



Identification and Classification of Illegal Dark Web Activities in East Africa Region

By

Tariku Eshetu

Supervised By

Dr. Fitsum Assamnew

**School of Information Technology and Engineering
Addis Ababa Institute of Technology
Addis Ababa University**

A Thesis Submitted to the Department of the School of Information Technology and Engineering in Partial Fulfillment for the Degree of Master of Science in Cyber Security

Addis Ababa, Ethiopia

16 August, 2024

**Identification and Classification of illegal Dark Web activities in East
Africa Region**

By
Tariku Eshetu

Examiners' Committee

Name	Signature	Date
_____ (Thesis Advisor)	_____	_____
_____ (Dean of the School)	_____	_____
_____ (Internal Examiner)	_____	_____
_____ (External Examiner)	_____	_____

A Thesis Submitted in Partial Fulfillment of the Requirements
For Masters of Science in Cyber Security
Addis Ababa Institute of Technology
School of Information Technology and Engineering
16 August, 2024

Abstract

Online criminal activity manifests in various forms across the Surface, Deep, and Dark Web layers of the Internet. The darknet environment is notorious for various illegal activities, including financial crimes, hacking, recruitment for terrorism and extremism, child pornography, human organ trafficking, drug trafficking, and illegal arms trading. Law enforcement faces significant challenges in identifying specific criminal websites due to the ineffectiveness of traditional investigative techniques. In East Africa, the growth of technology has created economic and social opportunities, but it has also led to increased internet penetration and connectivity, making the region an attractive target for cybercriminals. Compounding the issue are the insufficient readiness of security organizations and a lack of user awareness, which further facilitate cybercrime. This thesis investigates the landscape of cybercrime on the Dark Web, focusing specifically on East African Internet Protocol (IP) address spaces, an area that has been largely under-researched in the existing literature. This research seeks to address a pronounced gap in knowledge regarding the types of illegal activities and associated protocols on the Dark Web, particularly given existing studies' inadequacies in contextualizing research within East African socio-political frameworks. The research pivots around two key questions: (1) What types of protocols operate through the Dark Web in East African IP address spaces? and (2) What illegal activities are conducted through these protocols?

The objectives of this study are multifaceted, aiming to develop a robust methodology for data collection and analysis from Tor exit nodes within the East African, classify the prevalent communication protocols, and categorize the diverse illegal activities identified. Through thorough examination of Tor network traffic, the study reveals crucial patterns, including a dominance of TCP and TLS protocols, smaller percentages using other protocols such as DATA, Bitcoin, HTTP, DNS, and SSH and with illicit activities significantly associated with drug , violence, and software piracy. The findings underscore the pressing need for tailored law enforcement strategies, informed policymaking, and collaborative regional approaches to manage the escalating threats. By innovatively integrating advanced data analytics techniques and multithreaded computing, this thesis provides a unique framework for ongoing cybercrime analysis, enhancing situational awareness for stakeholders and facilitating more effective monitoring of the Dark Web. The implications of this research extend beyond academic inquiry; it offers practical resources for law enforcement agencies, policymakers, and researchers in mitigating cyber threats, thereby contributing to a safer digital environment in East Africa.

Acknowledgment

I would like to convey my sincere appreciation and thanks to several individuals and organizations whose unwavering support and contributions have been instrumental in completing this thesis. I want to start by genuinely thanking my advisor, Dr. Fistum Asaminew, for his invaluable guidance, mentorship, and expertise throughout the research process. His insightful feedback, constructive criticism, and encouragement have been pivotal in shaping this thesis.

I am also grateful to the Information Network Security Administration (INSA) for providing me with a conducive research environment and facilitating the procurement of all necessary materials for my study. In this regard, Special thanks to Ato Daniel Guta, Deputy General Director of INSA, for understanding my research objectives, granting me permission to conduct the research, and allowing my organization to cover the financial costs associated with the study.

I would like to thank my friends Tekeste Fikadu and Munir Kemal for their invaluable insights, discussions, and moral support throughout the research process. Their collaboration and intellectual contributions have enriched the quality of this thesis.

I am also thankful to Adonay Teshome, Akrem Abdulkadir, and Eyobed Elias for their contributions while crafting scripts for my research. Their dedication, expertise, and support have been instrumental in successfully completing this study.

I am forever grateful to my loving wife for her unwavering support, understanding, and patience throughout this research endeavor. Her encouragement, love, and belief in me have constantly motivated and inspired me. Additionally, I would like to acknowledge the Student Graduate Committee (SGC) of the School of Information Technology and Engineering (SiTE) at the Addis Ababa Institute of Technology (AAIT) for their unwavering support and assistance throughout the thesis.

Finally, I would like to express my heartfelt gratitude to all individuals, organizations, colleagues, and friends who have supported me in any capacity during the course of this thesis. Your encouragement, assistance, and belief in my work have been inspiring and contributed immensely to this research's successful completion.

Thank you all for your unwavering support, guidance, and encouragement.

Dedication

To my two little princesses, you are my shining stars, my greatest blessings, and my reason for everything I do. Thank you for bringing so much joy, laughter, and love into my life. Despite the demands of this research, you both have been my source of inspiration, reminding me of the importance of balance and family. Even though you may not understand all the big words in this thesis, know that you both made every moment brighter with your laughter and cuddles. This thesis is for you, my sweet daughters, for being my little rays of sunshine and bringing so much joy into my life.

To my dear brother, Although you are no longer with us, your memory lives on in my heart and in everything I do. Your tragic loss serves as a reminder of the fragility of life and the importance of cherishing the moments we share with loved ones. This thesis is dedicated to you, my beloved brother, as a tribute to your life, your sacrifice, and the enduring impact you have had on me.

To my father, You have been my rock, my guiding light, and my greatest supporter. Your wisdom, encouragement, and belief in me have been a constant source of motivation throughout my academic journey. Your tireless dedication to ensuring that I reach my full potential has shaped the person I am today. This thesis is dedicated to you, my dear father, for always pushing me to strive for excellence, for instilling in me a passion for learning, and for being my unwavering source of strength.

This thesis is dedicated to my two sweet daughters, my dear brother, and my father, whose love, support, and presence have enriched my life in ways I could never fully express. Thank you for being my pillars of strength, my sources of inspiration, and my reasons for continuing to pursue my dreams.

Table of Contents

Abstract	iii
Acknowledgment	iv
Dedication	v
Table of Contents	vi
List of Figures	ix
List of Tables	xi
List of Abbreviations	xii
1 Introduction	1
1 Introduction	1
1.1 Background	1
1.2 Motivation	5
1.3 Statement of the problem	6
1.4 Research Question	8
1.5 Objectives	8
1.5.1 General Objectives	8
1.5.2 Specific Objectives	9
1.6 Contribution of the Study	9
1.7 Scope of the study	10
1.8 Structure of the document	10
2 Literature review	12
2 Literature review	12
2.1 Dark Web traffic identification and classification	12
2.2 Illicit activities on the Dark Web	18
2.3 Related Works	19
3 Methodology	27
3 Methodology	27
3.1 Research Design	28
3.2 Population and Sampling Design	29
3.3 Data Collection Methods	30
3.3.1 Environment Setup	30

3.4	Research Procedures	31
3.5	Data Analysis Methods	32
3.6	Ethical Considerations	33
4	Proposed Solution	35
4	Proposed Solution	35
4.1	Environment setup	35
4.2	Data collection	37
4.3	Data processing and Analysis	38
4.3.1	Metadata Analysis	38
4.3.2	Content Analysis	39
4.4	Web content classification	40
4.5	Visualization	42
4.5.1	Forwarding JSON Results to Elasticsearch	42
4.5.2	Forwarding MongoDB Data to Elasticsearch	43
5	Experiments and Analysis	44
5	Experiments and Analysis	44
6	Results and Discussion	49
6	Results and Discussion	49
6.1	Results	49
6.2	Discussion	59
7	Summary and Future work	65
7	Summary and Future work	65
7.1	Summary	65
7.2	Future work	67
	References	69
	A Random sampling script	75
	B Pcap Analyzer script	77
	C MongoDB to Elastic connector script	81
	D Cleaning unfinished JSON	85
	E Filtering unique destination IP script	87
	F Filter All African countries script	89
	G DNS resolution script	91
	H Content scrapping script	93

I	Scrapping success script	96
J	Scrapping Fialed script	97
K	Text matching script – main	98
L	Text matching script – preprocessing	100
M	Text matching script – TF-IDF Algorithm	102
N	Text matching script – Cosign Algorithm	104
O	Text matching script – Mini-LM Algorithm	107
P	Text matching script – merge result	109
Q	Shallalist – Scrapping script	111
R	Shallalist – merge script	114
S	Shallalist cleaner Script - for TF-IDF and Cosign	116
T	Shallalist cleaner Script – for Mini-LM	119
U	Elastic Connector	122
V	Forward result script to Elasticsearch	123
W	Configuration and installation	126
1	Wireshark installation guide	126
2	Configuring Tor Exit Node	127
3	Elasticsearch installation guide	130
4	Kibana installation guide	132
X	Shallalist web categories	133
Y	Random sample distribution	141

List of Figures

1.1	layers of internet [4]	3
4.1	Proposed Solution	36
5.1	Single thread Pcap analyzer script	44
5.2	Multi-threaded Pcap analyzer script	45
6.1	Exit relay	50
6.2	Exit policy	50
6.3	Exit relay property	50
6.4	Destination countries of randomly selected data	52
6.5	random data protocol types	52
6.6	Random data TLS versions	53
6.7	Unique Africa countries	53
6.8	Unique African Protocol types	53
6.9	unique African TLS versions	54
6.10	Scrapped destination countries	54
6.11	scrapped protocol types	55
6.12	scrapped TLS version	55
6.13	Failed scrape countries	55
6.14	Failed scrape protocol types	56

6.15 Failed scrape TLS versions	56
6.16 Failed scrape error logs	56
6.17 Cosign algorithm result	58
6.18 TF-IDF algorithm result	58
6.19 Mini-LM algorithm result	58
6.20 Mini-LM algorithm result with statistics	59
6.21 East Africa	60

List of Tables

2.1	Summary of related works	24
3.1	Tools Used	28
3.2	Specification	29
W.1	Wireshark installation guide	126
W.2	Tor exit node configuration guide	128
W.3	Elasticsearch configuration	130
W.4	Kibana Configuration	132
X.1	Shallalist web categories	133
Y.1	Sample data distribution	141

List of Abbreviations

TOR	The Onion Router
I2P	Invisible Internet Project
PII	Personally identifiable information
CTI	Cyber Threat Intelligence
URL	Uniform Resource Locators
IP	Internet Protocol
EVA - DSSM	Exploit Vulnerability Attention Deep Structured Semantic Model
DVSM	Device Vulnerability Severity Metric
SCADA	Supervisory Control and Data Acquisition
DL	Deep Learning
ML	Machine Learning
DT	Decision Tree
GB	Gradient Boosting
RFR	Random Forest Regressor
XGB	Extreme Gradient Boosting
CNN- LSTM	Convolution-Long Short-Term Memory
CNN-GRU	Convolution-Gradient Recurrent Unit
VPN	Virtual Private Network
ICQ	“I Seek You” Instant Messaging Application

AIM	AOL Instant Messenger
SMTPS	Simple Mail Transfer Protocol Secure
POP3S	Post Office Protocol 3 Secure
IMAPS	Internet Message Access Protocol Secure
FTP	File Transfer Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
VoIP	Voice over Internet Protocol

Chapter 1

Introduction

1 Introduction

1.1 Background

Artificial Intelligence (AI), Quantum Computing, Blockchain, Internet of Things(IoT), and other emerging technologies have the potential to significantly enhance human lives. In order to increase productivity and convenience, Internet of Things devices can automate processes in our homes and businesses. Blockchain transforms financial and supply chain management by providing a transparent and safe means of recording transactions. AI can produce novel solutions across a variety of fields and analyze enormous volumes of data. Still, new threats are also brought about by these same technology. Infrastructure disruption and the compromise of personal data are possible consequences of hacking into IoT devices. Blockchain may serve as a sanctuary for illegal activity due to its decentralized structure. Cyberattacks may be aided by AI systems. The potential for cracking existing encryption techniques, which puts internet security at risk, is another worry associated with quantum computing[1][2].

Since society relies on technologies, criminals have opportunities to act in ways that are against the law for various reasons, such as financial gain, political convictions, the desire to establish their reputation as authority figures, or the desire to cause harm or suffering. From a financial point of view, online crime is a massive business that makes

enormous revenues and fuels an economy where data is an illicit commodity. Due to the prevalence of this sort of crime, anyone wishing to enter the world of cybercrime to profit from it can choose from various illicit goods and services. Online criminal activity takes many diverse forms and occurs on the surface, deep, and Dark Web layers of the Internet [3].

The surface web, commonly referred to as the "visible web," is the "Internet," the web portion available to everyone. Information on the surface web is indexed and, therefore, searchable with standard web search engines. Its content can be accessed without any special configuration. The surface web is far smaller than the Deep Web, accounting for less than 10% of all content on the World Wide Web. Google, Bing, Yahoo, YouTube, Wikipedia, news sites, commerce sites, blogs, etc. are some examples of the surface web[4].

The Deep Web, also called the "invisible web," is the portion of the Internet that is invisible to everyone. Its material cannot be found using standard search engines. Deep Web websites typically need a Uniform Resource Locator (URL) to be found and additional authorization and permission to be accessed (i.e. a username and password). Email accounts, social networking sites, cloud service accounts, online banking, internal network of companies, government-related pages, educational pages, and material exchanged on private social networks are a few examples of the Deep Web[4].

The Dark Web is a minor portion of the Deep Web, also called the "dark net." In order to access the Dark Web, specialized software is needed (i.e. TOR, I2P, Freenet), which anonymizes the user's Internet Protocol (IP) address. Most websites are hidden on the Dark Web, where users can browse almost the internet anonymously. Within this web level, illegal markets known as 'dark net markets' operate, and many of them constitute a space for the illegal selling of products, including drugs, firearms, child sexual abuse material, hitman services, etc. Examples of dark net markets include Alphabay Market, Hansa Market, Dream Market, etc. The Dark Web could also be used for legal purposes, such as accessing and sharing information, communication, and identity protection [4]. Figure 1.1 illustrates the relationship between these layers of the Internet.

According to [5], the best-known special tool for accessing and surfing on the Dark Web is the Onion Router (TOR). TOR is a free tool that uses the onion routing technique to provide anonymity. It was originally developed to protect the classified data of the US

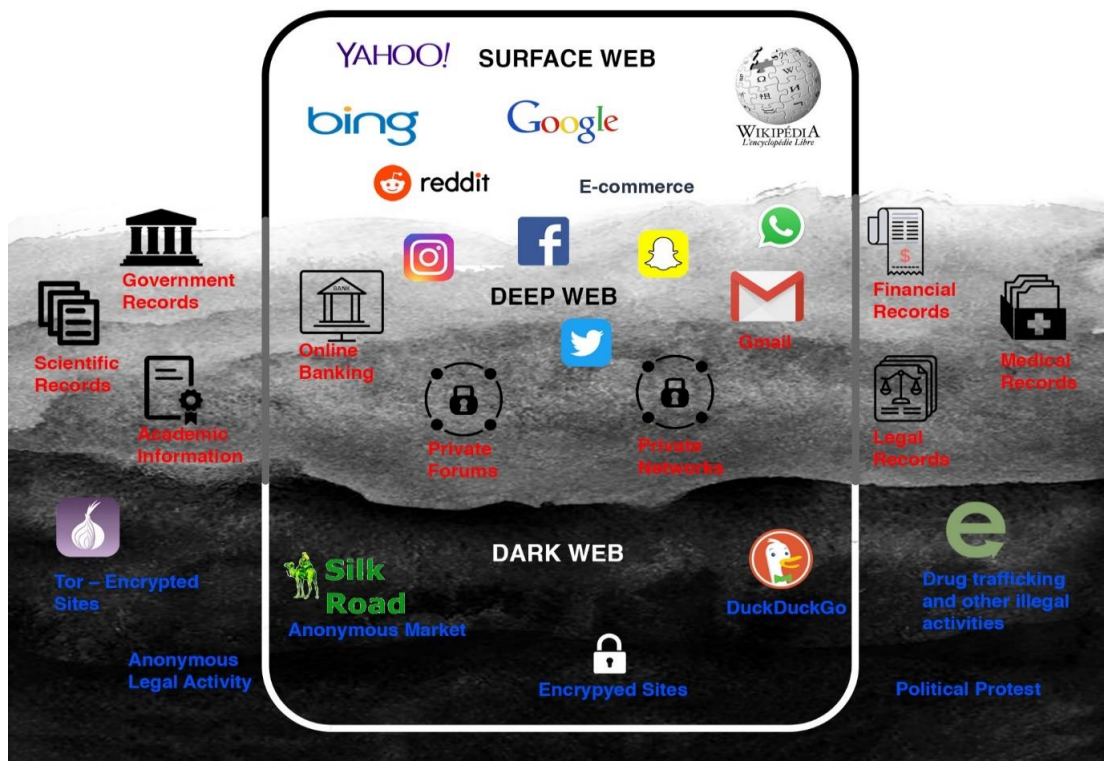


FIGURE 1.1: layers of internet [4]

Naval Research Laboratory, but it has since grown into an encryption tool for hiding users' activities and IP addresses. Criminals come together on the Dark Web using Tor platforms to perform illegal activities. Pornography, gun trafficking, illegal drug trade, fake documents, counterfeit currency, and terrorism cover 57% of Dark Web crime.

Terrorist activities can be carried out directly over the Dark Web, at the same time other Dark Web crimes can also aid in the spread of terrorism. Terrorist groups also widely conduct illegal actions, such as weapons trade, drug trafficking, human smuggling, and money laundering, to provide resources and finance for their organizations. Thus, other Dark Web crimes also become an element of terrorism. Dark Web terrorism is a worldwide problem that requires multilateral effort at the national, regional, and global levels [5]. Researchers and law enforcement have tended to focus on various illicit activities on the Dark Web to examine and determine the necessary precautions.

An overwhelming amount of criminal activity, including illegal financial transactions,

hacking, ransomware as a service, terrorism, recruiting for extremist groups, child pornography, human organ trafficking, drug trafficking, and the trade in illegal weapons, is carried out in the Darknet ecosystem. Law enforcement agencies find it extremely challenging to pinpoint specific websites that pose a security risk because conventional methods of investigation do not apply in this situation[6].

over the past decade, Africa has emerged as a significant hub for criminal activity. The continent has become a major route for drug shipments, a source of illegally obtained natural resources, and a starting point for human trafficking and migrant smuggling. Drug traffickers have developed sophisticated methods to disguise their products and evade detection, and many African countries lack the technology to identify dangerous substances like narcotics and weapons. The Internet has enabled online recreational drug markets to flourish, exacerbating the problem. To combat this issue, African nations need to invest in cutting-edge technologies and enhance border management law enforcement to prevent the entry of drugs and mitigate their impact on states and populations[7].

The dark web in Africa and African countries is active, offering a range of crime-related products and services, including malware, fake documents, stolen data, weapons, and drugs. The Middle East and North Africa underground market offers passport scans, identity documents, and drugs, while weapons are sold on English-language marketplaces catering to worldwide customers. A study found that the illegal trade of rhinoceros, tigers, and elephants occurs on the dark net, although it is not a major market compared to other commodities. High volumes of personal information from Nigerian and South African origins have been leaked on the dark web, targeting various companies and websites across industries[4].

The study [8] examines the role of Regional Economic Communities in combating human trafficking in the East African Community. It finds that human trafficking is prevalent in the region, with member states serving as source, transit, and destination countries. The study highlights political instability, poverty, and corruption as factors contributing to trafficking. Despite international and regional treaties, weak enforcement and criminal networks allow the crime to thrive. The study recommends strengthening the legal regime, adopting technology, and addressing root causes to combat human trafficking in the region.

Another study [9] concludes that the adoption of social media technologies like Facebook and Twitter by terrorist web publishers in Sub-Saharan Africa has been a significant factor in the rise of terrorist web presence in the region. While social media can give a voice to terrorists, it also allows ordinary citizens to counter their message. The study highlights the international nature of terrorism in Africa, with most terrorist websites hosted outside of the continent and often published in English. Al-Shabaab's use of social media during the Westgate mall attack in Nairobi is a prime example of the real-time, direct-to-target-audience propaganda that social media facilitates. The study concludes that there is a need for frameworks to compare metadata associated with different social media platforms and raises questions about the connections between terrorist web sites and group operations. The use of social media by terrorist groups during real-time operations highlights the need for further research on this topic, including the possibility that this trend may continue globally.

1.2 Motivation

The field of cybercrime is expanding quickly, and a lot of studies have concentrated on the destructive conduct that occurs on the Dark Web. Most websites are hidden inside the Dark Web, and users can browse online almost anonymously. Within this web level, illegal markets known as 'dark net markets' operate, and many of them constitute a space for the illegal selling of products, including drugs, firearms, child sexual abuse material, etc. The research community and law enforcement agencies greatly try to stop illegal activities online, but criminals constantly adapt and find new ways to accomplish their objectives [10]. Even though some research has investigated various types of illegal activities on the Dark Web, as criminals use new technologies such as the adoption of the Dark Web, the challenge to understand how they operate becomes more complex[11].

The dark web in Africa offers a range of crime-related products and services, including malware, fake documents, stolen data, weapons, and drugs[4]. The region is also plagued by political instability, poverty, and corruption, making it an attractive location for terrorist organizations to operate[8]. The use of social media technologies has enabled terrorist groups to spread their message and recruit new members[9]. Over the past decade, East Africa has become a significant hub for criminal activity, with human trafficking, and migrant smuggling being major concerns [7].

However, no research has been conducted on the Dark web to identify illegal activities in the East African region. Therefore, research on illegal activities in the East African region, specifically detecting Darknet traffic and hidden services on the Dark Web, is necessary to combat potential cyber threats in advance. Understanding how cybercriminals operate in the dark corners of the Web is crucial for developing specific and accurate interventions to prevent cybercrime and other illegal activities.

1.3 Statement of the problem

Cybercrime affects everyone, regardless of age or demographic, and can have serious consequences. Not only can it lead to personal harm, such as anxiety, depression, and trauma, but it also has a significant impact on the economy[12]. The impact of cybercrime is far-reaching and devastating, causing financial losses, reputational damage, and physical harm. Cybercrimes can also lead to identity theft and fraud, compromising sensitive information and causing long-term consequences for individuals' credit scores and financial security. Moreover, attacks on critical infrastructure can disrupt essential services, leading to loss of life and property damage. Furthermore, cybercrime can be used as a tool for political manipulation, propaganda, and social engineering, spreading fake news and disinformation campaigns that can cause confusion, fear, and division among people[13]. Research suggests that many cyber-attacks are linked to social, political, economic, and cultural conflicts, and understanding these factors can help predict, prevent, and trace attacks[14].

Like everywhere else in the globe, the region of East Africa has benefited from the growth of technology and digitalization in terms of economic, social, and political opportunities. Consequently, internet penetration, expansion of technology, and an increase in the number of Internet-connected devices in the region have shown the trend of creating a comfortable situation for Cyber Criminals. Moreover, the problem is made worse by the inadequate readiness of security organizations and user awareness in the region[15].

The Africa Cyber Surge II operation was launched in April 2023, coordinated by INTERPOL and AFRIPOL, was a four-month operation that aimed to identify cybercriminals and compromised infrastructure across 25 African countries [16]. But, the operation did not propose a framework for carrying out further activities in the area and did not focus

the detection method of illegal activities in cyberspace. Instead, it focused on investigating information provided by partner organizations Group-IB, Trend Micro, Kaspersky, and Coinbase.

