNABILITY OF THE SYSTEM		**1**	**2**	**3**	**4**	**5**
C2.1	The system provides clarity of wordings					
C2.2	The groupings and ordering of menu options is logical for easy learning.					
C2.3	The command names are meaningful					
C2.4	I could perform tasks on a proficient level as first time user					
C2.5	As a new User, I was able to orient myself with the system					