One notable limitation of the studies [17], [10], and [18] is that, it does not offer a comprehensive classification of illegal activities occurring in specific countries. While the research provides valuable insights into the protocols employed within the Tor network and presents overall usage statistics, it lacks a contextual analysis of geographic variations in illicit activities. Furthermore, although the paper by [18] delves into particular marketplaces on the Dark Web, this focus does not extend to a broader understanding of the socio-economic or legal factors that might influence the prevalence and types of illegal activities in different national contexts. Consequently, the findings may overlook critical nuances that could provide a more holistic view of the implications and dynamics of the Dark Web across regions.

Additionally, the paper [19] only covers cybersecurity threats within a specific geographical location. However, given that the threat at hand concerns cross-border issues, conducting a regional analysis can offer valuable insights into the distinctive features of each country that could potentially affect security threats at the regional level. Moreover, some of the researches used commercial and proprietary tools that are not available to the researcher and all of the related studies didn't apply multithreaded or parallel computing to enhance performance in detection, identification and investigation illegal activities and also, handling large volume of data. Furthermore, all of the above research did not specifically focus on East African countries as part of their study.

Research on illegal activities, particularly cybercrime, in East Africa is limited, hindering law enforcement's understanding of local dynamics. Existing studies often overlook the unique socio-political and economic contexts of East African nations, failing to classify distinct cyber threats and their manifestations. Initiatives like the Africa Cyber Surge II did not establish a frameworks or detection methods, relying instead on external data sources. Specific studies focusing on particular countries or Dark Web marketplaces neglect a broader regional perspective, missing critical insights necessary for tailored strategies. Additionally, reliance on metrics and self-reported data raises concerns about accuracy and generalizability. These limitations underscore the need for comprehensive research to enhance understanding and effective responses to cybercrime in the region

focusing on the protocols employed on the Dark Web and the types of illicit activities conducted using these protocols.

Therefore, research on illegal activities on the Dark Web and understanding how cybercriminals operate in those regions are important for several reasons. First, such research can help to identify potential cyber threats in advance and allow for preventive measures to be put in place. Second, it can help develop specific and accurate interventions that address cybercrime and illegal activities in the region. Third, understanding how cybercriminals operate can help law enforcement and other agencies to track, identify, and apprehend individuals involved in such activities.

1.4 Research Question

Identifying protocols is essential for understanding the various communication styles in use and providing situational awareness regarding the services being utilized. Additionally, developing an effective framework to identify and classify illegal activities within regions is another important challenge, as the nature of these activities varies significantly from country to country according to their respective legal documents. This research aims to fill the gap in Dark Web knowledge by collecting, classifying, and analyzing the malicious activity on the Dark Web in the East African region. To this end, we define our research question;

1. RQ1: What types of protocols are passing through the Dark Web on East African internet protocol address space?
2. RQ2: What types of Dark Web illegal activities are conducted on those address spaces?

1.5 Objectives

1.5.1 General Objectives

The primary objective of this study is to analyze TOR network traffic to identify the prevalence and nature of illegal activities originating from East African IP address spaces on the Dark Web.

1.5.2 Specific Objectives

- Develop a method that gathers, collects, and parses data from a Tor exit node.
- Perform an analysis of outgoing Tor exit node traffic.
- Classify protocols passing through the Dark Web in the East African region.
- Categorize the illegal activities in each country of the region.

1.6 Contribution of the Study

The study's contributions can be summarized into two key areas. Firstly, this study makes significant contributions by detailing the protocol landscape, categorizing illicit activities, and developing a framework for ongoing cybercrime analysis specific to East Africa. Through rigorous analysis of Dark Web traffic originating from East African IP addresses, the research identifies and classifies various communication protocols in use, thus enhancing situational awareness regarding the technological frameworks that underpin illicit activities and enabling stakeholders to better understand how cybercriminals operate and adapt. Moreover, the study reveals a spectrum of illegal activities prevalent on the Dark Web, providing context-specific categorizations that enrich understanding of the unique challenges posed by cybercrime in the region, distinct from those elsewhere. In addressing a notable gap in existing frameworks for detecting and classifying these Dark Web activities, the research proposes a framework tailored to the East African context, which integrates multi-threaded computing and advanced data analytics techniques to enhance performance in detecting, identifying, and investigating illegal activities. More specifically, the framework's innovative features, including multi-threaded DNS resolution, data scraping, text matching, and elastic connector scripts, enhance the efficiency of analysis time, increase the handling of large data volumes, and facilitate real-time implementation. This framework is particularly valuable for academics and investigators, offering a practical solution for processing, analyzing, and visualizing darkweb traffic.

Secondly, the processed dataset collected using the configured TOR exit node provides a unique opportunity for researchers to conduct advanced studies, uncover new insights, and identify patterns that can inform and improve various fields of study. Finally, this

study addresses a significant gap in the literature, as no previous research has analyzed Dark Web traffic specifically in East African countries.

The findings of this study will provide law enforcement agencies with a conceptual and practical tool to analyze the use and misuse of the Dark Web, aiding their efforts to combat cybercrime in the East African region. By highlighting potential security issues, the study offers valuable knowledge to policymakers and researchers, supporting initiatives to protect society by addressing illegal activities on the Dark Web.

1.7 Scope of the study

This study acknowledges limitations due to the nature of Tor exit node data collection. The finite number of nodes available for configuration and geographical distribution can bias the data toward specific usage patterns and regions. In this case, the research focuses solely on traffic exiting through an Ethiopian node, excluding data from surrounding countries due to resource constraints. While increasing the number of exit nodes would mitigate this bias, it's a resource-intensive endeavor. To minimize these limitations, the study collected the dataset using big bandwidth size within the available means and addressed the potential bias when interpreting the results.

1.8 Structure of the document

The thesis is structured into nine chapters. The introduction (Chapter I) provides background information on the research topic and its significance. The literature review (Chapter II) provides a comprehensive review of existing research on the topic. The methodology (Chapter III) describes the research design and methodology used to collect and analyze data. The proposed solution (Chapter IV) discusses the proposed solution to solve the research problem effectively. The experimentation and analysis (Chapter V) part covers the experimentation and analysis of the study. The results and discussion (Chapter VI) present the findings of the study, interpret the results, and discuss their implications. The summary and future work (Chapter VIII) summarizes the key findings, and the future work will be conducted. Finally, the references list all

sources cited in the thesis, and the appendices provide source code that supports the research.

Chapter 2

Literature review

2 Literature review

The promise of anonymity on the Dark Web opens itself up to both legal and illegal activities (such as command-and-control endpoints for botnets, markets allowing zero-day exploits to be traded, hackers for terrorist organizations, civilians seeking protection from irresponsible corporations, militaries hosting hidden command and control services, journalists conducting operations in nations without access to free media and speech, and law enforcement performing sting operations). Due to the differences in the legal systems among countries, legitimate uses may be required to access the Dark Web in certain countries but not in others. Furthermore, individuals who wish to remain anonymous in the physical world can have a persona that represents them across multiple services online[20].

2.1 Dark Web traffic identification and classification

The work in [5] stated that one of the most important issues governments face globally is crime on the Dark Web. Due to the anonymity provided by the Dark Web and specialized technologies like TOR, it is challenging to identify criminals and follow their movements. As a result, it has developed into a platform for a variety of illicit acts, including terrorism as it relates to this paper, as well as the trafficking of weapons,

drugs, and pornographic material. Dark Web analyses are essential for creating effective counter terrorism policies. The study [5] objectives are to conduct a critical review of the literature and to show the research efforts in terrorism-related Dark Web studies. The study's findings show that there have only been a few scientific investigations on terrorist activity and the scientific methods used to detect and combat them on the Dark Web should be varied. Advanced artificial intelligence, image processing, and classification by using machine learning, natural language processing methods, and hash value analysis techniques can be used to detect and predict terrorist incidents on the Dark Web.

A thorough analysis of Tor traffic's characteristics is done to develop an inspection process that can enhance Tor detection. This methodology specifically takes into account the observer's position within the network as new inputs, the traffic segment it can monitor, and the unique features of the Tor browser to aid in the detection process. Additionally, a proof-of-concept for the suggested Tor detection method was created using a set of Snort rules [21].

A large-scale, longitudinal measuring study on anonymous market listings and posts will be used to profile the opioid supply chain in anonymous markets and forums. To achieve this, the authors of [22] this study suggest several methods for data collection, identification of opioid jargon terms used in anonymous marketplaces and forums, and profiling of opioid products, suppliers, and transactions. To gather information, they first performed a full site crawl of anonymous internet marketplaces and forums. Then, to identify information pertinent to opioid trading activities, they created a suite of opioid domain-specific text mining techniques (such as opioid jargon recognition and opioid trade information retrieval) (e.g., commodities, price, shipping information, suppliers, etc.). Following that, they carried out a thorough, extensive, longitudinal study to shed light on opioid trade activities that take place in secret forums and markets. According to [22] A total of 1,138,961 traces (i.e., threads of posts) from 6 underground forums and 248,359 listings from 10 anonymous online markets were collected. From 5147 distinct opioid suppliers' IDs and 2778 unique opioid purchasers' IDs, the investigation found 28,106 opioid product listings and 13,508 opioid-related promotional and review forum traces among them. In order to gain a better understanding of the underground trading activities involved in the global supply and demand of opioids, the study characterized opioid suppliers (e.g., activeness and cross-market activities), commodities (e.g., popular

items and their evolution), and transactions (e.g., origins and shipping destination) in anonymous marketplaces and forums[22].

Indicator- or behavior-based analysis are two techniques the Cybersecurity and Infrastructure Security Agency (CISA) mentions for identifying Tor activity in the organization [23]. The first one uses log analysis and lists of Tor exit nodes to identify the suspicious activity while the latter inspects patterns in TCP and UDP ports, DNS queries, and inspects the payload of the packets. White-box machine learning models like decision trees and Random Forest are used in this paper [23] as a different approach. The outcomes show that Tor traffic can be recognized using the TCP window size, frame size, and time-related traffic attributes. The study describes a machine learning methodology that uses decision trees C5.0 and Random Forest with a classifier that achieves accuracy levels above 95% to identify Tor network traffic.

The goal of another paper [24], which uses a public dataset made available in 2017, is to investigate the degree to which a particular anonymity tool (and the traffic it hides) may be discovered when compared to the traffic of other potential anonymity tools using five machine learning classifiers. The effects of feature importance and temporal-related features on the network are first examined with regard to flow-based TC. Furthermore, the role of finer-grained features is established, including the (joint) histogram of packet lengths (and inter-arrival times). The "early" TC of anonymous networks is examined one by one. Results [24] demonstrate that the three anonymity networks under consideration (Tor, I2P, and JonDonym) can be easily distinguished (with an accuracy of 99.87% and 99.80%, for flow-based and early-TC, respectively).

However, another paper [25] proposes a hierarchical classifier of Darknet traffic that can distinguish between 25 different Darknet users' behaviors and four different forms of Darknet (Tor, I2P, ZeroNet, and Freenet). The author of this work set up a Darknet data probe that can capture actual Darknet traffic in ZeroNet, Freenet, Tor, and I2P. After gathering and categorizing Darknet traffic, the researchers extract 26 time-based flow variables that can characterize the traffic's properties. Next train a hierarchical classifier made up of six local classifiers. Results demonstrate that the classifier can easily distinguish between the four different types of Darknet clients—Tor, I2P, ZeroNet, and Freenet—with an accuracy of 96.9% and can recognize eight different types of user behaviors for each kind of Darknet with an average accuracy of 91.6%. The outcome also

shows that the DL approach is ineffective when the feature extraction can accurately represent traffic characteristics [25].

Anonymous online browsing not only compromises integrity but also raises issues regarding the motivation behind such behavior. Network traffic should be categorized to avoid source and destination hiding together unless it is for benign activity. The paper [26] suggests several strategies for categorizing the Dark Web into various levels or hierarchies. The selection of features and the reduction of dimensionality are proposed using a variety of preprocessing techniques. The model is trained and tested using the Anon17 dataset. The article [26] suggests three classification levels depending on the network, traffic type, and application. The feature selection approach is suggested for effective anonymity detection using information gain value. Out of 82 properties, the proposed method only takes 28 into account. As a result, This reduces the required computation for classification and achieves better results even with the reduced features. As a future direction, the authors proposed to test the algorithm in real-time dataset.

The work of [27] analyze a cutting-edge complicated dataset that contains a lot of information about Darknet traffic and preprocess the data. Next, examine various feature selection techniques to choose the optimal features for detecting and classifying Darknet traffic. Apply fine-tuned machine learning (ML) algorithms, such as Extreme Gradient Boosting (XGB), Gradient Boosting (GB), Random Forest Regressor (RFR), and Decision Tree (DT), on a selected of features and compare the performance. Next, to more precisely identify network traffic, use modified Convolution-Long Short-Term Memory (CNN-LSTM) and Convolution-Gradient Recurrent Unit (CNN-GRU) deep learning algorithms. The results show that the suggested approach performs better than the existing approaches by producing the highest level of Darknet traffic detection accuracy, 96% and 89% of Darknet traffic categorization through XGB as a feature selection approach and CNN-LSTM as a recognition model [27].

And the [28] study evaluates Auxiliary-Classifer Generative Adversarial Networks (ACGAN), Random Forest (RF), Convolutional Neural Networks (CNN), and Support Vector Machines (SVM) for the classification of such traffic and the underlying application categories to improve Darknet traffic detection. RF was determined to be the most effective machine learning classifier for categorizing Darknet traffic for both traffic and application types. It outperformed state-of-the-art studies on the CIC-Darknet2020

dataset, yielding a 99.9% F1-score for traffic classification and a 92.3% F1-score for application classification [28].

Tor traffic is heavily used for harmful activities including quick port scans, hacking attempts, exfiltration of stolen credentials, etc. Because of this, intrusion detection and prevention systems should include Tor traffic detection. In order to detect and categorize encrypted Tor traffic, [29] proposes a deep neural network (DNN) based system. On the UNB-CIC Tor network dataset, the system classified Tor and non-Tor traffic with an accuracy of 99.89%. The accuracy of experiments [29] for classifying different forms of Tor traffic was 6.2% greater than that of earlier research on the same dataset. Additionally, adversarial samples produced by a Generative Adversarial Network are used to evaluate how robust the suggested DNN classifier is (GAN).

Another research [30] proposes a unique framework for managing networks and analyzing Darknet traffic to automate the identification of malicious intent in real-time, utilizing weight-agnostic neural network architecture. A suggested automated searching neural net architectural strategy that can carry out a variety of tasks, such as finding zero-day threats, is based on a weight agnostic neural networks (WANNs) methodology. The suggested solution [30] reduces the skill and effort barrier that hinders many organizations from properly securing their most important assets by automating the malicious intent detection procedure from the Darknet.

The [11] article suggests a novel method called DeepImage that uses a two-dimensional convolutional neural network to recognize and categorize Darknet traffic by using feature selection to extract the most significant information from the gray image. A Darknet dataset is produced by combining two encrypted traffic datasets. They decided on IS-CXVPN2016 [27] and ISCXTor2017 [31] based on the results of the dataset evaluation criteria. Both datasets captured regular, VPN and Tor traffic for seven diverse categories under respective applications: Browsing (Firefox and Chrome), Chat (ICQ, AIM, Skype, Facebook, and Hangouts), Email (SMTPS, POP3S, and IMAPS), File Transfer (Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla and an external service), Streaming (Vimeo and Youtube), VoIP (Facebook, Skype and Hangouts voice calls), and P2P (uTorrent and Transmission (BitTorrent)).to evaluate the proposed approach which successfully characterizes Darknet traffic with 86% accuracy [11].

The [32] paper summarizes the effort to systematically identify, collect, and monitor a total of 1,212,004,819 exposed PII records across both the Dark Web and surface web. 5.8 million SSNs, 845,000 credit/debit cards, and 1.2 billion account passwords were stolen as a result of the research. They located and gathered more than 1.3 million PII records of the victims whose PII was disclosed on the dark and surface web. Despite its research and practical value, jointly collecting PII from both sources is a non-trivial task. Since, they collected the largest academic collection of exposed PII, which, if properly anonymized, enables various privacy research inquiries, including assessing privacy risk [32].

According to the paper [33], Cyber threat intelligence (CTI) is becoming more and more popular among enterprises as a tool for helping them prioritize their risks, existing CTI methods are frequently criticized for only being proactive when it comes to known exploits. The huge and constantly changing Dark Web is one interesting data source that can aid in the development of proactive CTI. In order to create a novel Deep Learning (DL)-based Exploit Vulnerability Attention Deep Structured Semantic Model (EVA-DSSM), [33] adopted the computational design science paradigm. This model includes bidirectional processing and attention mechanisms that automatically link exploits from the Dark Web to vulnerabilities. Across four testbeds (web application, remote, local, and Denial of Service), 52,590 exploit-vulnerability linkages were used to extensively test the EVA-DSSM against the most advanced non-DL and DL-based techniques for short text matching. Additionally, a novel Device Vulnerability Severity Metric (DVSM) that incorporates exploit postdate and vulnerability severity was developed to assist cybersecurity professionals with their device prioritization and risk management efforts. According to the findings of these assessments, the proposed EVA-DSSM produces Precision at 1 score that is 20% to 41% higher than non-DL techniques and 4% to 10% higher than DL-based approaches. The research [33] also used two CTI case studies—openly accessible systems in the top eight US hospitals and more than 20,000 Supervisory Control and Data Acquisition (SCADA) systems worldwide—to illustrate the practical value of the EVA-DSSM and DVSM.

2.2 Illicit activities on the Dark Web

The surface web and hidden service versions of well-known websites on the surface web, such as Facebook (social media), ProPublica (news), DuckDuckGo (search engine), and other services that may be restricted locally or raise privacy/ad tracking concerns, are also frequently accessed by ordinary people using the Tor Darknet infrastructure. Chinese citizens use it to bypass the country's "great firewall of China," which blocks access to several popular services. Additionally, journalists use the Dark Web to write on material that is politically sensitive and can be illegal in their nation. Similar to this, whistleblowers may speak with journalists and provide private information so that it can be published online [20]

Additionally, the Dark Web served as an arena for conducting illegal activities; as an illustration, criminals utilize the Dark Web to market their assassination abilities. Websites including MailOnline, White Wolves, and C'thuthlu advertised hiring for criminals with salaries of \$10,000 in the US and \$12,000 in Europe [6]. The cost of hiring someone from a police officer to a high-ranking politician goes from 40,000 to 15,000,000. The Dark Web is also used for marketing, where all kinds of illegal goods are sold, including cigarettes, credit cards, pornography, stolen jewelry, and drugs. Anonymous marketplaces are one of the main sources for cyber criminals to sell cybercrime tools and leaked or stolen data on the Dark Web. Sheep Market forum and the Silk Road were the most known and rising Dark Net marketplaces and Silk Road was one of the successful black marketplaces that could earn over US\$1.2 billion through this. Following the path of the infamous Silk Road numerous crypto markets evolved. One of those is Bitcoin, a cryptocurrency that lets users of the Dark Net marketplace remain anonymous when transacting. Human trafficking, which is regarded as a serious violation of human rights, is another unlawful activity carried out on the Dark Web [6]. According to the UN Office on Drugs and Crime, some 2.5 million people are trapped in some form of modern-day slavery worldwide. The victims are forced into working as domestic workers, factory employees, sex workers, child soldiers, beggars, and laborers in other commercial businesses. Operation Pacifier, which involved 2,000,000 users, 23,000 explicit photos, and 9,000 explicit video files, saw the FBI arrest several hundred pedophiles from the US and other countries. Also, the Deep Web is a significant threat to national security because of terrorism and terrorist organizations [6]. The advantages of the Dark Web have been

used by terrorist groups like al-Qaeda and ISIL/ ISIS to fulfill their negative motives and disseminate propaganda. The Dark Web is a tool used by the Islamic State in Iraq and Syria (ISIS) to solicit money, support their cause, and transmit information during a change of command. The U.S. Military tracks ISIS on the Dark Web to do so, but the issue is that law enforcement and the military have not yet discovered a mechanism to do so without violating individuals' right to privacy [6] [34].

The paper [20] summarizes the illegal activities that take place on the Dark Web. As a market, the Dark Web is used to trade illegal substances, malware, exploits, credit card information, identities, and stolen information, as well as trade-in firearms and movies depicting child abuse. As a platform for communication, it offers chat rooms and forums for discussion. As a facilitator for cybercrime, malware-as-a-service for illicit purposes, Command-and-Control (C2) servers are set up as covert services and terrorism operations are carried out in concert with other activities. The Dark Web used Bitcoin over Tor to facilitate transactions for anonymity and cryptocurrency money laundering and used as a proxy to the Surface Web to get over bans and escape censorship as well as for protection from local authorities' persecution.

2.3 Related Works

The dark web in Africa and African countries is active, with various crime-related products and services available, including malware, fake documents, stolen data, weapons, and drugs. The Middle East and North Africa underground offers a range of products and services, including passport scans, identity documents, and drugs, which are sold on Arabic forums and English-language forums on the dark web. Weapons are also sold on English-language marketplaces that cater to customers worldwide. Additionally, the study found that the illegal trade of rhinoceros, tigers, and elephants is occurring on the dark net, but it is not a booming market compared to other commodities such as drugs, counterfeit documents, and child abuse material. High volumes of personal information from Nigerian and South African origins have been leaked on the dark web, targeting various companies and websites across various industries. Furthermore, a dark web crawler between 2016 and 2020 revealed that more than 90% of searched dark web domains related to Africa mentioned crime-specific keywords such as "porn", "drug", "credit", "murder", "hitman", "trafficking", and "smuggling". These keywords were

found on domains that specialize in one specific crime area or those that offer services and products related to multiple crime areas[4].

On the study [16], INTERPOL and AFRIPOL have successfully conducted the Africa Cyber Surge II operation across 25 African countries, resulting in the arrest of 14 alleged cybercriminals and the identification of over 20,000 suspicious cyber networks, two Dark-net sites linked to more than USD 40 million in financial losses. Launched in April 2023, the four-month operation aimed to enhance coordination among African law enforcement by facilitating communication and sharing intelligence, ultimately targeting cyber extortion, phishing, business email compromise, and online scams. Private sector partners such as Group-IB and Uppsala Security provided operational support, leveraging their intelligence resources to bolster efforts in disrupting cybercriminal networks.

Despite of [16] its achievements , the Africa Cyber Surge II operation did not establish a comprehensive framework for sustained action against cybercrime nor did it focus on developing detection methods for illegal activities in cyberspace. Instead, the operation relied heavily on the investigative information shared by partner organizations, including Group-IB, Trend Micro, Kaspersky, and Coinbase.

Using honeypots, specialized crawlers, and extraction tools, the study article [10] explores the methods used by cybercriminals on the Dark Web and the Surface Web to understand how criminality develops in various Web layers. Additionally, quantitative analysis is done to compare the two Web environments side by side. Three studies make up this thesis. To understand how cybercriminals use stolen credentials in the wild, the first study looks at how stolen account credentials were used after they were leaked in various places on the Surface and Dark Web. In the second study, malvertising is examined from the user's point of view to determine whether the likelihood of malware infection may vary according to the technology used to access the Web. The last study examines trading trends in both the Surface and Dark Web's underground forums to find discrepancies between them. To create policies and remedies to combat cybercrime more effectively, it is crucial to understand how criminals work in various Web layers, but this research goes beyond the typical surface-level examination of illegal activities on the Dark Web.

The goal of the thesis [17] is to examine intercepted Tor network traffic in order to ascertain what information is transmitted over the network. The gathered data is compared

with earlier research and aids in drawing conclusions about Tor usage. For a week, three Tor exit nodes are installed and running in the US, Germany, and Japan. With the help of the deployed packet sniffers, each data flow was thoroughly examined to determine factors such as the application protocol, the number of bytes sent in a flow, and the content type if the communication was sent in plain text. The [17] results show that 100.35 million flows were recorded, with 32.47% of them sending 4 or fewer packets in total. TLS accounted for 55.03% of all traffic and was the most used application protocol. 15.91% of people used HTTP, while 16% used an unidentified protocol (s). The US received more than 45% of all reported traffic, followed by the Netherlands, the United Kingdom, and Germany, each of which received fewer than 10%. 443 was the most utilized destination port at 49.5%, 5222 at 12.7%, 80 at 11.9%, and 25 at 9.3%. While the paper acknowledges the existence of illegal activities taking place on the Dark Web, it provides a broader view of the phenomenon by examining the technical aspects of Tor usage. By analyzing the protocols used, the paper sheds light on the network behavior and usage patterns of Tor in different countries. Based on these statistics, the paper draws conclusions about the level of adoption and popularity of Tor in different regions around the world.

However, the paper [17] does not provide a comprehensive classification of illegal activities occurring in specific countries that utilize the Tor network, thereby limiting its contributions to understanding the localized dynamics of cybercrime. Instead the focus is primarily on analyzing the protocols employed and the overall usage statistics of Tor within those countries. This lack of granularity means that critical insights into the contexts that influence the prevalence and nature of cybercriminal activities remain unaddressed. Without this contextual classification, it becomes challenging for policymakers, researchers, and law enforcement agencies to fully grasp the unique challenges and threats posed by cybercrime in specific national contexts, which may differ significantly due to variations in legal frameworks, enforcement capabilities, and available resources for combating such activities. Consequently, the findings may overlook essential nuances necessary for developing targeted interventions and responsive strategies tailored to the distinct realities of each country's cybercriminal landscape.

On the Dark Web, illicit behavior can be effectively classified and visualized using the method provided in this [35] study. They trained the classifiers and used creative selection to determine the laws and rules related to each category of criminal activity. Remove

all of the unnecessary data from the categories of narcotics, gamblers, firearms, child pornography, and counterfeit credit cards before utilizing TF-IDF (Term Frequency - In- versa Document Frequency) as a feature extraction technique to generate a vector space model. This makes it simple to add more categories for classification. To begin the crawling and collect the textual data for this study, the Spyder crawler was utilized to access an illicit Dark Web corpus using the Tor network and later, used SVM (Support Vector Machine) and Naïve Bayes algorithm for classification on which we achieved performance with good accuracy. Despite the increasing use of machine learning and deep learning algorithms in detecting and classifying illegal activities on the Dark Web, there is still some controversy surrounding the results produced by these methods.

Previous research has shown conflicting outcomes regarding the effectiveness of different algorithms for detecting and classifying illicit activities on the Dark Web. While some studies have reported better results using machine learning algorithms, others have found that deep learning approaches are more accurate in identifying and categorizing such activities. However, there are still gaps in our understanding of which algorithms perform best in detecting and classifying Dark Web activities accurately. Therefore, the controversial nature of the results produced by different algorithms suggests that further research is necessary to refine and improve the current detection methods.

The National Research Foundation of South Africa, BankSeta, and the Council on Scientific and Industrial Research provided financial support for this [19] study to trace the nefarious usage of the Dark Web through the South African Internet protocol address space. The method of Dark Web crawling utilizing an onion router is used in the study to monitor traffic with a higher tendency for cybersecurity threats. The crawling experiment's findings show that child pornography, spyware sales, hacking, drug sales, violent crime planning, and the selling of dangerous weapons are among South Africa's most common Dark Web abuses. The findings of this study can aid in the accurate disclosure of cybersecurity concerns, assisting law enforcement agencies in their fight against cybercriminals in the nation.

As limitation, the paper [19] only covers cyber security threats and focuses on specific geolocation, i.e., South Africa. Even though understanding the impact of specific cybersecurity threats is essential to developing effective solutions that can ultimately benefit

society, understanding threats within specific locations is crucial. But, examining the issue from a regional perspective can provide valuable insights into unique characteristics of different countries that may impact security threats at the regional level. Analyzing the threat at regional level fostering more effective cross-border cooperation and enhancing overall cybersecurity resilience in the region. Furthermore, this approach can help in crafting region-specific policies and interventions that take into account the diverse motivations and operational methods of cybercriminals, ultimately leading to a more nuanced understanding of the cyber threat landscape in the region.

Additionally, [18] research offers an analytical framework for automating Dark Web scraping and analysis using free web-based technologies and successfully tested a Web crawler created with AppleScript to extract the account information for thousands of vendors and their related marketplace listings using case study marketplace. This study explains why scraping Dark Web marketplaces with AppleScript was the most practical and effective technique. The findings of this case study confirm the effectiveness of the suggested analytical framework, which is important for researchers looking at criminal activities on the Dark Web as well as academics researching this expanding phenomenon [18].

The research by [18] has limitations, the primary focus of this research is on a particular marketplace on the Dark Web, with the purpose of validating the effectiveness of the tool or the research intends to examine the targeted Dark Web marketplace in detail to verify certain assumptions or hypotheses. Furthermore, it does not aim to classify the types of threats that might be present in specific countries. so, the research often neglects to classify the specific types of threats present in individual countries, which is essential for understanding localized cybercriminal activities. This lack of contextual analysis limits the ability of researchers and law enforcement to develop effective, tailored strategies for combating cyber threats in specific locales, thereby hampering efforts to predict emerging trends and implement targeted interventions.

The [36] research aims to identify the overall geographic distribution of visitor groups into time zones. The plan is to create profiles of frequent visitors by taking advantage of the duration of every post in the Dark Web forums. Then, by comparing the crowd's profile to that of users from well-known places on ordinary web platforms, it will be possible to determine the crowd's geographic origin. The paper [36] also evaluates the

methodology's efficacy on two known-origin Dark Web platforms and the ordinary web and applies it to three controversial anonymous Dark Web forums. Through a sociological lens, this study aids in the community's understanding of the Dark Web, which also supports government investigations when people's safety is at risk.

The research by [36] has limitations, including its reliance on user behavior metrics like post duration, which may not accurately reflect true engagement or intentions. The study's findings are based on a limited number of known-origin Dark Web platforms, raising concerns about generalizability to other sites with different user dynamics.

Another study [37] employed a descriptive survey design and 400 participants were chosen using a stratified random sample technique. The "Dark Web and Cybercrime Threat Mitigation Questionnaire (DWCTMQ)" was the instrument used to collect the data and was given to the respondents. Before the reliability test with 40 computer experts who were not a part of the main work was conducted, the instrument was reviewed by experts in computer science as well as test and measurement, and it produced an average reliability coefficient of 0.87, proving the instrument to be reliable for the study. Percentage analysis, mean statistics, and chi-square analysis for the goodness of fit test are techniques that data were analyzed. According to the data analysis's findings, Nigeria has a significant amount of Dark Web and cybercrime activity. Additionally,[37] it was noted that cybercrimes and Dark Web threats in Nigeria include data theft, havoc on individuals and organizations by criminals, advanced persistent threats, distributed denial of service attacks, botnets, destructive malware, and the escalating problem of ransomware, etc.

The study by [37] has limitations: it uses a descriptive survey design, which may lack depth compared to qualitative approaches, and relies on self-reported data that may introduce biases. These limitations highlight the need for more comprehensive studies to understand the dynamics of Dark Web activities.

A comparison analysis of the selected related works based on methodology, contribution, weakness, and strength is summarized in Table 2.1.

TABLE 2.1: Summary of related works

Paper	Methodology	Contribution area	Weakness	Strength
[19]	Dark Web Crawling and Analysis	Dark Web cyber-security threats through the South African IP address	Only covers cyberse- curity threats and fo- cuses on specific ge- olocation i.e. South Africa	Accurate disclosure of cyber security concerns and threats
[18]	Dark Web scraping and analysis	Automated Analyti- cal framework (Us- ing Applescript)	Limited to analyze specific marketplace (for validation pur- poses)	The proposed method- ology to measure the ef- fectiveness of the sug- gested analytical frame- work
[11]	Deep learning algorithm	Identify and classify Darknet traffic	The accuracy and it's not tested on real- time data	The Darknet dataset is produced by combining two traffic datasets.
[17]	Web crawler and traffic analysis	Protocol classifica- tion	The Paper didn't classify the illegal activities in targeted countries	Collect large numbers of traffic Using three different countries' exit node
[10]	Honeypots, specialized crawlers, and extraction tools	Difference in mali- cious activity on Sur- face Web and Dark Web	The study does not classify which threats are present in which country	Use a hybrid method for collection
[37]	Descriptive survey design	Dark Web and cy- bercrime activities in Nigeria	Relies on self- reported data that may introduce biases	Large Number of par- ticipants involved in the data collection

[36]	Pattern and behavioral analysis	Identify the overall darkweb geographic distribution of visitor groups into time zones	Reliance on user behavior metrics like post duration, which may not accurately reflect true engagement or intentions	Use creative way to identify darkweb users origin country
------	---------------------------------	--	--	---

In conclusion, Despite efforts like the Africa Cyber Surge II operation, there remains a critical need for a efficient framework to systematically identify and classify illegal activities in cyberspace, as these vary significantly by country. Existing literature on cybercrime lacks region-specific analysis, highlighting the need for deeper contextual understanding. Research that focuses solely on individual countries or specific Dark Web marketplaces often overlooks the broader regional context, resulting in missed critical insights needed for developing tailored strategies. Furthermore, the dependence on metrics and self-reported data gives rise to concerns regarding accuracy and the generalizability of the findings. To address this gap in knowledge, my research will focus specifically on gathering comprehensive information about the illegal activities prevalent in East Africa. This will provide critical insights to law enforcement agencies, enabling them to take more effective measures to mitigate the risks associated with these activities.

Chapter 3

Methodology

3 Methodology

The methodology of this research focuses on developing a robust framework for identifying and categorizing network traffic associated with the dark web. To achieve this, a dedicated server was established, equipped with the necessary software and tools for data collection and analysis. Approximately 5 terabytes of network traffic data were gathered, consisting of around 20,304 individual files. A random sampling strategy was employed to ensure a representative subset of data was analyzed, while also acknowledging the potential biases and limitations inherent in this sampling approach.

A secure environment for data collection was created by configuring a high-speed internet connection (100Mbps) and a dedicated public IP address, in collaboration with network administrators to enhance security. An Ubuntu virtual machine was set up using VMware ESXi, where essential tools, including Wireshark, were installed for effective network traffic monitoring. The server was specifically configured to act as a gateway for accessing dark web content, ensuring efficient data retrieval and analysis without compromising security.

During the data analysis phase, the collected information was processed to generate manageable files, focusing on identifying unique addresses to classify successful and unsuccessful connections. A scraping tool was employed to gather additional data based

on successful connections, and any errors encountered during this process were meticulously logged. The insights derived from the analysis were integrated into a centralized system, visualized to enhance the understanding of network and content behaviors, and subjected to comparative analysis using text matching techniques to uncover correlations across datasets.

The following sub section of this chapter further expounds on the research design, population and sampling design, data collection methods, data analysis methods, and research procedures.

3.1 Research Design

The research design is the procedures and measures undertaken in research, including experimental and exploratory research methods. The choice of both experimental and exploratory research methods is justified as they offer complementary strengths for analyzing complex network traffic data and developing an effective framework for traffic identification and classification. Exploratory research provides valuable insights into intricate network behaviors and helps identify key variables, while experimental research allows for controlled manipulation of these variables. A well-managed and controlled environment is significant for the investigation carried out, so setting up a controlled environment to capture and analyze network traffic data using various tools and scripts is useful to propose a framework for traffic identification and classification. Together, they create an iterative process where initial findings inform rigorous testing, leading to a comprehensive and practical framework that effectively addresses the challenges of network traffic analysis.

The environment used in this research was set up using the tools illustrated and explained in Table 3.1.

TABLE 3.1: Tools Used

No.	Tools	Function of the tools
1.	ESXi-7.0U3n-21930508-standard (VMware, Inc.)	For the management purpose (hosted system)

2.	Ubuntu 22.04	Platform to conduct the experimental tests.
3.	Wireshark	To collect the traffic from the tor exit node
4.	Mongo DB	To store all the necessary filtered data in structure format
5.	ElasticSearch	To index the data for later use
6.	Kibana	To visualize all the intermediate and final results

A server used for the data capturing and analysis had the properties highlighted in Table 3.2 below and was used for data accusation and analysis of the acquired data. This server had a fairly good capacity for processing, which allowed the running of the required tools to achieve the study objectives.

TABLE 3.2: Specification

No.	Software/Hardware	Features
1.	Processor	Intel(R) Xeon®, CPU core 8, CPU E5-2640 v4 @ 2.40GHz
2.	RAM	43 GB
3.	Storage	5.5TB
4.	Operating system	Ubuntu 22.04 (64-bit)

3.2 Population and Sampling Design

During the research study's data collection, a substantial volume of data was amassed, totaling approximately 5 terabytes. This data was stored in the form of around 20,304 PCAP files, each 250 MB in size. The sheer magnitude of the collected data presented a challenge in managing, analyzing, and processing the data efficiently. A sampling

strategy was employed to address this challenge and facilitate the analysis of the data. Specifically, a random sampling technique was utilized to select a representative sample from the vast dataset. The decision to opt for random sampling was grounded in the need to ensure the selection of a diverse and unbiased subset of data for analysis and upholds the integrity of the statistical analysis, ultimately leading to more reliable and applicable conclusions.

This sampling approach allowed for the creation of a manageable subset of data that could be more feasibly manipulated, analyzed, and interpreted within the constraints of the research study.

In defining the inclusion criteria for the sample, the random sampling technique ensured that data points were selected without any predetermined bias or preference. This approach aimed to capture a representative cross-section of the original dataset, thus enhancing the generalizability of the findings derived from the sample.

Furthermore, the exclusion criteria for the sample were not explicitly defined as the random sampling methodology inherently excluded any deliberate selection or exclusion of specific data points. Instead, the focus was on ensuring a random and unbiased selection process to yield a sample reflective of the overall dataset.

3.3 Data Collection Methods

3.3.1 Environment Setup

To ensure a secure and controlled environment for data collection from the darkweb, several preparatory steps were taken.

- **Internet connection:** Recognizing the importance of high-speed data acquisition, I diligently pursued the procurement of a 100Mbps bandwidth connection to ensure efficient and timely data collection.
- **Setting up hosting environment:** To collect data securely, a dedicated public IP address was obtained for hosting purposes. The appropriate security measures to protect the hosting environment from potential threats were configured by

closely collaborating with our organization's network administrators. The hosting environment was effectively established by getting direct fiber cable to my server, ensuring the safety of the collected data, and making sure that the configuration did not disturb or make the organization vulnerable.

- **Installation of software :** The process involves setting up a server with VMware ESXi as the hypervisor and creating a virtual machine running Ubuntu as the guest operating system. This setup enables multiple virtual machines to run simultaneously with different operating systems for increased flexibility and resource management.

To configure an ESXi server with two separate networks, one for management and the other for a virtual machine with a public IP, steps involve creating vSwitches and network adapters. The process includes logging into the ESXi Host, creating vSwitches for Management and Public IP, adding a network adapter to the VM, configuring IPs, and accessing the ESXi Host and Virtual Machine accordingly.

The Ubuntu operating system was selected as the guest OS, and resources like CPU, memory, and storage were allocated, and network settings were configured. The Ubuntu ISO file was mounted to proceed with the installation. Also, to capture the data, a protocol analysis software called Wireshark was installed on top of the Ubuntu virtual machine running on the ESXi host. The steps taken for Wireshark installation are shown in Table W.1. Additionally, a Tor exit node was configured as part of a process aimed at developing a method to gather, collect, and parse data from this node. This configuration was successfully achieved using the resources available from the Tor Project community, in alignment with the research objectives as shown W.2.

3.4 Research Procedures

The research procedure involves setting up a server running on the Ubuntu operating system with specialized functional capabilities, such as memory and disk capacity, on top of ESXI for efficient management. The server is further configured as a Tor exit node with a dedicated server having a 100Mbps internet bandwidth to ensure smooth operation.

To guarantee the effective configuration of the Tor exit node, detailed prove can be accessed at [38]. Additionally, essential software tools are installed on the server, including Wireshark for packet capturing to monitor the traffic passing through the configured exit node and MongoDB for storing filtered data in a user-friendly format.

A random sampling script is employed to filter and select representative data for subsequent analysis. Metadata is extracted from the sampled data using a custom packet analyzer script to provide insights for further processing. These intermediate results are then transferred to ElasticSearch by utilizing an Elastic connector script and visualized through Kibana, enhancing the data visualization process.

3.5 Data Analysis Methods

Furthermore, the filtered data using the custom packet analyzer script is exported in JSON format, allowing for additional investigation, and various investigative avenues are opened. This includes the extraction of unique destination IP addresses in different scopes of workflows using a custom script that is capable of cleaning the exported JSON file and filtering unique destination IPs in the world and in Africa. The subsequent identification of successful and unsuccessful destination IPs through the use of a Custom DNS resolution script. These findings are then transmitted back to ElasticSearch for more thorough analysis and visualization through Kibana to gain a deeper understanding of the data.

Moving forward, A custom Scraper script is activated utilizing the successful DNS results as its input, scraping the successful and logged the failed results with its error status code.

In conjunction with the data collection task mentioned above, we download Shallalist labeled web content categories from the official website [39]. Shallalist is a widely recognized and respected web content categorization tool that assigns labels to different categories of web content. This categorization system is crucial for various purposes, such as content filtering, parental controls, and network security. By accurately labeling web content categories, Shallalist enables users to control and manage the types of content that can be accessed on their networks or devices. This helps ensure a safe and productive online environment for individuals and organizations[19].

The labeled web content categories provided by Shallalist cover a wide range of topics and themes, including but not limited to adult content, gambling, violence, and hate speech as shown in [X.1](#). Each category is assigned a specific label based on the nature and subject matter of the content, allowing users to easily identify and filter out undesired or inappropriate material. This categorization system helps users make informed decisions about the type of content they want to allow or block within their network, ensuring a tailored and secure browsing experience. Moreover, Shallalist's web content categorization system is continually updated and maintained to reflect changes in online content and trends. This ensures that users have access to accurate and up-to-date information about web content categories, enabling them to make informed decisions about content filtering and access control. The comprehensive and regularly updated nature of Shallalist's categorization system enhances its effectiveness as a tool for managing and controlling web content, making it a valuable resource for individuals and organizations seeking to create a safe and secure online environment.

Finally, the labeled shallalist data is compared with the scraped content (collected using the Wireshark packet capturing tool) through a text matching algorithm script designed to identify content patterns. The resulting data is then sent to Elasticsearch for storage and visualized through Kibana to analyze the research findings comprehensively.

3.6 Ethical Considerations

When configuring a Tor exit node and capturing traffic from it for analysis, several ethical considerations had to be taken into account to ensure responsible and lawful use of the data collected. One important aspect of protecting user privacy was addressed during the configuration phase, where measures were implemented to safeguard both Tor users and the organization running the exit node. For instance, a separate network directly from the service provider was used to help protect both parties from potential cyber attacks and mitigate risks associated with operating the exit node.

In the context of traffic capture and data analysis, specific solutions were implemented to address ethical concerns and protect user privacy. For example, As per described in Pcap Analyzer part, metadata information was effectively used in research endeavors to anonymize and protect the privacy of Tor users while still allowing for valuable insights

to be gleaned from the data. Additionally, a dedicated server was employed to strengthen privacy and security safeguards, lowering the possibility of data breaches or illegal access to sensitive user data. By using dedicated server infrastructure, the potential vulnerabilities associated with data transportation between machines were minimized, thereby protecting the integrity and confidentiality of user data.

To enhance security and mitigate risks associated with operating an exit node, a dedicated network was established that functions independently from the service provider's standard infrastructure. This network is supplied by a direct fiber optic connection from Ethio Telecom, offering a fast and reliable link that minimizes latency and enhances performance for users. The use of fiber cable ensures high bandwidth availability and resilience against potential bottlenecks that could arise with shared connections, thereby contributing to the overall robustness of the network. Furthermore, lowering the possibility of data breaches or illegal access to sensitive user data. By using dedicated server infrastructure, the potential vulnerabilities associated with data transportation between machines were minimized, thereby protecting the integrity and confidentiality of user data. In addition to the infrastructure components, the configuration of the exit node adheres to the best practices outlined by the Tor community. By implementing these recommendations, which include traffic encryption, proper handling of exit traffic, and rigorous logging policies, we can maintain an effective operation of the exit node without compromising users' anonymity.

Overall, by incorporating these solutions into the Tor exit node configuration, traffic capture, and data analysis processes, researchers and organizations were able to uphold ethical standards, protect user privacy, and ensure the responsible and transparent handling of data collected from the Tor network. These measures promoted ethical conduct in research and data analysis and contributed to maintaining the trust and integrity of the Tor network as a platform for secure and anonymous communication.

Chapter 4

Proposed Solution

4 Proposed Solution

The research utilized flowcharts to visually depict the sequence of decisions and steps within the proposed framework as shown in figure [4.1](#). The theoretical aspect of the solution involved proposing a framework for data processing, analysis, and interpretation, including the development of custom scripts for data filtering and packet analysis. In practical terms, systems and tools were developed to implement the proposed solution effectively. This included setting up a dedicated server with specialized capabilities as a Tor exit node for data collection and analysis. Software tools such as Wireshark, MongoDB, and Elasticsearch were employed for efficient data processing and storage. A DNS resolution script was introduced to identify successful and unsuccessful destination IPs to improving the understanding of network traffic patterns. Additionally, while a Scraper module facilitated data extraction and a text matching module was developed to classify the content of the page by comparing the pre-labeled dataset using text comparison algorithms.

4.1 Environment setup

To establish a secure hosting environment for data collection, a dedicated public IP address was obtained, and appropriate security measures were configured in collaboration

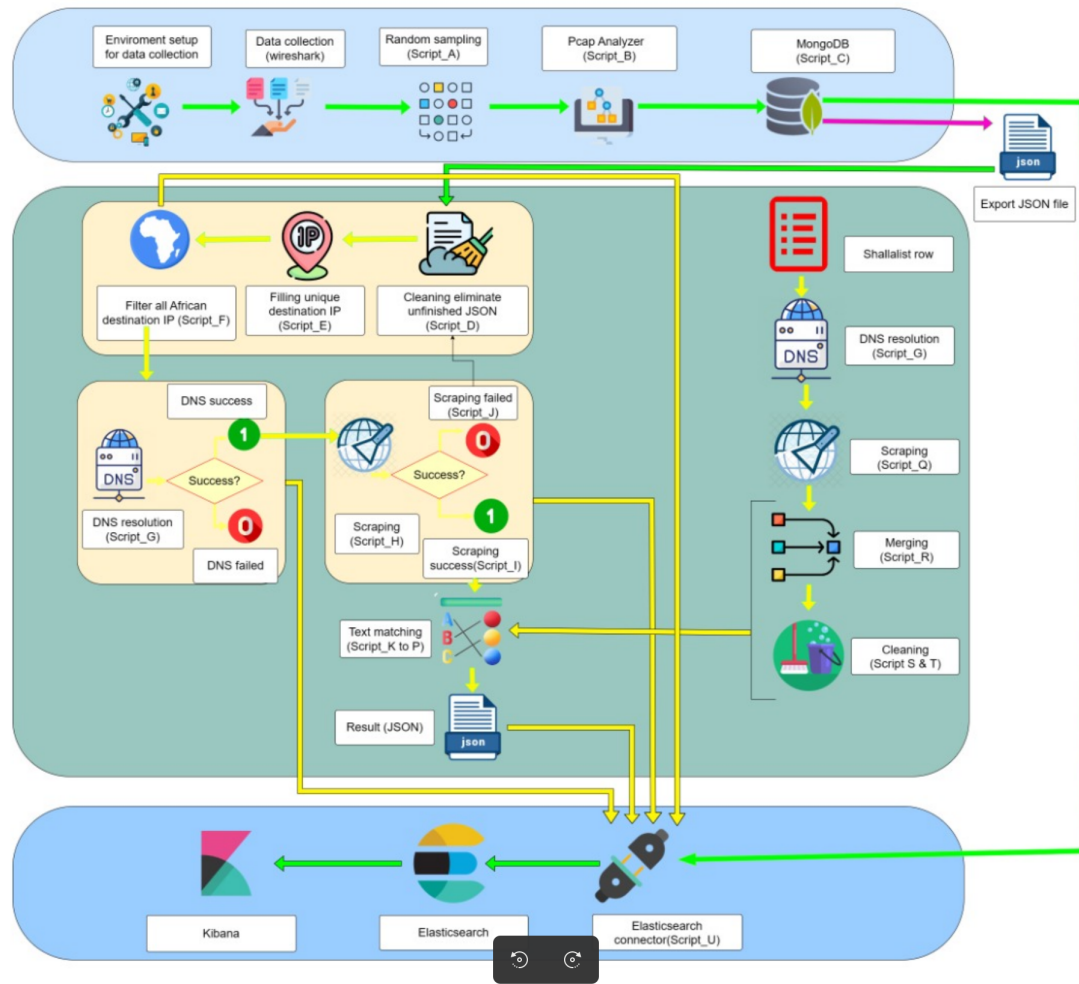


FIGURE 4.1: Proposed Solution

with organization network administrators. The hosting environment was effectively established by connecting the server directly to a fiber optic cable, ensuring data safety and network stability. This setup aimed to protect the collected data and prevent vulnerabilities in the organizational network. The hosting setup also involved installing software by creating a server with VMware ESXi as the hypervisor and deploying a virtual machine running Ubuntu as the guest operating system. This allowed for the simultaneous operation of multiple virtual machines with different operating systems, enhancing flexibility and resource management. Configuration steps included setting up separate networks for management and the virtual machine with a public IP, creating vSwitches, adding network adapters, configuring IPs, and finalizing setup details for ESXi Host and Virtual Machine access. Within the environment setup phase, the configuration of a Tor exit node was also accomplished to gather and parse data effectively in alignment with research goals. This process involved installing Tor software on

the server, editing the Tor configuration file to enable exit node functionality, configuring port forwarding for Tor traffic, ensuring a static IP address for reliability, setting exit policies to manage traffic types, and prioritizing security considerations for server protection.

The rationale for establishing a Tor exit node for the purpose of data collection from the dark web includes the following; First, it enables us to access hidden services and content that are otherwise inaccessible through conventional internet connections. Furthermore, it enhances our anonymity, allowing for the collection and analysis of traffic patterns and user behaviors within dark web communities without revealing our identities. Additionally, operating an exit node contributes to the overall integrity and stability of the Tor network, thereby facilitating ethical data collection in accordance with legal considerations surrounding internet privacy. By managing our own exit node, we can also circumvent common issues related to exit node blocking, ensuring more reliable access to dark web resources.

4.2 Data collection

In the data collection process using Wireshark, packets are captured and analyzed to monitor network activity in real time or to analyze data retrospectively. By installing Wireshark on a virtual machine running Ubuntu within the hosting environment, data packets can be intercepted, inspected, and recorded for further analysis. Wireshark allows for the detailed examination of network traffic, including protocols, source and destination IP addresses, ports, and even the content of the transmitted data. To begin data collection using Wireshark, the software must be launched on the Ubuntu virtual machine, and the network interface must be selected to capture packets. By applying filters, specific types of data can be targeted for recording, enhancing efficiency in data collection. Once data packets are captured, they can be saved in different files for further analysis or shared with other tools for in-depth investigation. Additionally, Wireshark provides various features for packet inspection, including protocol analysis, flow visualization, and the capacity to spot irregularities or trends in network flow. The software's user-friendly interface, powerful packet analysis capabilities, and compatibility with multiple network protocols make it adaptable to data collection and analysis in research projects or security monitoring activities.

A script was developed to systematically extract a representative sample comprising 5% of the total data from each PCAP file randomly as shown in Appendix A. The rationale for selecting random sampling lies in its ability to minimize bias and ensure that every data point has an equal chance of being included, which enhances the generalizability of the findings. The script initiates by iterating through a pre-defined list of Wireshark folder names, verifying the existence of each folder, and subsequently analyzing the contained files. It categorizes these files based on the date embedded in their filenames, tracking the count of files corresponding to each date. Finally, it efficiently transfers 5% of the files from each date group, thereby creating a manageable dataset that facilitates rigorous analysis while upholding the statistical integrity of the sampling process.

4.3 Data processing and Analysis

4.3.1 Metadata Analysis

As shown from Appendix B to G, the data analysis phase conducted using different techniques. The PCAP analysis Module revolves around the efficient processing of network packet capture (PCAP) files and storing the extracted data in MongoDB. The the multi threaded custom PCAP analysis module serves as the core component of the system, the Module utilizes the Pyshark library for comprehensive packet analysis, extracting crucial attributes such as source and destination IPs, ports, protocols, and timestamps. And also, it integrates GeoIP lookup functionality to enrich the extracted data with geographic location information, thereby enhancing analytical depth.

Following the pre analysis, the Module securely saves the processed data to a MongoDB database for persistent storage. MongoDB employs efficient indexing and querying mechanisms, delivering quick read and write operations—a crucial component for processing and storing real-time network data, enhancing overall system performance and responsiveness. MongoDB's distributed architecture enables horizontal scalability, allowing the database to handle large volumes of data and high throughput, ensuring optimal performance and reliability even under heavy workloads. By establishing a secure connection to the MongoDB server, the Module inserts the transformed data into designated collections, ensuring accessibility and integrity for subsequent analysis. Throughout this

process, comprehensive logging mechanisms capture pertinent events for monitoring and troubleshooting, ensuring the robustness and reliability of the system.

To enhance data integrity by systematically removing duplicate entries from specified datasets. This process involves reading the data files, identifying and eliminating duplicate records, and subsequently saving the refined dataset to a designated output file. Furthermore, an ancillary objective of this research is to extract and retain only unique destination IP addresses associated with African countries, which will be saved in a separate output file.

DNS resolution is a crucial step in the process of accessing websites or servers on the internet. When a user enters a domain name into their web browser, the IP address of the server hosting the website is converted from that domain name by the DNS system. This allows the user's device to connect with the server and retrieve the desired information. The custom script automates this process by extracting IP addresses from a unique African countries IP addresses and resolve the corresponding domain names. In the script, successful DNS resolutions are stored in a separate JSON file , while failed resolutions are logged in another file.

The script utilizes threading to resolve DNS for multiple items concurrently to improve processing efficiency. Threading allows the script to delegate the DNS resolution tasks to separate threads, enabling them to run simultaneously. This parallel processing approach can significantly reduce the overall execution time of the script, especially when handling a large number of IP addresses. By leveraging threading, the script optimizes resource utilization and enhances the overall performance of the DNS resolution process.

4.3.2 Content Analysis

The custom web scraping scripts as shown from Appendix [H](#) to [J](#) are a sophisticated tools designed to automate the process of gathering data from various websites, specifically focusing on extracting textual content from designated web domains. This script operates by first performing the essential task of pulling text from a specified domain, subsequently saving that content to a text file. To enhance reliability, the module is equipped with robust error-handling capabilities, which log any exceptions that may

be encountered during the scraping process. This ensures that errors are documented systematically, allowing for easier troubleshooting and quality control.

One significant feature of the script is its ability to perform concurrent scraping across multiple domains through the implementation of threading functions. This multithreading capability allows the script to initiate multiple scraping processes simultaneously, thereby improving efficiency and significantly reducing overall data retrieval time. When scraping large numbers of domains, serial execution could lead to bottlenecks, as each web scraping task would need to be completed before the next one begins. By leveraging multithreading, the script can operate asynchronously, allowing it to handle multiple requests at once. This is particularly advantageous in scenarios where response times may be unpredictable, such as when dealing with various server loads or network latencies.

As the script performs concurrent data scraping, it not only expedites the retrieval process but also organizes the results into distinct JSON files, categorizing them based on the scraping status—clearly distinguishing between successful and failed attempts. Additionally, the script’s effective management of exceptions throughout the scraping process ensures a smooth user experience while minimizing disruptions. This reliability provides researchers with a powerful tool to efficiently harness large volumes of web data.

4.4 Web content classification

The custom Text Matching collection of scripts as shown from Appendix [K](#) to [P](#) are tailored to classify scraped data into categories by comparing with the labeled textual data of Shallalist using diverse similarity scoring metrics. The text-matching Module boasts a robust system architecture designed to facilitate the efficient processing of textual data and computation of similarity metrics using multithreading approach. Let’s dive into the various components and implementation details of the system: The first step in the Web Content Classification module is data loading and preprocessing. This process focuses on cleaning the scrapped text to enhance its effectiveness for subsequent algorithms. The goal is to optimize the performance of TF-IDF and Cosine similarity calculations, thereby creating a more organized and relevant text representation. Here are the detailed steps:

1. **Lowercasing:** Changing all letters to lowercase for consistency and uniformity in analysis.
2. **Cleaning:** Removing non english characters .
3. **Tokenization:** Breaking the text into separate tokens, where each word becomes a unit that can be processed.
4. **Stopwords:** Eliminating common words that don't contribute significantly to the analysis. It's for only TF-IDF and cosign. In MiniLM, stop words are not removed because the language model treats them as they are.
5. **lemmatization:** Changing words to their base form to treat words with the same root as the same entity.

The similarity computation layer consists of multiple algorithms designed to compute similarity scores between pairs of textual data. These algorithms include TF-IDF similarity, Cosine similarity, and Minilm similarity computation. TF-IDF similarity computation measures the similarity between documents based on the frequency of terms in each document relative to the entire corpus. Cosine similarity computation calculates the cosine of the angle between vector representations of documents, indicating their similarity in terms of direction. Minilm similarity computation, a variation of Jaccard similarity, computes similarity based on term intersections and unions. Each algorithm operates on preprocessed textual data, leveraging mathematical models and statistical techniques to quantify the similarity between text pairs accurately. By offering multiple similarity computation algorithms, the module provides flexibility and adaptability to different use cases and data characteristics.

Once similarity scores are computed using various algorithms, the module generates JSON files containing the results. Each JSON file is named after the corresponding similarity metric used and stored in the specified output folder. These files serve as the output of the text matching process, providing detailed similarity scores between textual data pairs. The merging script connects the matching results with their corresponding DNS information obtained during the DNS resolution phase, thereby offering additional context and insights into the matched textual data. This module enhances the value and utility of the matching results, allowing users to gain a deeper understanding of the textual data and its relevance within specific domain contexts.

4.5 Visualization

The proposed solution integrates powerful tools for comprehensive analysis and visualization of processed network traffic data. Elasticsearch supports efficient retrieval and analysis through its robust full-text search capabilities, enabling complex searches across large datasets with remarkable speed and accuracy, and also users can derive actionable insights and make informed decisions based on searchable and indexed data. These features facilitates a profound understanding of network traffic while allowing real-time indexing and analytics.

Complementing Elasticsearch, Kibana provides a user-friendly interface for visualizing and creating dashboards from the indexed data. Users can generate a variety of visualizations, including tables, graphs, charts, and maps, to effectively examine and interpret network traffic. This integration ensures that users can create meaningful insights and comprehensive reports from the analyzed data. The combination of Elasticsearch's powerful search capabilities with Kibana's visualization tools offers a cohesive framework that transforms raw network traffic into actionable intelligence, empowering users to make informed decisions based on real-time data analysis.

In the realm of data integration and analysis, rapid connection and processing of data are crucial. The Elastic Connector Module plays a vital role in this system setup by efficiently sending JSON results in smaller segments and employing multiprocessing to enhance speed and performance. Within the Elastic Connector Module, there are two types of forwards designed to optimize data handling:

4.5.1 Forwarding JSON Results to Elasticsearch

The Elastic Connector Module as shown in Appendix [U](#) and [V](#) is designed to enhance the efficiency of indexing and storing data in Elasticsearch by forwarding JSON results in manageable chunks through the use of multiprocessing. This innovative approach allows the module to process data concurrently, optimizing resource utilization and significantly improving performance. By breaking down data into smaller segments, the module prevents potential overload on the Elasticsearch instance, ensuring smoother operations. It effectively manages connection parameters such as host, port, and authentication credentials to establish a robust and secure communication channel with

Elasticsearch. Leveraging well-established libraries like the Elasticsearch Python client enables seamless and reliable indexing operations. Comprehensive logging capabilities further enhance the module's functionality by monitoring system behavior in real-time. Using Python's logging module, it captures relevant events and errors, storing logs securely for easy analysis and timely issue resolution. This structured approach makes the Elastic Connector Module a vital tool to optimize the data handling processes effectively while ensuring reliability and efficiency in data integration and analysis.

4.5.2 Forwarding MongoDB Data to Elasticsearch

Also, the Elastic Connector Module provides an efficient method for forwarding MongoDB data to Elasticsearch, enabling seamless migration and indexing. By establishing secure connections, creating necessary indices, and utilizing multiprocessing, this module allows users to harness Elasticsearch's powerful capabilities for comprehensive data analysis and retrieval. The structured process not only ensures effective data migration but also maximizes performance through optimized techniques. The implementation begins with establishing secure connections to both MongoDB and Elasticsearch, identifying essential connection parameters such as hosts, ports, and authentication details. It verifies the existence of the Elasticsearch index for data forwarding, creating it dynamically if necessary. The module then queries MongoDB collections to retrieve and transform the data into a format compatible with Elasticsearch's index structure. By employing multiprocessing to handle data in manageable chunks concurrently, it efficiently indexes the data into the appropriate fields within Elasticsearch, maintaining data integrity. Throughout this process, robust logging and error handling mechanisms are in place to capture key events and facilitate troubleshooting, ensuring a reliable migration experience.

The appendix includes all essential details, including program code, experimental data, and proofs, for reference. The concepts and methodologies utilized in the solution are grounded in the existing literature, and relevant citations can be found in the literature review section for further elaboration. In conclusion, the solution presented in this thesis section represents a significant contribution to the field of network traffic monitoring and data analysis of darkweb data.

Chapter 5

Experiments and Analysis

5 Experiments and Analysis

In the evolution from Version 1 to Version 2 of the PCAP processing system, significant improvements have been made across various modules to enhance functionality, scalability, and performance.

In our analysis, we aimed to quantify the task completion times and resource utilization of the two versions of the Pcap analyzer scripts. As shown in 5.1 and 5.2 , to achieve this, we employed a variety of profiling tools in Python to monitor overall memory usage. However, we adopted a more straightforward methodology by utilizing htop, a cross-platform interactive process viewer, which enabled us to visually capture and compare the disparities in CPU utilization and task completion times between the two approaches.

In Version 1, the system relied on sequential processing methods and stored processed data as CSV files. While functional, Version 1 had limitations in processing speed, scalability, and search capabilities. To address these deficiencies, Version 2 was developed,

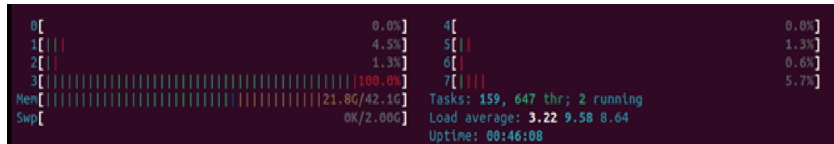


FIGURE 5.1: Single thread Pcap analyzer script

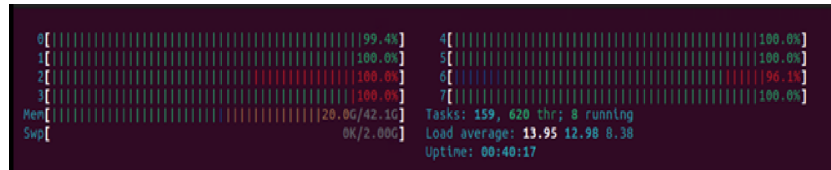


FIGURE 5.2: Multi-threaded Pcap analyzer script

incorporating multi-threading capabilities. When tasks are processed sequentially, the estimated duration is around 1 hour and 20 minutes. However, employing multithreaded approach significantly shortens this time to just 30 minutes. This enhancement significantly improves both processing speed and scalability, allowing for more efficient data handling and retrieval. The transition from a sequential to a multithreaded architecture enables the system to better manage larger datasets and perform complex searches more rapidly, ultimately enhancing overall performance and user experience.

Additionally, the system transitions from CSV files to MongoDB for data storage, addressing limitations associated with large volumes of data. MongoDB's document-oriented architecture and efficient indexing make it well-suited for handling extensive datasets and Version 2 leverages the Elastic-connector script to enhance data indexing, search capabilities, and visualization. The Elastic connector optimizes data indexing strategies and leverages Elasticsearch's powerful search capabilities. Although processed data is stored directly in MongoDB, Elasticsearch remains integral for real-time analytics and visualization through Kibana.

In terms of transferring data to Elasticsearch via the elastic connector, the sequential approach requires 2 hours and 47 minutes. On the other hand, using multithreaded processing for the operation enhances efficiency, cutting the time down to 1 hour. Overall, Version 2 represents a comprehensive PCAP processing script, incorporating advanced technologies and optimization techniques to address limitations encountered in Version 1. The strategic decisions regarding multithreading, MongoDB adoption, and the Elastic-connector reflect a concerted effort to optimize system performance, scalability, and resource utilization while leveraging Elasticsearch's strengths for real-time analytics and visualization.

The first version of the pcap analyzer code had the following Features and limitations;

Features:

- Basic functionalities for extracting, transforming, and storing network packet data from PCAP files.
- Utilization of CSV files for storing processed data.
- Sequential processing methods employed for data extraction and transformation.

Limitations:

- Processing speed bottlenecks, especially with large volumes of data.
- Scalability challenges due to reliance on CSV files and sequential processing methods.
- limited search functionality and limited visualization functionality

To overcome the aforementioned limitations, we enhanced the code by adding additional features. However, we still faced some challenges due to the fact that the tools required a lot of resources.

Features:

- Introduction of multi-threading capabilities for concurrent execution of processing tasks, improving processing speed and scalability.
- Transition from CSV files to MongoDB for more efficient data storage, handling large volumes of data with ease.
- Enhanced data indexing and search capabilities through the Elastic connector, optimizing indexing strategies and leveraging Elasticsearch's powerful search capabilities.
- Introduction of the Elastic connector script to streamline data forwarding from MongoDB to Elasticsearch, enabling bulk indexing and multi-threaded processing for improved performance.
- Refinement of visualization capabilities enables the creation of meaningful visualizations and dashboards directly from indexed data in Elasticsearch.

Changes:

- Despite improvements, processing speed may still be affected when dealing with extremely large volumes of data.
- While MongoDB enhances scalability, challenges may still arise with very high data throughput.

Upon reviewing the scraping script, we discovered that it was initially designed only to scrape the first page of the web content. As a result, the script did not display the full web content inside the web page. To address this limitation, we implemented modifications to the script to ensure that all contents within the web page were collected. This measure proved to be effective in improving the results of the text matching process, as it allowed for a more comprehensive and accurate analysis of the web content.

In a performance analysis conducted on an Intel Core i7-8700K processor equipped with 16 GB of RAM and a 100 Mbps internet connection, the process of scraping 1,000 URLs demonstrated significant differences in execution time based on the method used. Sequential execution took approximately 33.33 minutes to complete, highlighting the inefficiencies of a single-threaded approach for this task. In contrast, employing a multi-threaded execution strategy with 10 threads dramatically improved performance, reducing the scraping time to just 200 seconds, or 3.33 minutes. This represents a remarkable reduction in execution time, underscoring the advantages of parallel processing when handling multiple web requests simultaneously.

Regarding the text matching module, significant enhancements were made to the first version of the code. These improvements included optimizing the code, enhancing performance, and minimizing code complexity through the utilization of an online Python code complexity analysis tool. So, in the latest update of the text matching script, significant enhancements have been made to boost its overall efficiency and functionality. we have updated the structure, making the codebase more streamlined and readable. This restructuring simplifies future maintenance and improves scalability as the project evolves, minimizing redundancy and enhancing the architecture for better adaptability and we have focused on elevating code complexity to enhance performance. By utilizing parallel processing techniques, especially through Torch multiprocessing capabilities, we have notably increased the script's performance. This optimization allows us to handle larger datasets and execute computations more swiftly, significantly reducing processing times and making our text matching algorithm more responsive.

Moreover, we have integrated a comprehensive pre-processing phase to enhance the accuracy and reliability of our text matching system. This phase includes essential steps like lowercasing, cleaning, tokenization, and lemmatization, ensuring standardized and optimized input text for matching. Refining the preprocessing pipeline has improved data quality and consequently boosted the precision of our matching algorithm, setting a solid groundwork for future feature additions and refinements. we have also introduced a robust logging mechanism for easier troubleshooting and monitoring. This logging mechanism offers detailed insights into script execution, facilitating error tracking and output analysis. By logging key events and metrics, users can diagnose issues more efficiently and make informed decisions to optimize performance further. This logging mechanism enhances usability and maintainability, ensuring smooth operation and supporting continuous improvement over time.

Additionally, the analysis revealed that TF-IDF and the Cosine algorithm employed a method of counting words in text for comparison. These algorithms determined similarity by analyzing similar word counts and the angle between the word vectors. While both methods produced comparable results, they were found to be insufficient in categorizing text based on behavioral patterns as shown in [6.17](#), and [6.18](#). Furthermore, their effectiveness in accurately categorizing web pages was hindered by the limited similarity in word count across web content.

To address these shortcomings, it was crucial to incorporate a behavioral language corpus such as Mini-LM. By utilizing Mini-LM, we were able to overcome the aforementioned limitations and achieve more precise text categorization. This corpus offered a comprehensive understanding of behavioral language patterns, enabling a more nuanced analysis of text content. By integrating Mini-LM, we enhanced the accuracy and effectiveness of text categorization methods, ultimately improving the overall quality of analysis.

Chapter 6

Results and Discussion

6 Results and Discussion

6.1 Results

A critical aspect of the hosting setup included the configuration of a Tor exit node to streamline data gathering and parsing. This process involved installing Tor software, editing the Tor configuration file to enable exit node functionality, setting up port forwarding for Tor traffic, securing a static IP address for reliability, and defining exit policies to manage traffic types in the configuration file. We can look at the evidence in visual representation in the following figures [6.1](#), [6.2](#), and [6.3](#) after successfully establishing the hosting environment and configuration setup. The result has been obtained directly by searching the relay, i.e., EastAfricanRelay on [\[38\]](#).

The integration of Wireshark within the hosting environment proved instrumental in successfully capturing and analyzing a substantial volume of TOR traffic over a 35-day period. The ambitious undertaking focused on gathering data from the hidden realm of the internet, known as the "darkweb," leveraging Wireshark's packet capture capabilities for comprehensive network traffic analysis. The collected dataset exceeded an impressive 5 terabytes (TB), This data was stored in the form of around 20,304 PCAP files, each with a size of 250 MB.

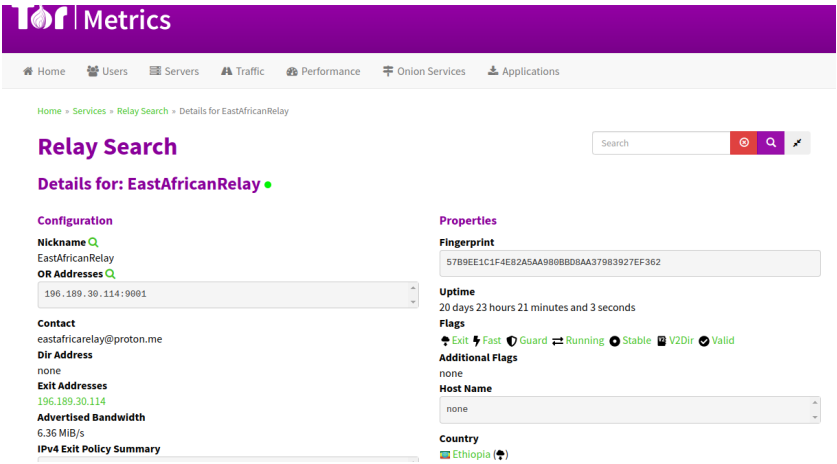


FIGURE 6.1: Exit relay

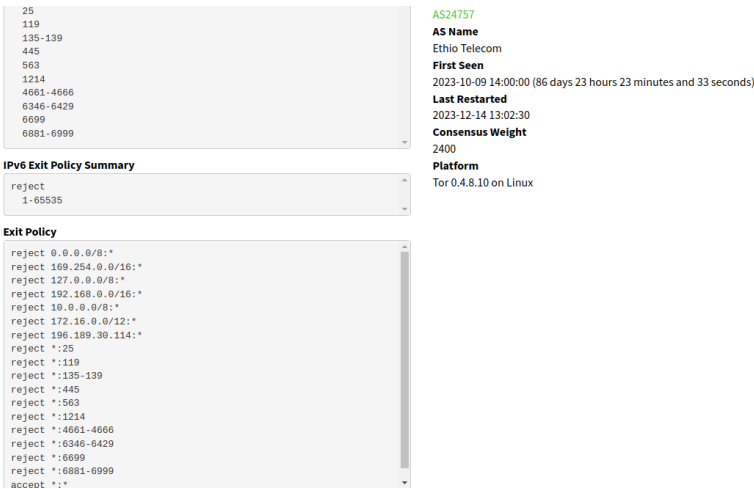
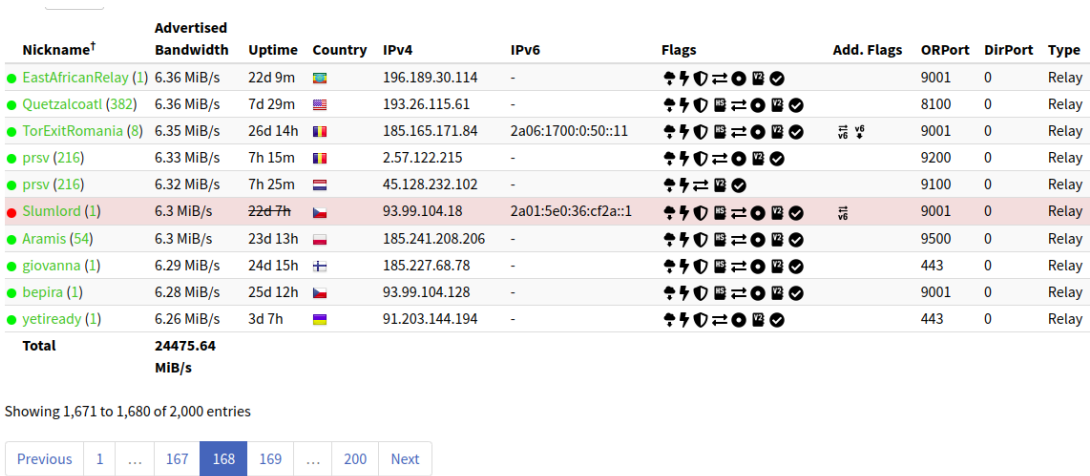


FIGURE 6.2: Exit policy



[†]The number shown in parentheses is the total effective family size for the relay including the relay itself. A size of 1 indicates that the relay does not have any other effective family members. Note that bridge relays do not advertise family members in their descriptors and so there is no value shown for bridge relays.

Information for relays was published: 2024-01-05 11:00:00 UTC.

Information for bridges was published: 2024-01-05 09:26:33 UTC.

FIGURE 6.3: Exit relay property

The successful implementation of the random sampling methodology through the development of a specialized script facilitated the extraction of a representative sample, constituting 5% of the total data across multiple PCAP files for in-depth analysis. This meticulous categorization process enabled the script to generate a comprehensive report displaying the dates alongside the corresponding file counts in ascending order illustrated in [Y.1](#) to provide a clear overview of the distribution.

Upon executing the source code for the PCAP analysis Module, the efficient processing and storage of network packet capture (PCAP) data in MongoDB was successfully achieved. After filtering essential attributes, we obtained around 258.6 million records with the size of around 136.8 GB JSON file after exportation, each containing destination and source IP addresses and ports, timestamps, the source and destination countries, the protocol type, and the TLS version of the transmitted packet as shown in the figures [6.4](#), [6.5](#), and [6.6](#).

Also, the result revealed that the top three destinations countries on that pass through the configured Tor exit node are the United States of America, Germany, and France, and Ethiopia and South Africa are also among the top 50 destination countries from Africa. While we analyzed the protocol of those destination IPs, we found that 58.03% are TCP, 38.54% TLS, 2.38% DATA, and the other protocol types include Bitcoin, HTTP, DNS, and SSH protocols. The TLS version of the randomly selected data consisted of 98.1% TLS 1.2, 0.66% TLS 1.0, and 1.24% SSL 2.0.”

This result shows that the network is being used for various activities, including general web browsing and email communication, and more secure connections are established using TLS encryption. Additionally, there may be remote access to devices or servers using SSH, indicating potential administrative or maintenance tasks. Furthermore, the presence of Bitcoin and BitTorrent traffic suggests that users may be engaging in cryptocurrency-related activities or file sharing, respectively. Overall, this analysis provides a snapshot of the network’s usage patterns, highlighting the various ways in which users are interacting with the network and potentially identifying areas where security measures may be necessary.

Since the scope of the research is focused on African countries, the cleaning and filtering script was utilized to filter only African Destination IPs from the randomly selected data. During this filtering process, the script processed a 3.2MB JSON file containing

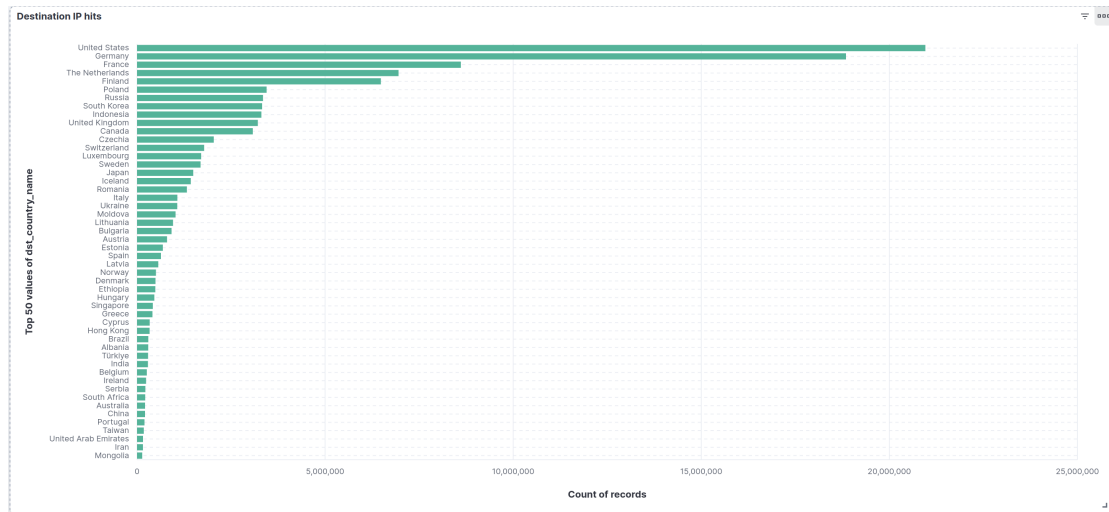


FIGURE 6.4: Destination countries of randomly selected data

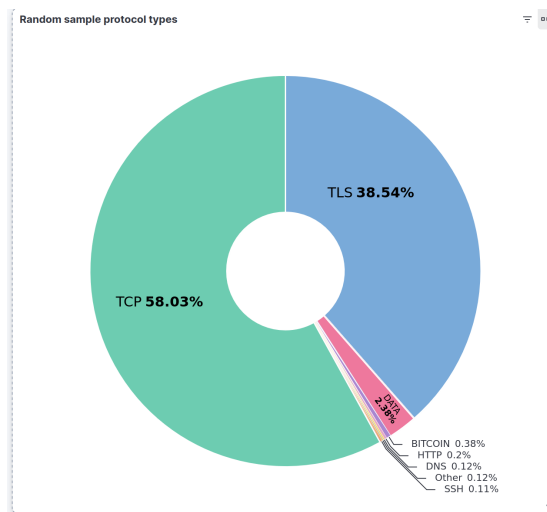


FIGURE 6.5: random data protocol types

unique destination IPs of all African countries. However, when we filtered only unique destination IPs of African countries, the top five darkweb traffic destination countries of Africa are South Africa, Egypt, Morocco, Nigeria, and Kenya. While we analyzed the protocol of the destination IPs, we found that 89.89% are TCP, 4.22% is TLS, 4.53% are DATA, and the rest are covered by SSH, HTTP, BitTorrent, and other protocol types. The TLS version of the African destination countries consisted of 79.78% TLS 1.2 and 20.22% TLS 1.0 as shown in figure 6.7, 6.8, 6.9.

The first part of the scrapping script focuses on scraping web content from the collected traffic using Wireshark after conducting DNS resolution. The script effectively handles both successful and failed web content scraping scenarios, providing detailed

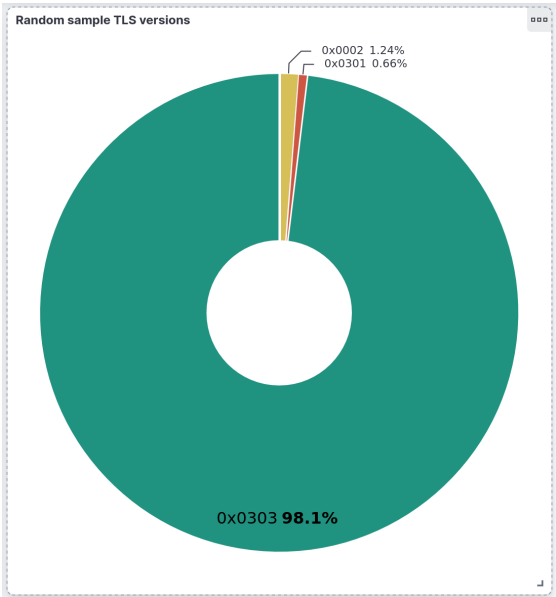


FIGURE 6.6: Random data TLS versions

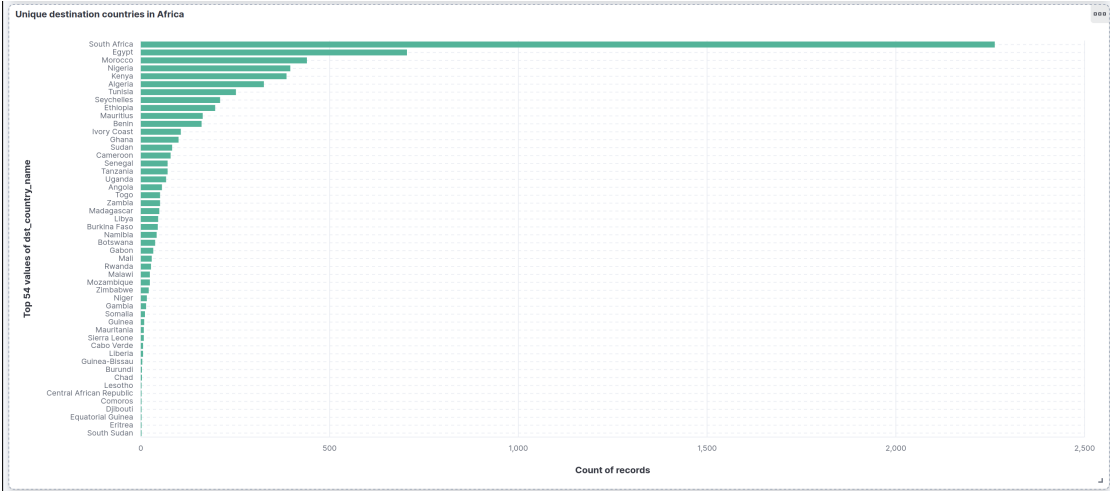


FIGURE 6.7: Unique Africa countries

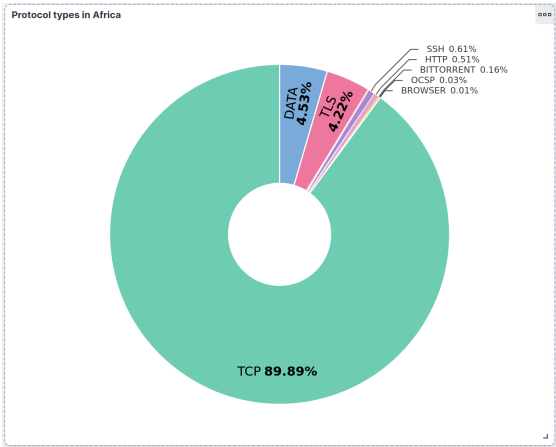


FIGURE 6.8: Unique African Protocol types

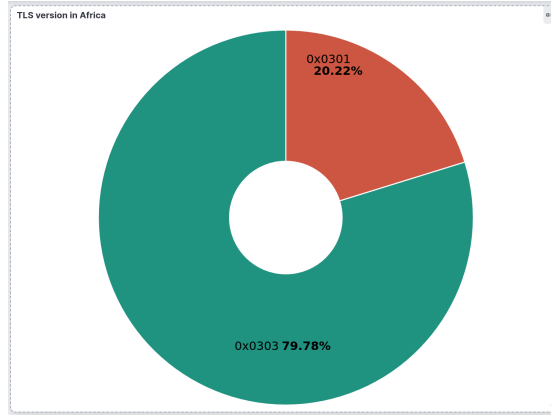


FIGURE 6.9: unique African TLS versions

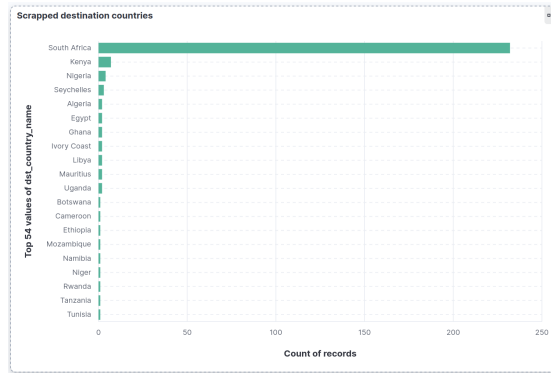


FIGURE 6.10: Scrapped destination countries

logs and outcomes for each domain processed. Concurrent scraping of multiple domains is achieved using threading function. As shown in 6.10, 6.11, and 6.12, the top five successfully scrapped destinations IPs are from South Africa, Kenya, Nigeria, Seychelles and Algeria. While we analyzed the protocol of successfully scrapped IPs, we found that 84.7% are TCP, 9.7% are TLS, 3.73.7% are HTTP, and the rest are covered by SSH and DATA protocol types. The TLS version of the successfully scrapped destination IPs are consisted of 65.38% TLS 1.2 and 34.62% TLS 1.0.

In cases where the web content scraping process encounters issues, the script logs the failures accordingly. As shown in 6.13, 6.14, 6.15, and 6.16 the top five failed scrapped destinations IP's are from South Africa, Kenya, Nigeria, Seychelles and Egypt. While we analyzed the protocol of failed scrapped IPs, we found that 84.97% are TCP, 8.6% are TLS, 5.03% are DATA, and the rest are covered by SSH and HTTP protocol types. The TLS version of the failed scrapped destination IPs are consisted of 75.31% TLS 1.2 and 24.69% TLS 1.0. The error logs encountered during the scraping stage indicate a

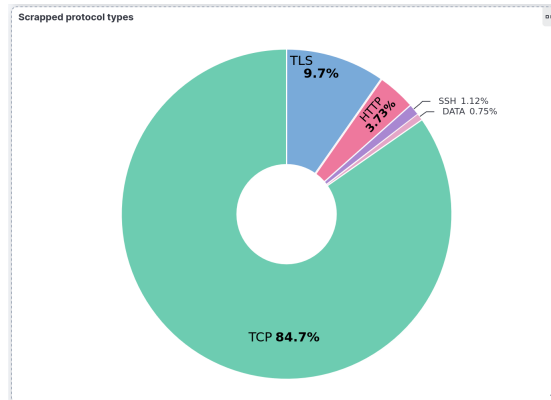


FIGURE 6.11: scrapped protocol types

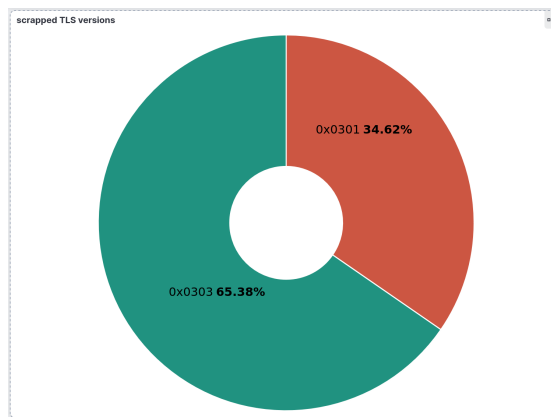


FIGURE 6.12: scrapped TLS version

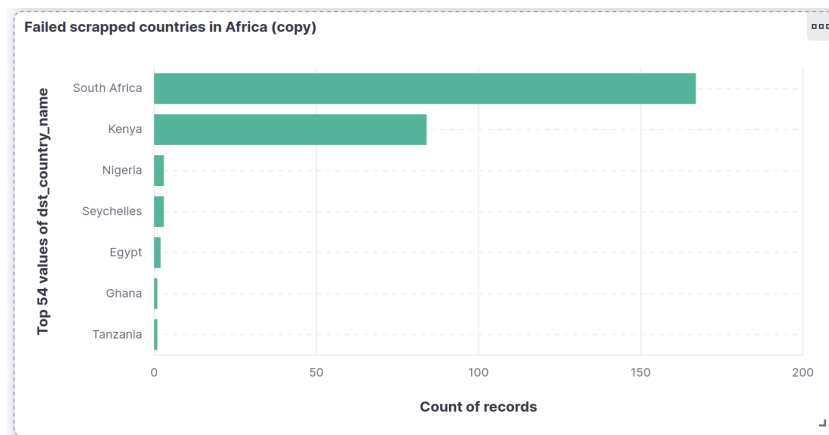


FIGURE 6.13: Failed scrape countries

significant number of errors, including 400 error status codes (85.77%), 404 error status codes (5.77%), and other errors, such as 401, 406, 410, 418, and 500.

The second scrapping part involves scraping the contents of the Shallalist domain and URLs. This content contained 55 categories of domains and URLs that we needed to

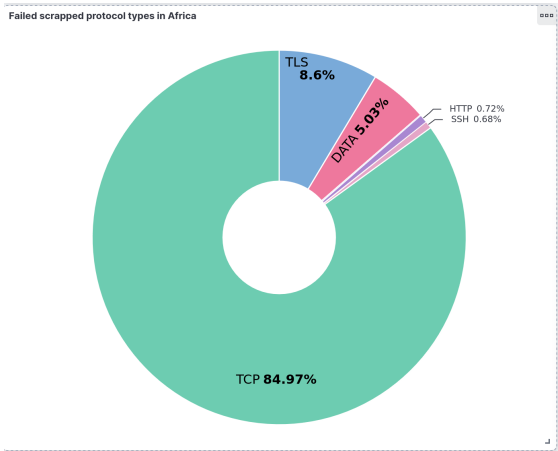


FIGURE 6.14: Failed scrape protocol types

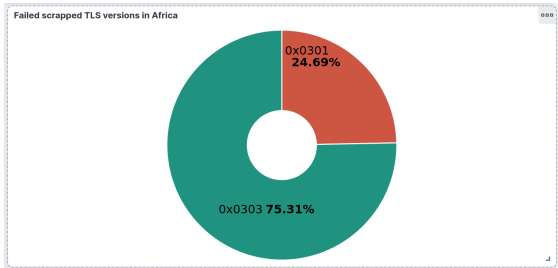


FIGURE 6.15: Failed scrape TLS versions

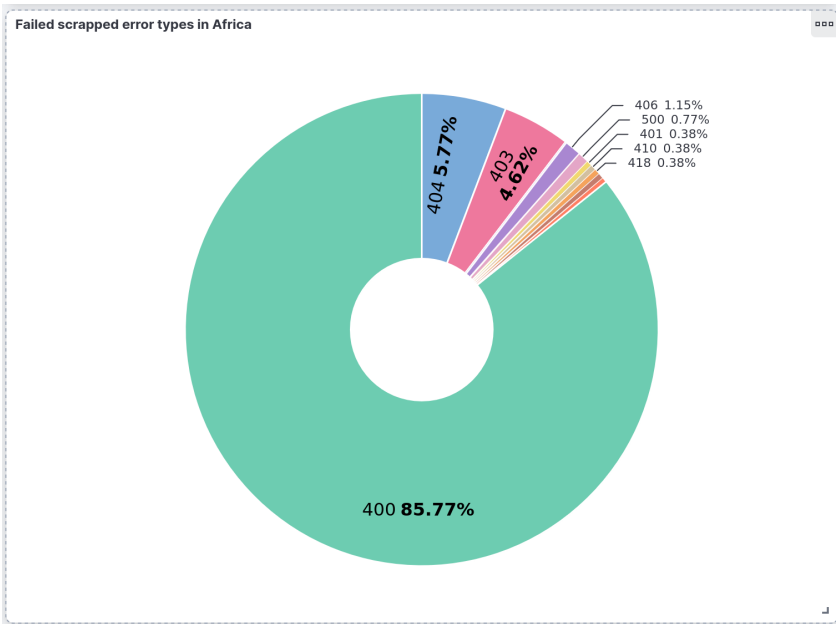


FIGURE 6.16: Failed scrape error logs

compare with the collected data. Through this scraping process, we were able to compile and merge the contents of different URLs within each category. By consolidating the information, we could effectively summarize the content under each category and assign a corresponding category name to the merged result.

The text matching computes similarity scores using various algorithms and generates detailed results that takes the Shallalist labeled dataset and web content scraping results as input, and stored in JSON files named after the specific similarity metric used. This output facilitates easy parsing and integration with other systems for further analysis or visualization, aiding users in interpreting and comparing the results effectively. Additionally, the script integrates with the Elastic Connector Module, leveraging Elasticsearch's indexing and visualization capabilities to offer users comprehensive insights into the matched textual data. This integration enhances the module's functionality by enabling the indexing and storage of merged JSON data for visualization purposes, bridging the gap between the textual matching results and Elasticsearch.

As shown in figures 6.17, 6.18, and 6.19 Jobsearch, ISP, AnonVPN, Hacking, Webphone, Dynamic, Forum, Military, Warez and Ads are the top 10 darkweb activities are for TF-IDF similarity algorithm, similarly for Cosign similarity algorithm, Jobsearch, Imagehosting, AnonVPN, Hacking, Dynamic, ISP, Violence, Warez, Military, and Webphone are the top 10 darkweb activities and finally, for Mini-LM similarity algorithm Webmail, Drugs, Ads, Warez, Hacking, Porn, Webphone, Downloads, Dynamic, and Forum are the top 10 darkweb activities conducted on African internet protocol address space.

In conclusion, based on the performance analysis of the text matching algorithms during the experimentation stage, the Mini-LM algorithm emerged as the top performer. This is because the Mini-LM algorithm takes a more nuanced approach by analyzing the behavior between two texts, rather than simply focusing on the similarity of words between them, as is the case with TF-IDF and Cosine similarity algorithms. In contrast, the Mini-LM algorithm is able to identify more subtle patterns and relationships between texts, which enables it to achieve better results.

As shown in 6.20, the Mini-LM algorithm identified illegal categories of Drugs, Warez, Hacking, and Porn with percentages of 18.92%, 4.02%, 3.6%, 3.15% respectively are conducted in African Internet protocol address space.

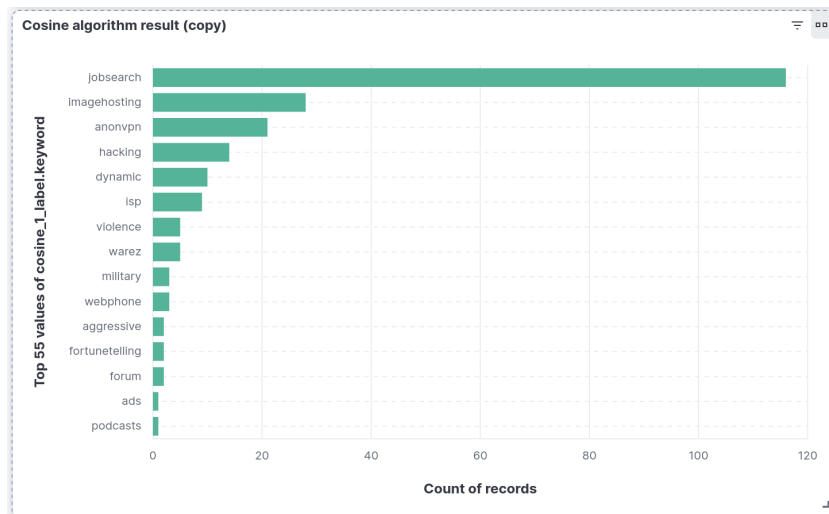


FIGURE 6.17: Cosign algorithm result

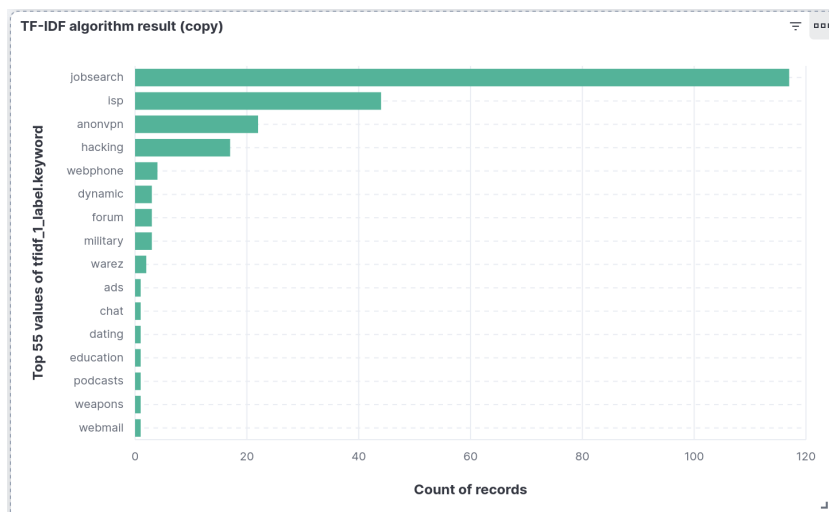


FIGURE 6.18: TF-IDF algorithm result

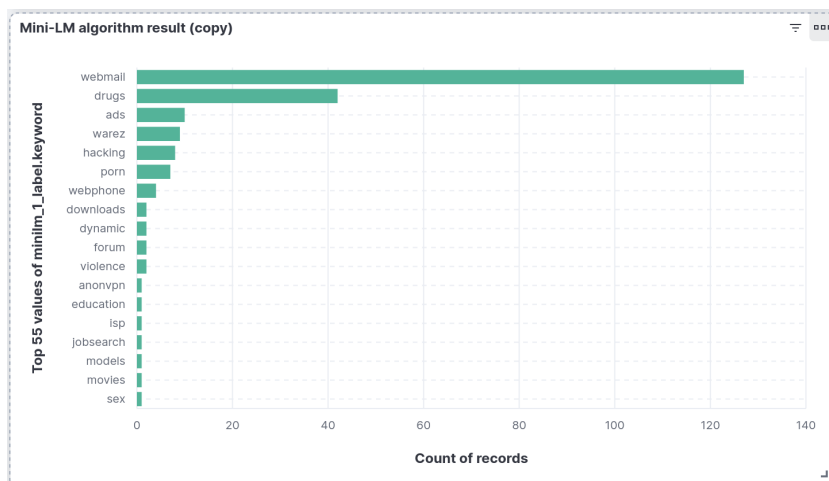


FIGURE 6.19: Mini-LM algorithm result

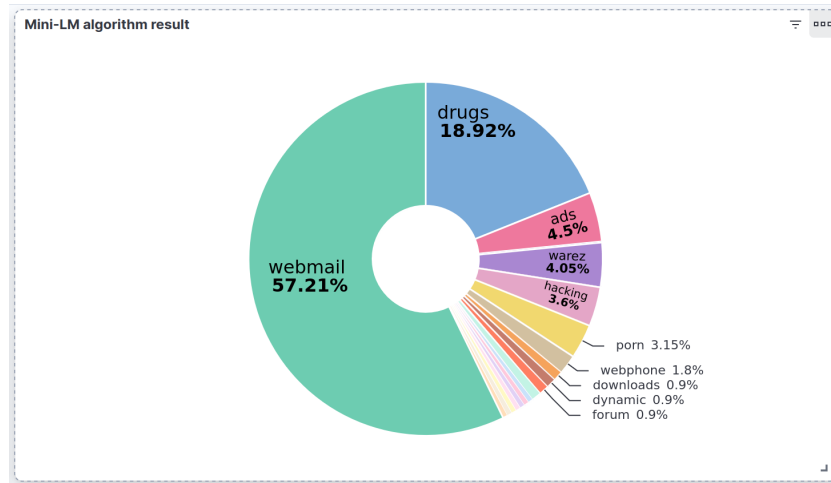


FIGURE 6.20: Mini-LM algorithm result with statistics

As shown in 6.21, Ethiopia, Kenya, Uganda, and Tanzania are among the East African countries where the Mini-LM algorithm found prohibited categories. The data classified as Warez, Violence, and Porn (with percentages of 27.27%, 9.09%, and 9.09%, respectively) is conducted in the Internet protocol address space of East Africa. This illegal activity is a significant concern in many African countries, where such behavior is strictly prohibited by law.

According to [40], [41], [42] and [43], most online activities, such as violence, drugs, warez, and pornography, are illegal in many East African countries. The regulations highlight that many countries in the region have laws and regulations prohibiting these activities and that law enforcement agencies are working to combat these crimes.

The report also notes that many East African countries have signed international agreements to combat cybercrime, such as the Convention on Cybercrime and the Budapest and Malabo Convention and others [44] [45] [46]. These agreements provide a framework for cooperation between countries to combat cybercrime and share best practices in this area.

6.2 Discussion

This study aimed to investigate the Dark Web's malicious activities in the East African region, an area that has been largely under-researched in this field. To achieve this, two research questions were posed: what types of protocols are prevalent in the Dark Web

The finding that the United States, Germany, and France are the top three destination countries implies that a significant volume of Dark Web activities originates in developed countries. This suggests that individuals engaged in these activities may have an awareness about the darkweb ecosystem, or that these countries have the necessary infrastructure to support such activities. The inclusion of Ethiopia and South Africa among the top 50 destination countries indicates a growing engagement with the Dark Web in East Africa. This suggests a potential increase in illicit activities, necessitating closer monitoring and potentially tailored law enforcement strategies in these regions.

The predominance of TCP (58.03%) in the analyzed traffic suggests that users are relying on a widely accepted, reliable protocol for establishing connections. Meanwhile, the significant usage of TLS (38.54%) highlights the emphasis on secure communications, which can complicate monitoring efforts by law enforcement. This security focus indicates that users are cognizant of privacy and may be intentionally seeking to anonymize their activities. The presence of the Bitcoin protocol within the protocol type serves as a significant indicator that users are engaging in cryptocurrency-related activities. By monitoring activities associated with Bitcoin transactions, including the addresses involved, transaction volumes, and patterns of use, researchers and analysts can glean insights into the underlying purposes for which the cryptocurrency is being utilized.

The dominance of TLS 1.2 (98.1%) in the traffic suggests that the users are adopting relatively modern security standards, which is a positive trend from a cybersecurity perspective. However, it also presents challenges for monitoring and data analysis since encrypted traffic can hinder the ability of law enforcement to inspect and understand the content and intent of communications. The very small usage of older versions like TLS 1.0 (0.66%) and SSL 2.0 (1.24%) raises concerns, as legacy protocols may have known vulnerabilities that could be exploited by malicious actors. This could indicate a lack of awareness among some users or applications that still rely on outdated protocols, providing another opportunity for law enforcements to inspect and monitor illegal darkweb users.

Understanding the types of illegal activities conducted on East African Internet Protocol (IP) address spaces is crucial for improving cybersecurity and informing policy development. By identifying prevalent threats, cybersecurity professionals can devise

appropriate defense mechanisms to protect networks and users in the region. Additionally, this knowledge aids governments and regulatory bodies in creating effective laws and regulations aimed at combating cybercrime, while law enforcement agencies can leverage insights for more targeted investigations and resource allocation. Insights into the scope of cybercrime can also facilitate international collaboration in addressing cross-border threats and help assess the political, economic, and social impact of such activities. Overall, this knowledge promotes proactive measures in securing and safeguarding the online landscape in East Africa.

The identification of specific illegal activities (e.g., warez, violence, and pornography) provides law enforcement agencies with actionable intelligence. By concentrating efforts on the most prevalent activities (e.g., the almost 19% share of drug-related activities in Africa and 27.27% the distribution of cracked software (warez) in east Africa), authorities can allocate resources more effectively and develop targeted operations against high-impact threats. Recognizing the percentages associated with various illegal activities allows law enforcement agencies to prioritize actions. The prominent identification of illegal activities within specific internet protocol address spaces can facilitate greater collaboration between regional and international law enforcement agencies. This emphasizes the need for information sharing to tackle cross-border cybercrimes effectively.

Additionally, The statistics on illegal activities provide a data-driven basis for developing and revising local laws and regulations. Governments can use this information to create more effective legislation aimed specifically at combating violence, the distribution of cracked software (warez), and online pornography. Recognizing the significant percentages of certain activities can lead to the creation of preventive policies. For example, enhancing internet safety education or implementing stricter regulations on internet service providers (ISPs) to monitor and report suspected illegal activities could be effective.

Furthermore, the recognition that warez-related activities are notable in the region raises the need for comprehensive cybersecurity policies. Policymakers can emphasize developing national cybersecurity strategies that include initiatives for minimizing the vulnerability of networks used in these illegal activities. The international nature of the observed traffic emphasizes the need for enhanced cross-border cooperation among law enforcement agencies. The study highlights the involvement of various nations in illicit

activities originating from East Africa. As a result, law enforcement agencies should foster regional collaboration and information-sharing agreements. By working together, agencies can pool resources and knowledge to tackle cybercriminal activities more effectively.

Regarding academic and research Implications, the findings can encourage further academic research into the trends emerging from African nations' increased involvement in the Dark Web. This could reveal insights into the motivations for engagement, the types of crimes prevalent in the region, and how these initiatives may evolve. The findings open up various avenues for academic research, particularly in criminology, cybersecurity, and digital sociology. Researchers could explore the motivations behind these activities in the context of East Africa, potentially uncovering social, economic, and cultural drivers of cybercrime.

Finlay, highlighting these illegal activities necessitates increased public awareness campaigns about the risks associated with engaging in or accessing such content. Informing citizens about the implications of warez-related activities, violence, and pornography can instigate community-driven solutions. Given the prevalence of pornography and violence-related content, there is potential concern regarding the exposure of youth to such materials. This points toward the need to establish protective measures and promote healthy online behaviors among younger populations. The emphases on warez or cracked software can have broader repercussions on the formal digital economy, potentially impacting local businesses and software developers. Raising awareness about the consequences of piracy and promoting legitimate software usage could foster growth in the software sector. These findings are concerning as they highlight the prevalence of illegal behavior in African countries, where such activities are strictly prohibited by law. The proposed solution ability to detect these types of activities demonstrates its potential for use in real-world applications, such as web content analysis and online monitoring.

One of the limitations of this study is that it only focused on DNS resolved domains of collected darkweb data. This could happen due to the change in the protocol used for the TOR domain name system generation. Future studies can build upon this research by exploring other methods for identifying illegal activities on the darkweb, specifically by including resolving .onion addresses. In addition, future studies could explore using

deep learning algorithms to identify illegal activities on the dark web. Deep learning algorithms can be trained on large datasets to identify patterns and trends in the data, providing valuable insights into illegal activities on the dark web. The study's conclusions have a number of ramifications for law enforcement organizations and other interested parties that are attempting to reduce illicit activity on the dark web. The results of the study imply that analyzing Tor network data can be a useful method for spotting illicit activity on the dark web.

Chapter 7

Summary and Future work

7 Summary and Future work

7.1 Summary

This study investigates the landscape of cybercrime on the Dark Web focused specifically on East African Internet Protocol (IP) address spaces, addressing a pronounced gap in the existing academic literature concerning the region. Cybercrime poses serious threats with personal, economic, and political ramifications. Resulting consequences include anxiety, depression, trauma, financial losses, and assaults on critical infrastructure. Moreover, cybercrime exploits societal conflicts and can become a tool for political manipulation, as seen in fake news and disinformation campaigns. Unfortunately, in East Africa, the burgeoning digital landscape, aided by technology and internet penetration, presents fertile ground for these criminal activities, exacerbated by inadequate readiness among security organizations and low user awareness. Recent initiatives like the Africa Cyber Surge II operation led by INTERPOL aimed to combat cybercrime but largely fell short in offering frameworks for ongoing regional analysis and detection methods. Prior research has neglected a thorough classification of illegal activities pertinent to the unique socio-political contexts of East African nations. Therefore, this thesis aims to fill that gap by presenting two primary research questions: (1) What types of protocols operate through the Dark Web in East African IP address spaces? and (2) What illegal activities are conducted through these protocols?

The objectives of this study were to develop a methodology for gathering and analyzing traffic data from Tor exit nodes specific to East Africa, classify the protocols, and categorize the illegal activities associated with them. An analysis of traffic passing through a configured Tor exit node revealed that the top three destination countries are the United States of America, Germany, and France. Additionally, Ethiopia and South Africa were found to be among the top 50 destination countries from Africa. Our examination of the protocols used by these destination IPs showed that TCP (58.03%) and TLS (38.54%) were the most commonly used, with smaller percentages using other protocols such as DATA, Bitcoin, HTTP, DNS, and SSH. Notably, our analysis of the TLS version used in the randomly selected data revealed that a vast majority (98.1%) employed TLS 1.2, with smaller percentages using TLS 1.0 (0.66%) and SSL 2.0 (1.24%).

The framework also detected illegal activities in African Internet Protocol address spaces, specifically related to Warez, Violence, and Porn (with percentages of 27.27%, 9.09%, and 9.09%, respectively) is conducted in the Internet protocol address space of East Africa. The prevalence of these illegal activities in African countries is a significant concern, as they are strictly prohibited by law. The proposed framework ability to detect these types of activities highlights its potential for use in real-world applications, such as web content analysis and online monitoring.

The framework devised herein incorporates advanced data analytics techniques and multithreaded computing, enhancing the performance of detection, identification, and analysis of illegal activities. Through rigorous analysis, the study identifies crime patterns—ranging from violence to software piracy—within the East African context, enriching the understanding of these issues and providing actionable data for law enforcement and policymakers. The study finds that the predominant protocols in use are TCP and TLS, indicating a preference for reliable and secure communications, which complicates monitoring activities by law enforcement and the presence of bitcoin reveal that the users are engaged with cryptocurrency activities. The implications of these findings highlight the necessity for targeted strategies in law enforcement operations, given the observable trends that indicate a growing engagement with the Dark Web in East Africa. Additionally, the data can contribute to the formulation of local policies and laws to combat cybercrime effectively.

This research makes significant contributions to the understanding of Dark Web dynamics in the East African context. The findings not only provide a foundation for improved situational awareness among cybersecurity professionals, law enforcement agencies, and policymakers but also pave the way for enhanced public awareness about the risks associated with illegal online activities. The developed framework offers a practical tool for analyzing Dark Web traffic, enabling stakeholders to better monitor and mitigate emerging cyber threats. By exhibiting specific illegal behaviors and their correlating percentages, this thesis provides law enforcement with crucial intelligence for resource allocation and strategic operations, particularly in prioritizing areas of high concern, such as warez. Furthermore, the nature of this study underscores the importance of regional collaboration and information sharing among law enforcement across national borders to tackle cross-border cybercrime effectively.

In summary, this study presents an insight into the challenges posed by cybercrime in East Africa and emphasizes the critical need for comprehensive research to guide effective responses in combating these illicit activities. The implications of this research extend beyond mere academic interest, advocating for a proactive approach to cybersecurity in a region experiencing rapid digital transformation. By understanding how cybercriminals operate and identifying the prevalent threats within the Dark Web, stakeholders can enact preventive measures, develop targeted interventions, and ultimately contribute to a safer digital environment in East Africa. This thesis not only enhances academic understanding of the topic but also serves as a valuable resource for law enforcement, policymakers, and researchers dedicated to mitigating cyber threats in the evolving digital landscape.

7.2 Future work

The current research has laid a solid groundwork for text matching and content categorization, albeit with certain limitations that present opportunities for future work and enhancements. One key limitation is the exclusion of content categorization for .onion addresses due to the unavailability of DNS resolution caused by changes in protocol specifications in the Tor name system. To address this issue, future research can focus on collecting .onion addresses from the dark web and mapping them with their corresponding Public IPs, enabling the utilization of the existing framework for labeling

and categorizing the content. By incorporating .onion addresses and advanced mapping techniques, this extension can provide a more comprehensive analysis of web content from such addresses, enriching the dataset and enhancing the framework's utility.

Furthermore, the existing research utilized light weight statistical and language model algorithms for content categorization, indicating a potential for further improvement through advanced language models using deep learning algorithms. Future work could involve enhancing the framework's capabilities by implementing sophisticated deep learning techniques to improve the accuracy and efficiency of content categorization. By leveraging deep learning algorithms, researchers can explore more complex patterns and structures within the textual data, leading to more robust and accurate categorization results.

Additionally, expanding the scope of the language detection algorithm beyond English to include multilingual capabilities can significantly enhance the framework's effectiveness in content classification. Integrating natural language processing (NLP) techniques for various local languages can broaden the framework's applicability and ensure more accurate categorization across diverse linguistic contexts. This multilingual approach can lead to a more comprehensive and inclusive content classification system that accommodates a wider range of languages, promoting greater accuracy and relevance in the categorization process. By addressing these limitations and incorporating these advancements, future research can significantly enhance the capabilities and effectiveness of the existing framework for content categorization and text matching, ultimately contributing to the development of more advanced and versatile tools for web content analysis.

References

- [1] Daksh Dave, Gauransh Sawhney, Pushkar Aggarwal, Nitish Silswal, and Dhruv Khut. The new frontier of cybersecurity: emerging threats and innovations. In *2023 29th International Conference on Telecommunications (ICT)*, pages 1–6. IEEE, 2023.
- [2] Petar Radanliev. Cyber diplomacy: defining the opportunities for cybersecurity and risks from artificial intelligence, iot, blockchains, and quantum computing. *Journal of Cyber Security Technology*, pages 1–51, 2024.
- [3] Dario Adriano and Bermudez Villalva. Understanding the difference in malicious activity between Surface Web and Dark Web. 2022.
- [4] Lucian V. Online African organized crime from surface to dark web. *Interpol*, (July):0–78, 2020.
- [5] Eda SÖNMEZ and Keziban SEÇKİN CODAL. Terrorism in Cyberspace : A Critical Review of Dark Web Studies under the Terrorism Landscape. *Sakarya University Journal of Computer and Information Sciences*, 5(1), 2021. doi: 10.35377/saucis...950746.
- [6] Saiba Nazah, Shamsul Huda, Jemal Abawajy, and Mohammad Mehedi Hassan. Evolution of dark web threat analysis and detection: A systematic approach. *IEEE Access*, 8:171796–171819, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3024198.
- [7] Sazelo Michael Mkhize. Internet and drug trafficking in africa. pages 137–158, 01 2023. doi: 10.1007/978-981-99-3057-9_6.

-
- [8] Job N. Manani. Regional economic communities and transnational crimes: the case of human community trafficking in the east africa, 2018. URL <http://erepository.uonbi.ac.ke/handle/11295/104824>.
- [9] Stewart Bertram and Keith Ellison. Sub saharan african terrorist groups' use of the internet. *Journal of Terrorism Research*, 5, 02 2014. doi: 10.15664/jtr.825.
- [10] Dario Adriano Bermudez Villalva. Understanding the difference in malicious activity between Surface Web and Dark Web. Technical report.
- [11] Arash Habibi Lashkari, Gurdip Kaur, and Abir Rahali. DIDarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning. *ACM International Conference Proceeding Series*, pages 1–13, 2020. doi: 10.1145/3442520.3442521.
- [12] Robin Gandhi, Anup Sharma, William Mahoney, William Soudan, Qiuming Zhu, and Phillip Laplante. Dimensions of cyber-attacks: Cultural, social, economic, and political. *Technology and Society Magazine, IEEE*, 30:28 – 38, 02 2011. doi: 10.1109/MTS.2011.940293.
- [13] Impact of cybercrime on individuals, businesses, and society. how to safeguard yourself?, 03 2024. URL <https://kce.ac.in/impact-of-cybercrime-on-individuals-businesses-and-society-how-to-safeguard-yourself/>.
- [14] UoP Online. What is the impact of cybercrime on society and the economy?, 02 2022. URL <https://studyonline.port.ac.uk/blog/what-is-the-impact-of-cybercrime-on-society-and-the-economy>.
- [15] Old Castle Brewery, Cape Town, and South Africa. Exploring Data Anonymisation and Internet safety in East Africa Submitted to : Research ICT Africa Type of Document : Policy Paper. 2020.
- [16] Cybercrime: 14 arrests, thousands of illicit cyber networks disrupted in africa operation. URL <https://www.interpol.int/News-and-Events/News/2023/Cybercrime-14-arrests-thousands-of-illicit-cyber-networks-disrupted-in-Africa-operation>.
- [17] Torbjörn Jonsson and Gustaf Edeby. Collecting and analyzing Tor exit node traffic. (June), 2021. URL www.bth.se.

-
- [18] Darren R. Hayes, Francesco Cappa, and James Cardon. A framework for more effective dark web marketplace investigations. *Information (Switzerland)*, 9(8):1–17, 2018. ISSN 20782489. doi: 10.3390/info9080186.
- [19] C. Gokhale and O. O. Olugbara. Dark Web Traffic Analysis of Cybersecurity Threats Through South African Internet Protocol Address Space. *SN Computer Science*, 1(5), 2020. ISSN 26618907. doi: 10.1007/s42979-020-00292-y. URL <https://doi.org/10.1007/s42979-020-00292-y>.
- [20] Abhineet Gupta, Sean Maynard, and Atif Ahmad. The Dark Web Phenomenon: A Review and Research Agenda. *Australasian Conference on Information Systems 2019, Perth*, pages 1–11, 2019.
- [21] Bruno Dantas, Paulo Carvalho, Solange Rito Lima, and João Marco C. Silva. Detection of anonymised traffic: Tor as case study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12526 LNCS:95–109, 2020. ISSN 16113349. doi: 10.1007/978-3-030-65729-1_9.
- [22] Zhengyi Li, Xiangyu Du, Xiaojing Liao, Xiaoqian Jiang, and Tiffany Champagne-Langabeer. Demystifying the dark web opioid trade: Content analysis on anonymous market listings and forum posts. *Journal of Medical Internet Research*, 23(2), 2021. ISSN 14388871. doi: 10.2196/24486.
- [23] Adrian Calvo, Paulo; Guevara-Coto, Jose ; Lara. Classifying and Understanding Tor Traffic Using Tree-based Models-2020.pdf. 2020.
- [24] Antonio Montieri, Domenico Ciuonzo, Giuseppe Aceto, and Antonio Pescape. Anonymity Services Tor, I2P, JonDonym: Classifying in the Dark (Web). *IEEE Transactions on Dependable and Secure Computing*, 17(3):662–675, 2020. ISSN 19410018. doi: 10.1109/TDSC.2018.2804394.
- [25] Ping Hu, Yuzong ; Zou, Futai ; Li, Linsen; Yi. Traffic Classification of User Behaviors in Tor, I2P, ZeroNet, Freenet-2020.pdf. 2020.
- [26] Akshobhya M. Machine learning for anonymous traffic detection and classification. pages 942–947, 01 2021. doi: 10.1109/Confluence51648.2021.9377168.

- [27] Muhammad Bilal Sarwar, Muhammad Kashif Hanif, Ramzan Talib, Muhammad Younas, and Muhammad Umer Sarwar. DarkDetect: Darknet Traffic Detection and Categorization Using Modified Convolution-Long Short-Term Memory. *IEEE Access*, 9(D1):113705–113713, 2021. ISSN 21693536. doi: 10.1109/ACCESS.2021.3105000.
- [28] Nhien Rust-Nguyen and Mark Stamp. Darknet Traffic Classification and Adversarial Attacks. jun 2022. URL <http://arxiv.org/abs/2206.06371>.
- [29] Debmalaya Sarkar, P. Vinod, and Suleiman Y. Yerima. Detection of Tor Traffic using Deep Learning. *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, 2020-Novem, 2020. ISSN 21615330. doi: 10.1109/AICCSA50499.2020.9316533.
- [30] Konstantinos Demertzis, Konstantinos Tsiknas, Dimitrios Takezis, Charalabos Skianis, and Lazaros Iliadis. Darknet traffic big-data analysis and network management for real-time automating of the malicious intent detection process by a weight agnostic neural networks framework. *Electronics (Switzerland)*, 10(7), 2021. ISSN 20799292. doi: 10.3390/electronics10070781.
- [31] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of tor traffic using time based features, 2017.
- [32] Yizhi Liu, Fang Yu Lin, Zara Ahmad-Post, Mohammadreza Ebrahimi, Ning Zhang, James Lee Hu, Jingyu Xin, Weifeng Li, and Hsinchun Chen. Identifying, Collecting, and Monitoring Personally Identifiable Information: From the Dark Web to the Surface Web. *Proceedings - 2020 IEEE International Conference on Intelligence and Security Informatics, ISI 2020*, pages 1–6, 2020. doi: 10.1109/ISI49825.2020.9280540.
- [33] Sagar Samtani, Yidong Chai, and Hsinchun Chen. Linking Exploits from the Dark Web to Known Vulnerabilities for Proactive Cyber Threat Intelligence: An Attention-Based Deep Structured Semantic Model. *MIS Quarterly*, 46(2):911–946, may 2022. ISSN 02767783. doi: 10.25300/MISQ/2022/15392. URL <https://misq.umn.edu/linking-exploits-from-the-dark-web-to-known-vulnerabilities-for-proactive-cyber-threat-intelligence-an-attention-based-deep-structured-semantic-model.html>.

- [34] Julia Buxton and Tim Bingham. The Rise and Challenge of Dark Net Drug Markets. *Policy Brief*, 7(January):1–24, 2015.
- [35] Siyu He, Yongzhong He, and Mingzhe Li. Classification of illegal activities on the dark web. *ACM International Conference Proceeding Series*, Part F1483(May): 73–78, 2019. doi: 10.1145/3322645.3322691.
- [36] Massimo La Morgia, Alessandro Mei, Simone Raponi, and Julinda Stefa. Time-zone geolocation of crowds in the dark web. *Proceedings - International Conference on Distributed Computing Systems*, 2018-July(Idc):445–455, 2018. doi: 10.1109/ICDCS.2018.00051.
- [37] JOEL EBUTE; U. MITIGATION IN NIGERIA BY INSTITUTE OF GOVERNANCE AND DEVELOPMENT STUDIES. pages 1–13.
- [38] Relay search. URL <https://metrics.torproject.org/rs.html>.
- [39] 2024. URL <https://web.archive.org/web/20210502020725if/http://www.shallalist.de/Downloads/shallalist.tar.gz>.
- [40] Fighting violence against women online. URL https://www.apc.org/sites/default/files/Legal_Analysis_FINAL.pdf.
- [41] Africa digital foreign policy and diplomacy - diplo resource, 11 2022. URL <https://www.diplomacy.edu/resource/report-stronger-digital-voices-from-africa/>.
- [42] New York. Harmonizing cyberlaws and regulations: The experience of the east african community. URL https://au.int/sites/default/files/newsevents/workingdocuments/27223-wd-harmonizing_cyberlaws_regulations_the_experience_of_eacl.pdf.
- [43] A.J. Mwiburi. *Preventing and Combating Cybercrime in East Africa: Lessons from Europe’s Cybercrime Frameworks*. Schriften zum Strafrechtsvergleich. Duncker & Humblot, 2019. ISBN 9783428156016. URL <https://books.google.com.et/books?id=iz9zvWEACAAJ>.
- [44] Council of Europe . Parties/observers to the budapest convention and observer organisations to the t-cy. URL <https://www.coe.int/en/web/cybercrime/parties-observers>.

-
- [45] A. L. T. Team. Africa: Au's malabo convention set to enter force after nine years, 05 2023. URL <https://dataprotection.africa/malabo-convention-set-to-enter-force/>.
- [46] Untc, 2009. URL https://treaties.un.org/Pages/ViewDetails.aspx?src=TREATY&mdsg_no=VI-19&chapter=6&clang=en.

Appendix A

Random sampling script

```
import os
from datetime import datetime
import shutil
import random

cd = "./"

folder_names = ["wireshark_capture1", "wireshark_capture2", "wireshark_capture3", "wireshark_cap
file1 = open('info.txt', 'a')

for folder_name in folder_names:
    folder_path = os.path.join(cd, folder_name)

    if os.path.exists(folder_path):
        print(f"Analysis for folder '{folder_name}':")

        files = os.listdir(folder_path)

        file_count_per_date = {}

        for file in files:
            date_str = file.split('_')[-1].split('.')[0]

            if len(date_str) == 14:
                date_format = "%Y%m%d%H%M%S"
            elif len(date_str) == 8:
                date_format = "%Y%m%d"
            else:
                print(f"Ignoring file '{file}' in folder '{folder_name}' due to invalid date for
                continue
```

```

    date_obj = datetime.strptime(date_str, date_format)
    formatted_date = date_obj.strftime("%d/%m/%Y")

    if formatted_date in file_count_per_date:
        file_count_per_date[formatted_date].append(file)
    else:
        file_count_per_date[formatted_date] = [file]

    # print(file_count_per_date)

print("Date\t\tFile Count")
print("-----")
for date, count in sorted(file_count_per_date.items(), key=lambda x: datetime.strptime(x
    # print(f"{date}\t{len(count)}")
    file1.write(f"{date}\t{len(count)}\n")

random25_folder_path = os.path.join(cd, "random25")
os.makedirs(random25_folder_path, exist_ok=True)

# 5%
for date, files in file_count_per_date.items():
    num_files_to_move = round(len(files) * 0.05)
    # print(num_files_to_move)
    files_to_move = random.sample(files, num_files_to_move)
    for file_to_move in files_to_move:
        src_path = os.path.join(folder_path, file_to_move)
        dest_path = os.path.join(random25_folder_path, file_to_move)
        shutil.move(src_path, dest_path)
    # print(f"Moved '{file_to_move}' from folder '{folder_name}' to 'random25' folder.")

print()

else:
    print(f"Folder '{folder_name}' does not exist.")

```

Appendix B

Pcap Analyzer script

```
import os
import argparse
import logging
from geoip2 import database
from datetime import datetime
from pymongo import MongoClient
import multiprocessing as mp
import time
import pyshark

GEOLITE_DB_PATH = './geo_db/GeoLite2-Country.mmdb'
MONGODB_HOST = 'localhost'
MONGODB_PORT = 27017
MONGODB_DB_NAME = 'pcap_db'
MONGODB_COLLECTION_NAME = 'random05'
MONGODB_USERNAME = 'admin'
MONGODB_PASSWORD = 'tata2116'

LOG_DIR = 'logs'
LOG_FILE = os.path.join(LOG_DIR, 'pcap_processing.log')

if not os.path.exists(LOG_DIR):
    os.makedirs(LOG_DIR)

logging.basicConfig(filename=LOG_FILE, level=logging.INFO, format='%(asctime)s - %(message)s')

def connect_to_database():
    client = MongoClient(
        f'mongodb://{MONGODB_USERNAME}:{MONGODB_PASSWORD}@{MONGODB_HOST}:{MONGODB_PORT}',
        connect=False
```

```

    )
    return client[MONGODB_DB_NAME]

def ensure_collection_exists(db):
    if MONGODB_COLLECTION_NAME not in db.list_collection_names():
        logging.info("Collection '%s' doesn't exist. Creating...", MONGODB_COLLECTION_NAME)
        collection = db.create_collection(MONGODB_COLLECTION_NAME)
        logging.info("Collection '%s' Created Successfully, Processing PCAP...", MONGODB_COLLECTION_NAME)
    else:
        collection = db[MONGODB_COLLECTION_NAME]

    index_fields = [
        ("timestamp", 1),
        ("source_ip", 1),
        ("destination_ip", 1),
        ("source_port", 1),
        ("destination_port", 1),
        ("protocol", 1),
        ("TLS_version", 1),
        ("source_mac", 1),
        ("destination_mac", 1),
        ("src_country_iso_code", 1),
        ("src_country_name", 1),
        ("dst_country_iso_code", 1),
        ("dst_country_name", 1)
    ]

    for field, direction in index_fields:
        collection.create_index([(field, direction)])

def process_pcap_file(pcap_file):
    logging.info("Starting processing %s", pcap_file)
    print(f"Starting processing {pcap_file}")
    reader = database.Reader(GEOLITE_DB_PATH)
    cap = pyshark.FileCapture(pcap_file)
    data_to_insert = []
    db = connect_to_database()
    ensure_collection_exists(db)

    for packet in cap:
        if 'IP' in packet:
            unix_timestamp = packet.sniff_time.timestamp()
            timestamp = datetime.fromtimestamp(unix_timestamp)
            src_ip = packet.ip.src
            dst_ip = packet.ip.dst
            protocol = packet.highest_layer
            src_mac = packet.eth.src
            dst_mac = packet.eth.dst

```

```
TLS_version = None
if packet.transport_layer:
    src_port = packet[packet.transport_layer].srcport
    dst_port = packet[packet.transport_layer].dstport
try:
    if 'tls' in packet:
        tls_layer = packet.tls
        TLS_version = tls_layer.record_version
except AttributeError:
    pass
try:
    response_src = reader.country(src_ip)
    src_country_iso_code = response_src.country.iso_code
    src_country_name = response_src.country.name
except Exception:
    src_country_iso_code = "PRIV"
    src_country_name = "PRIVATE_IP"
try:
    response_dst = reader.country(dst_ip)
    dst_country_iso_code = response_dst.country.iso_code
    dst_country_name = response_dst.country.name
except Exception:
    src_country_iso_code = "PRIV"
    src_country_name = "PRIVATE_IP"
data = {
    "timestamp": timestamp,
    "source_ip": src_ip,
    "destination_ip": dst_ip,
    "source_port": src_port,
    "destination_port": dst_port,
    "protocol": protocol,
    "TLS_version": TLS_version,
    "source_mac": src_mac,
    "destination_mac": dst_mac,
    "src_country_iso_code": src_country_iso_code,
    "src_country_name": src_country_name,
    "dst_country_iso_code": dst_country_iso_code,
    "dst_country_name": dst_country_name
}

data_to_insert.append(data)

if len(data_to_insert) >= 4000:
    insert_into_database(data_to_insert, db)
    data_to_insert = []

cap.close()
logging.info("Processing completed for %s", pcap_file)
```

```
def insert_into_database(data, db):
    collection = db[MONGODB_COLLECTION_NAME]
    chunk_size = 4000
    for i in range(0, len(data), chunk_size):
        collection.insert_many(data[i:i + chunk_size])

def main():
    parser = argparse.ArgumentParser(description='Process PCAP files and save to MongoDB database')
    parser.add_argument('-i', '--input', type=str, help='Input folder containing PCAP files')
    args = parser.parse_args()
    input_folder = args.input
    if not args.input:
        parser.error('Please provide the input folder path.')

    if not os.path.exists(args.input):
        parser.error(f'Input folder {args.input} does not exist.')

    pcap_files = [os.path.join(input_folder, file) for file in os.listdir(input_folder)
                   if file.endswith((".pcap", ".pcapng", ""))]

    num_processes = mp.cpu_count() * 2
    pool = mp.Pool(processes=num_processes)
    pool.map(process_pcap_file, pcap_files)
    pool.close()
    pool.join()

if __name__ == "__main__":
    main()
```

Appendix C

MongoDB to Elastic connector script

```
import os
import logging
from elasticsearch import Elasticsearch
import pymongo
from concurrent.futures import ThreadPoolExecutor
from datetime import datetime

log_folder = "logs"
if not os.path.exists(log_folder):
    os.makedirs(log_folder)

logging.basicConfig(level=logging.INFO)
log_file = os.path.join(log_folder, "logfile.log")
handler = logging.FileHandler(log_file)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
handler.setFormatter(formatter)
logger = logging.getLogger(__name__)
logger.addHandler(handler)

def connect_to_mongodb(mongodb_uri, db_name, collection_name, username, password):
    try:
        client = pymongo.MongoClient(mongodb_uri, username=username, password=password)
        db = client[db_name]
        collection = db[collection_name]
        return collection
    except Exception as e:
        logger.error(f"Error connecting to MongoDB: {e}")
        return None
```

```

def process_mongodb_documents(collection, skip=0, limit=30000):
    try:
        processed_documents = []
        for doc in collection.find().skip(skip).limit(limit):
            processed_doc = {
                'source_ip': doc['source_ip'],
                'destination_ip': doc['destination_ip'],
                'source_port': doc['source_port'],
                'destination_port': doc['destination_port'],
                'protocol': doc['protocol'],
                'timestamp': doc['timestamp'],
                'src_country_iso_code': doc['src_country_iso_code'],
                'src_country_name': doc['src_country_name'],
                'dst_country_iso_code': doc['dst_country_iso_code'],
                'dst_country_name': doc['dst_country_name'],
                'TLS_version': doc['TLS_version']
            }
            processed_documents.append(processed_doc)
        return processed_documents
    except Exception as e:
        logger.error(f"Error processing MongoDB documents: {e}")
        return []

def connect_to_elasticsearch(elasticsearch_host, username, password, cert_fingerprint, es_index_name):
    try:
        client = Elasticsearch(
            elasticsearch_host,
            ssl_assert_fingerprint=cert_fingerprint,
            basic_auth=(username, password)
        )
        if not client.indices.exists(index=es_index_name):
            mapping = {
                "mappings": {
                    "properties": {
                        "source_ip": {"type": "ip"},
                        "destination_ip": {"type": "ip"},
                        "source_port": {"type": "keyword"},
                        "destination_port": {"type": "keyword"},
                        "protocol": {"type": "keyword"},
                        "timestamp": {"type": "keyword"},
                        "src_country_iso_code": {"type": "keyword"},
                        "src_country_name": {"type": "keyword"},
                        "dst_country_iso_code": {"type": "keyword"},
                        "dst_country_name": {"type": "keyword"},
                        "TLS_version": {"type": "keyword"}
                    }
                }
            }
    
```

```

    }

    client.indices.create(index=es_index_name, body=mapping)
    logger.info(f"Created index: {es_index_name} with mapping")
else:
    logger.info(f"Index {es_index_name} already exists")

    return client
except Exception as e:
    logger.error(f"Error connecting to Elasticsearch: {e}")
    return None

def forward_to_elasticsearch(es_client, es_index_name, documents):
    try:
        bulk_data = []
        for doc in documents:
            bulk_data.append({
                "index": {
                    "_index": es_index_name,
                }
            })
        bulk_data.append(doc)

        response = es_client.bulk(index=es_index_name, body=bulk_data, refresh=True)

        if response['errors']:
            logger.error(f"Bulk indexing error: {response}")
        else:
            logger.info(f"Indexed {len(documents)} documents to Elasticsearch")

    except Exception as e:
        logger.error(f"Error indexing data to Elasticsearch: {e}")

def process_batch(args):
    collection, skip, limit = args
    return process_mongodb_documents(collection, skip=skip, limit=30000)

if __name__ == '__main__':
    mongodb_uri = "mongodb://0.0.0.0:27017/"
    db_name = "pcap_db"
    collection_name = "random05"
    es_index_name = "search-pcap-random-analyzer"
    mongodb_username = "admin"
    mongodb_password = "tata2116"

    es_client = connect_to_elasticsearch("https://0.0.0.0:9200", "elastic", "n8u0WneEI180KQhkr0q")
    if es_client:
        collection = connect_to_mongodb(mongodb_uri, db_name, collection_name, mongodb_username,

```

```
if collection is not None:
    total_documents = 259000000
    batch_size = 30000
    current_skip = 0

    while current_skip < total_documents:
        documents = process_mongodb_documents(collection, skip=current_skip, limit=batch_size)
        forward_to_elasticsearch(es_client, es_index_name, documents)
        current_skip += batch_size
```

Appendix D

Cleaning unfinished JSON

```
import re
import json
import os
from multiprocessing import Process

pattern = r'\{\s*_id":\s*("[^"]*"|{[^{}]*}),\s*"timestamp":\s*("[^"]*"|{[^{}]*}),\s*"source_ip"

def process_chunk(chunk, chunk_num):
    filtered_data = []
    matches = re.findall(pattern, chunk)
    for match in matches:
        obj = {
            "_id": json.loads(match[0]),
            "timestamp": json.loads(match[1]),
            "source_ip": json.loads(match[2]),
            "destination_ip": json.loads(match[3]),
            "source_port": json.loads(match[4]),
            "destination_port": json.loads(match[5]),
            "protocol": json.loads(match[6]),
            "TLS_version": json.loads(match[7]),
            "source_mac": json.loads(match[8]),
            "destination_mac": json.loads(match[9]),
            "src_country_iso_code": json.loads(match[10]),
            "src_country_name": json.loads(match[11]),
            "dst_country_iso_code": json.loads(match[12]),
            "dst_country_name": json.loads(match[13])
        }
        filtered_data.append(obj)
    with open(f'filtered_output_{chunk_num}.json', 'w') as outfile:
        json.dump(filtered_data, outfile, indent=2)
```

```
def main():
    chunk_size = 1000000
    num_processes = 4
    processes = []
    with open('output.json', 'r') as infile:
        chunk_num = 1
        while True:
            chunk = infile.read(chunk_size)
            if not chunk:
                break
            p = Process(target=process_chunk, args=(chunk, chunk_num))
            p.start()
            processes.append(p)
            chunk_num += 1

            if len(processes) >= num_processes:
                for proc in processes:
                    proc.join()
                processes = []

        for p in processes:
            p.join()

    print("Filtering completed. The filtered data has been saved to files.")

if __name__ == "__main__":
    main()
```

Appendix E

Filtering unique destination IP script

```
import os
import json
from concurrent.futures import ProcessPoolExecutor
import multiprocessing

def filter_duplicates(data_chunk, attributes, seen, lock):
    unique_data = []
    duplicates = []
    for item in data_chunk:
        key = tuple(item[attr] for attr in attributes)
        with lock:
            if key not in seen:
                seen[key] = None
                unique_data.append(item)
            else:
                duplicates.append(item)
    return unique_data, duplicates

def process_file(file_path, attributes, seen, lock):
    with open(file_path) as f:
        data = json.load(f)

    chunk_size = len(data) // 15 # Dividing data into 5 chunks for multiprocessing

    final_result = []
    duplicates = []

    with ProcessPoolExecutor(max_workers=15) as executor: # Set max_workers to 5
```

```
futures = []
for i in range(0, len(data), chunk_size):
    data_chunk = data[i:i + chunk_size]
    future = executor.submit(filter_duplicates, data_chunk, attributes, seen, lock)
    futures.append(future)

for future in futures:
    unique_data_chunk, duplicates_chunk = future.result()
    final_result.extend(unique_data_chunk)
    duplicates.extend(duplicates_chunk)

return final_result

def main():
    input_folder = 'input_files'
    output_file_path = 'input/unique.json'
    attributes = ['destination_ip']
    manager = multiprocessing.Manager()
    seen = manager.dict() # Use Manager().dict() instead of Manager().list()
    lock = manager.Lock() # Use Manager().Lock() for synchronization

    all_unique_data = []

    for filename in os.listdir(input_folder):
        if filename.endswith('.json'):
            file_path = os.path.join(input_folder, filename)
            unique_data = process_file(file_path, attributes, seen, lock)
            all_unique_data.extend(unique_data)

    with open(output_file_path, 'w') as f:
        json.dump(all_unique_data, f, indent=2)

if __name__ == '__main__':
    main()
```

Appendix F

Filter All African countries script

```
import os
import json
from concurrent.futures import ProcessPoolExecutor
import multiprocessing

ALL_AFRICA_COUNTRIES = [
    'Burundi', 'Comoros', 'Djibouti', 'Eritrea', 'Ethiopia', 'Kenya', 'Madagascar', 'Malawi', 'Mali', 'Mozambique',
    'South Sudan', 'Tanzania', 'Uganda', 'Zambia', 'Zimbabwe', 'Angola', 'Cameroon', 'Central African Republic',
    'Democratic Republic of the Congo', 'Equatorial Guinea', 'Gabon', 'Sao Tome and Principe', 'Sierra Leone',
    'eSwatini', 'Lesotho', 'Botswana', 'Namibia', 'South Africa', 'Benin', 'Burkina Faso', 'Cape Verde',
    'Guinea-Bissau', 'Liberia', 'Mali', 'Mauritania', 'Niger', 'Nigeria', 'Senegal', 'Sierra Leone'
]

def filter_duplicates(data_chunk, attributes, seen, lock):
    unique_data = []
    duplicates = []
    for item in data_chunk:
        if item['dst_country_name'] in ALL_AFRICA_COUNTRIES:
            key = tuple(item[attr] for attr in attributes)
            with lock:
                if key not in seen:
                    seen[key] = None
                    unique_data.append(item)
                else:
                    duplicates.append(item)
    return unique_data, duplicates

def process_file(file_path, attributes, seen, lock):
    with open(file_path) as f:
        data = json.load(f)
```

```
chunk_size = len(data) // 15 # Dividing data into 5 chunks for multiprocessing

final_result = []
duplicates = []

with ProcessPoolExecutor(max_workers=15) as executor: # Set max_workers to 5
    futures = []
    for i in range(0, len(data), chunk_size):
        data_chunk = data[i:i + chunk_size]
        future = executor.submit(filter_duplicates, data_chunk, attributes, seen, lock)
        futures.append(future)

    for future in futures:
        unique_data_chunk, duplicates_chunk = future.result()
        final_result.extend(unique_data_chunk)
        duplicates.extend(duplicates_chunk)

return final_result

def main():
    input_folder = 'input'
    output_file_path = 'input/unique-ALL_Africa.json'
    attributes = ['destination_ip']
    manager = multiprocessing.Manager()
    seen = manager.dict() # Use Manager().dict() instead of Manager().list()
    lock = manager.Lock() # Use Manager().Lock() for synchronization

    all_unique_data = []

    for filename in os.listdir(input_folder):
        if filename.endswith('.json'):
            file_path = os.path.join(input_folder, filename)
            unique_data = process_file(file_path, attributes, seen, lock)
            all_unique_data.extend(unique_data)

    with open(output_file_path, 'w') as f:
        json.dump(all_unique_data, f, indent=2)

if __name__ == '__main__':
    main()
```

Appendix G

DNS resolution script

```
import json
import socket
import os
import queue
import threading

# Open the JSON file and load its contents
with open('input/unique-ALL_Africa.json', 'r') as f:
    data = json.load(f)

# Initialize lists to store the successful and failed DNS resolutions
success = []
failure = []

# Define lock for file writing operations
file_lock = threading.Lock()

# Define the function to perform DNS resolution for a single item
def resolve_dns(item):
    try:
        # Perform DNS resolution
        result = socket.gethostbyaddr(item['destination_ip'])
        dns_resolution = result[0]

        # Add the DNS resolution to the item and append it to the success list
        item['dns_resolution'] = dns_resolution
        success.append(item)

        # Write the successful DNS resolution to a text file
        with file_lock:
            with open(os.path.join('output', 'dns_success.txt'), 'a') as f:
```

```
f.write(f"{dns_resolution}\n")

except socket.herror:
    # Append the item to the failure list
    failure.append(item)

    # Write the failed DNS resolution to a text file
    with file_lock:
        with open(os.path.join('output', 'dns_fail.txt'), 'a') as f:
            f.write(f"{item['destination_ip']}\n")

# Define the function to run the DNS resolution in a worker thread
def worker_thread():
    while True:
        # Get the next item from the queue
        item = q.get()

        # Perform DNS resolution for the item
        resolve_dns(item)

        # Let the queue know that the item has been processed
        q.task_done()

# Create the queue and worker threads
q = queue.Queue()
for _ in range(10):
    thread = threading.Thread(target=worker_thread)
    thread.daemon = True
    thread.start()

# Add the items to the queue
for item in data:
    q.put(item)

# Wait for all the items to be processed
q.join()

# Write the successful DNS resolutions to a new JSON file
with open('output/dns_success.json', 'w') as f:
    json.dump(success, f, indent=4)

# Write the failed DNS resolutions to a new JSON file
with open('output/dns_failure.json', 'w') as f:
    json.dump(failure, f, indent=4)
```

Appendix H

Content scrapping script

```
import requests
from bs4 import BeautifulSoup
import concurrent.futures
import json
import os
from urllib.parse import urljoin, urlparse

# Function to scrape text from a given domain and its internal links
def scrape_domain_and_links(domain):
    try:
        # Use requests Session for improved performance
        with requests.Session() as session:
            # Fetch HTML content of the domain with timeout
            response = session.get(f'http://{domain}', timeout=30)
            response.raise_for_status()

            # Parse HTML using BeautifulSoup
            soup = BeautifulSoup(response.text, 'html.parser')

            # Extract text content
            text_content = soup.get_text()

            # Scrape links found in the domain
            links = soup.find_all('a', href=True)
            for link in links:
                href = link['href']
                # Check if the link is relative and construct full URL if necessary
                href = urljoin(response.url, href)

                # Parse the URL and check if it belongs to the same domain
                parsed_href = urlparse(href)
```

```

        if parsed_href.netloc == domain:
            # Fetch and parse link contents
            try:
                link_response = session.get(href, timeout=10)
                link_response.raise_for_status()
                link_soup = BeautifulSoup(link_response.text, 'html.parser')
                link_text_content = link_soup.get_text()
                # Append link text content to domain text content
                text_content += "\n\n" + link_text_content
            except requests.RequestException as e:
                print(f'Error requesting internal link {href}: {e}')

    # Save text content to a file in the output folder
    with open(os.path.join(output_folder, f'{domain}.txt'), 'w', encoding='utf-8') as f:
        # Iterate over each line in the text content and save it to the file
        for line in text_content.splitlines():
            # Check if the line contains any non-whitespace characters
            if line.strip():
                f.write(f'{line}\n')

    print(f'Successfully scraped {domain} and its internal links')
    return domain, True

except requests.RequestException as e:
    # Handle errors and log them to error files
    error_log_file = os.path.join("errors", 'error_log.txt')
    if hasattr(e, 'response') and e.response is not None:
        status_code = e.response.status_code
        error_data = f'{domain}: Status Code {status_code}\n'
        with open(error_log_file, 'a') as error_log:
            error_log.write(error_data)
        print(f'Error requesting {domain}: Status Code {status_code}')
    else:
        error_data = f'{domain}: {e}\n'
        with open(error_log_file, 'a') as error_log:
            error_log.write(error_data)
        print(f'Error requesting {domain}: {e}')
    return domain, False

# Function to scrape multiple domains concurrently
def scrape_domains_multithreaded(domains):
    results = []
    # Specify the number of worker threads to be 6
    with concurrent.futures.ThreadPoolExecutor(max_workers=6) as executor:
        futures = [executor.submit(scrape_domain_and_links, domain) for domain in domains]

        # Wait for all futures to complete
        for future in concurrent.futures.as_completed(futures):

```

```
        try:
            result = future.result()
            results.append(result)
        except Exception as e:
            print(f'Error: {e}')

    return results

# File containing domain names
input_file = 'output/dns_success.txt'

# Create a folder to store output files if it doesn't exist
output_folder = 'domain_output'
os.makedirs(output_folder, exist_ok=True)

# Read domain names from the input file
with open(input_file, 'r') as file:
    domains = file.read().splitlines()

# Scrape domains concurrently
scrape_results = scrape_domains_multithreaded(domains)

# Split results into successful and failed
successful_domains = [result[0] for result in scrape_results if result[1]]
failed_domains = [{"domain": result[0]} for result in scrape_results if not result[1]]

# Save results to JSON files
with open('analyze/input/successful_scrape.json', 'w') as success_file:
    json.dump(successful_domains, success_file)

with open('analyze/input/failed_scrape.json', 'w') as fail_file:
    json.dump(failed_domains, fail_file)
```

Appendix I

Scrapping success script

```
import json

def filter_domains(ip_json_file, json_file, output_file):
    # Read IPs from the JSON file
    with open(ip_json_file, 'r') as f:
        ip_data = json.load(f)
        ips = ip_data # If ip_data is a list, directly assign it to ips

    # Read JSON data
    with open(json_file, 'r') as f:
        data = json.load(f)

    filtered_data = []
    for item in data:
        if 'dns_resolution' in item:
            domain = item['dns_resolution']
            if domain in ips:
                filtered_data.append(item)

    # Writing the selected class data to the output file
    with open(output_file, 'w') as f:
        json.dump(filtered_data, f, indent=4)

# Example usage:
ip_json_file = 'analyze/input/successful_scrape.json'
json_file = 'output/dns_success.json'
output_file = 'analyze/output/scrape_output.json'
filter_domains(ip_json_file, json_file, output_file)
```

Appendix J

Scrapping Fialed script

```
import json

def filter_domains(ip_json_file, json_file, output_file):
    # Read IPs from the JSON file
    with open(ip_json_file, 'r') as f:
        ip_data = json.load(f)
        ips = {entry['domain']: entry['status_code'] for entry in ip_data}

    # Read JSON data
    with open(json_file, 'r') as f:
        data = json.load(f)

    filtered_data = []
    for item in data:
        domain = item.get('dns_resolution') # Accessing 'domain' key safely
        if domain in ips:
            item['status_code'] = ips[domain]
            filtered_data.append(item)

    # Writing the selected class data to the output file
    with open(output_file, 'w') as f:
        json.dump(filtered_data, f, indent=4)

# Example usage:
ip_json_file = 'analyze/input/failed_scrape.json'
json_file = 'output/dns_success.json'
output_file = 'analyze/output/fail_output.json'
filter_domains(ip_json_file, json_file, output_file)
```

Appendix K

Text matching script – main

```
import os
import json
import logging
import torch.multiprocessing as mp
from torch.multiprocessing import Pool, set_start_method
import preprocessing as pre
import cosign as cos
import tf_idf as tf_idf
import minilm as minilm

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

error_log = logging.FileHandler('./log/error.log')
error_log.setLevel(logging.ERROR)
error_formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
error_log.setFormatter(error_formatter)
logging.getLogger('').addHandler(error_log)

success_log = logging.FileHandler('./log/success.log')
success_log.setLevel(logging.INFO)
success_formatter = logging.Formatter('%(asctime)s - %(message)s')
success_log.setFormatter(success_formatter)
logging.getLogger('').addHandler(success_log)

scrapped_dir = './scrapped-All_Africa'
minilm_lang_dir = './lang-corpus-1'
tf_IDF_dir = './corpus-tf/cleaned_tf'
clean_scrapp = './output/cleaned'
output_dir = './output'
success_log = './success.json'
```

```

def process_dns_resolution(json_data):
    dns_resolution = json_data.get('dns_resolution')
    if dns_resolution:
        logging.info(f"Successfully processed DNS resolution: {dns_resolution}")
        dns_filename = os.path.join(scraped_dir, dns_resolution + '.txt')
        if os.path.exists(dns_filename):
            pre.clean_file(dns_filename)
        else:
            logging.error(f"File {dns_resolution}.txt not found.")
    else:
        logging.error("DNS resolution not provided in JSON data.")

def calculate_similarity():
    tf_labeled_dir = '/home/darkweb/Desktop/Text_matching/final_detector/corpus-tf/cleaned_tf'
    lang_labeled_dir = '/home/darkweb/Desktop/Text_matching/final_detector/lang-corpus-1'
    tf_idf.get_tfidf_similarity(clean_scrapp, tf_labeled_dir)
    print("tf_idf Result saved to output directory")
    cos.get_cosine_similarity(clean_scrapp, tf_labeled_dir)
    print("cosign Result saved to output directory")
    minilm.get_minilm_similarity(clean_scrapp, lang_labeled_dir)

    print("minilm Result saved to output directory")

if __name__ == '__main__':
    try:
        output_file_path = os.path.join(output_dir, 'result.json')
        if not os.path.exists(output_file_path):
            with open(output_file_path, 'w') as f:
                json.dump([], f)

        with open(success_log, 'r', encoding="utf-8", errors="replace") as json_file:
            json_data_list = json.load(json_file)

        set_start_method('spawn', force=True)

        num_processes = mp.cpu_count() * 2
        with mp.Pool(processes=num_processes) as pool:
            results = pool.starmap(process_dns_resolution, [(data,) for data in json_data_list])

            similarity_result = pool.apply(calculate_similarity)
            pool.close()
            pool.join()

    except Exception as e:
        logging.error(f"An error occurred: {e}")

```

Appendix L

Text matching script – preprocessing

```
import os
import chardet
import multiprocessing
import re
import nltk

from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')

def text_to_vector(text):
    words = WORD.findall(text)
    return Counter(words)

def detect_encoding(file_path):
    with open(file_path, 'rb') as f:
        rawdata = f.read()
    encoding = chardet.detect(rawdata)['encoding']
    return encoding

def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s.,;:!?]', '', text)
```

```
text = re.sub(r'\s+', ' ', text)
return text

def tokenize_text(text):
    tokens = word_tokenize(text)
    return tokens

def lemmatize_tokens(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return lemmatized_tokens

def extract_keywords(text):
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    cleaned_text = " ".join(lemmatized_tokens)
    return cleaned_text

def clean_file(input_file):
    file_name = os.path.basename(input_file)
    output_directory = './output/cleaned'
    os.makedirs(output_directory, exist_ok=True)
    output_filename_cleaned = os.path.join(output_directory, os.path.splitext(file_name)[0] + '_')

    encoding = detect_encoding(input_file)
    with open(input_file, 'r', encoding=encoding, errors='ignore') as txtfile:
        content = txtfile.read()
        cleaned_text = clean_text(content)
        keywords = extract_keywords(cleaned_text)
        with open(output_filename_cleaned, 'w', encoding='utf-8') as txt_out:
            txt_out.write(keywords)

def clean_files_in_directory(input_directory, output_directory):
    files_to_clean = []
    for root, dirs, files in os.walk(input_directory):
        for file in files:
            if file.endswith('_lang.txt'):
                files_to_clean.append(os.path.join(root, file))

    num_processes = multiprocessing.cpu_count()
    with multiprocessing.Pool(processes=num_processes) as pool:
        pool.starmap(clean_file, [(file, output_directory) for file in files_to_clean])
```

Appendix M

Text matching script – TF-IDF Algorithm

```
import os
import json
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

nltk.download('stopwords')

def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(filtered_words)

def calculate_tfidf_similarity(text1, text2):
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([text1, text2])
    cosine_sim = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
    return cosine_sim[0][1]

def get_label_from_filename(file_name):
    return os.path.splitext(file_name.replace("_cleaned", ""))[0].split('_')[0]

def save_results_to_json(results, output_dir, batch_count):
    output_file = os.path.join(output_dir, f'result_tfidf_batch_{batch_count}.json')
    with open(output_file, 'w') as outfile:
```

```

        json.dump(results, outfile, indent=4)

def get_tfidf_similarity(cleaned_dir, labeled_dir, output_dir="./output/result_tfidf"):
    results = {}
    batch_size = 130
    batch_count = 1

    labeled_files = {}
    for root, dirs, files in os.walk(labeled_dir):
        for file in files:
            if file.endswith(".txt"):
                with open(os.path.join(root, file), 'r', encoding='utf-8') as labeled_file:
                    labeled_files[file] = labeled_file.read()

    for cleaned_file_name in os.listdir(cleaned_dir):
        if cleaned_file_name.endswith(".txt"):
            cleaned_file_path = os.path.join(cleaned_dir, cleaned_file_name)
            with open(cleaned_file_path, 'r', encoding='utf-8') as cleaned_file:
                cleaned_text = cleaned_file.read()

            if not cleaned_text.strip():
                print(f"Empty file: {cleaned_file_name}. Skipping...")
                continue

            cleaned_text = remove_stopwords(cleaned_text)

            top_matches = []
            for labeled_file, labeled_text in labeled_files.items():
                try:
                    similarity = calculate_tfidf_similarity(cleaned_text, labeled_text)
                except Exception as e:
                    print(f"Error calculating similarity for {cleaned_file_name} and {labeled_file}")
                    continue
                label = get_label_from_filename(labeled_file)
                top_matches.append({'label': label, 'similarity': similarity})
            top_matches.sort(key=lambda x: x['similarity'], reverse=True)
            results[cleaned_file_name] = top_matches[:3]

            if len(results) == batch_size:
                save_results_to_json(results, output_dir, batch_count)
                results = {}
                batch_count += 1

    if results:
        save_results_to_json(results, output_dir, batch_count)

```

Appendix N

Text matching script – Cosign Algorithm

```
import os
import json
import math
from collections import Counter
from langdetect import detect
import logging
import gc

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

def get_cosign(vec1, vec2):
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])

    sum1 = sum([vec1[x] ** 2 for x in list(vec1.keys())])
    sum2 = sum([vec2[x] ** 2 for x in list(vec2.keys())])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator

def preprocess_text(text):
    return text.lower().split()

def get_text_vector(text):
```

```

    return Counter(preprocess_text(text))

def calculate_cosine_similarity(text1, text2):
    vector1 = get_text_vector(text1)
    vector2 = get_text_vector(text2)

    return get_cosign(vector1, vector2)

def get_label_from_filename(file_name):
    return os.path.splitext(file_name)[0].split('_')[0]

def save_results_to_json(results, output_dir, batch_count):
    output_file = os.path.join(output_dir, f'result_cosine_batch_{batch_count}.json')
    with open(output_file, 'w') as outfile:
        json.dump(results, outfile, indent=4)

def get_cosine_similarity(cleaned_dir, labeled_dir, output_dir="./output/result_cosine"):
    results = {}
    batch_size = 130
    batch_count = 1

    labeled_files = {}
    for root, dirs, files in os.walk(labeled_dir):
        for file in files:
            if file.endswith(".txt"):
                with open(os.path.join(root, file), 'r', encoding='utf-8') as labeled_file:
                    labeled_files[file] = labeled_file.read()

    for cleaned_file_name in os.listdir(cleaned_dir):
        if cleaned_file_name.endswith(".txt"):
            cleaned_file_path = os.path.join(cleaned_dir, cleaned_file_name)
            with open(cleaned_file_path, 'r', encoding='utf-8') as cleaned_file:
                cleaned_text = cleaned_file.read()

            if not cleaned_text.strip():
                print(f"Empty file: {cleaned_file_name}. Skipping...")
                continue

            top_matches = []
            for labeled_file, labeled_text in labeled_files.items():
                try:
                    similarity = calculate_cosine_similarity(cleaned_text, labeled_text)
                except Exception as e:
                    print(f"Error calculating similarity for {cleaned_file_name} and {labeled_file}")
                    continue

                label = get_label_from_filename(labeled_file)
                top_matches.append({'label': label, 'similarity': similarity})
            top_matches.sort(key=lambda x: x['similarity'], reverse=True)

```

```
results[cleaned_file_name] = top_matches[:3]

if len(results) == batch_size:
    save_results_to_json(results, output_dir, batch_count)
    results = {}
    batch_count += 1

if results:
    save_results_to_json(results, output_dir, batch_count)

# Clear resources
for file_object in (labeled_files, top_matches, results):
    if hasattr(file_object, 'close'):
        file_object.close()

del labeled_files, top_matches, results

gc.collect()
```

Appendix O

Text matching script – Mini-LM Algorithm

```
import os
import json
from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')

def calculate_minilm_similarity(text1, text2):
    embedding1 = model.encode(text1, convert_to_tensor=True)
    embedding2 = model.encode(text2, convert_to_tensor=True)
    similarity = float(util.pytorch_cos_sim(embedding1, embedding2))
    return similarity

def get_label_from_filename(file_name):
    return os.path.splitext(file_name)[0].split('_')[0]

def save_results_to_json(results, output_file):
    if os.path.exists(output_file):
        with open(output_file, 'r') as infile:
            existing_data = json.load(infile)
            existing_data.update(results)
            results = existing_data

    with open(output_file, 'w') as outfile:
        json.dump(results, outfile, indent=4)

def get_minilm_similarity(cleaned_dir, labeled_dir, output_dir="./output/result_minilm"):
    batch_size = 6
    batch_count = 1
```

```

labeled_files = {}
for root, dirs, files in os.walk(labeled_dir):
    for file in files:
        if file.endswith(".txt"):
            with open(os.path.join(root, file), 'r', encoding='utf-8') as labeled_file:
                labeled_files[file] = labeled_file.read()

results = {}
for cleaned_file_name in os.listdir(cleaned_dir):
    if cleaned_file_name.endswith(".txt"):
        cleaned_file_path = os.path.join(cleaned_dir, cleaned_file_name)
        with open(cleaned_file_path, 'r', encoding='utf-8') as cleaned_file:
            cleaned_text = cleaned_file.read()

        if not cleaned_text.strip():
            print(f"Empty file: {cleaned_file_name}. Skipping...")
            continue

        top_matches = []
        for labeled_file, labeled_text in labeled_files.items():
            try:
                similarity = calculate_minilm_similarity(cleaned_text, labeled_text)
            except Exception as e:
                print(f"Error calculating similarity for {cleaned_file_name} and {labeled_file}")
                continue

            label = get_label_from_filename(labeled_file)
            top_matches.append({'label': label, 'similarity': similarity})
        top_matches.sort(key=lambda x: x['similarity'], reverse=True)
        results[cleaned_file_name] = top_matches[:3]

    output_file = os.path.join(output_dir, f'result_minilm_batch_{batch_count}.json')
    save_results_to_json({cleaned_file_name: top_matches[:3]}, output_file)

    if len(results) == batch_size:
        batch_count += 1
        results = {}

if results:
    output_file = os.path.join(output_dir, f'result_minilm_batch_{batch_count}.json')
    save_results_to_json(results, output_file)

```

Appendix P

Text matching script – merge result

```
import json
import os
import logging
from concurrent.futures import ThreadPoolExecutor

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

def process_folder(folder_path, success_dns_resolution):
    result_data = {}
    for filename in os.listdir(folder_path):
        if filename.endswith('.json'):
            with open(os.path.join(folder_path, filename), 'r') as processed_file:
                processed_data = json.load(processed_file)
                for key, value in processed_data.items():
                    dns_resolution = key.split('_')[0]
                    if dns_resolution == success_dns_resolution:
                        result_data[key.replace('_cleaned.txt', '')] = value
    return result_data

def main():
    with open('success.json', 'r') as success_file:
        success_entries = json.load(success_file)

    all_entries = []

    for success_data in success_entries:
        success_dns_resolution = success_data['dns_resolution']
```

```
result_types = ['tfidf', 'minilm', 'cosine']
combined_results = {}

for result_type in result_types:
    folder_path = os.path.join('output', f'result_{result_type}')
    combined_results[result_type] = process_folder(folder_path, success_dns_resolution)

success_data['processed_results'] = combined_results
all_entries.append(success_data)

with open("processed_result.json", 'w') as combined_success_file:
    json.dump(all_entries, combined_success_file, indent=4)

logging.info("All success entries with processed results have been saved to Combined_Success")

if __name__ == "__main__":
    main()
```

Appendix Q

Shallalist – Scrapping script

```
import logging
import os
from urllib.parse import urljoin, urlparse
import requests
from bs4 import BeautifulSoup
import concurrent.futures
import gc

# Set up logging
logging.basicConfig(filename='scraping_errors.log', level=logging.ERROR, format='%(asctime)s - %

def save_content(url, content, output_folder, root_folder, input_file_path):
    """Save the scraped content to a text file."""
    parsed_url = urlparse(url)

    # Determine the relative path from the root folder to the input file
    relative_path = os.path.relpath(input_file_path, root_folder)
    file_dir = os.path.dirname(relative_path)

    # Create the directory structure in the output folder including domain name
    save_folder = os.path.join(output_folder, file_dir)
    os.makedirs(save_folder, exist_ok=True)

    # Use the domain for the filename
    file_name = os.path.join(save_folder, f"{parsed_url.netloc}.txt")

    # Prepare the content to be appended
    path_parts = parsed_url.path.strip('/')
    header = f"URL: {url}\nPath: {path_parts}\n{'='*80}\n"

    with open(file_name, 'a', encoding='utf-8') as file:
```

```

        file.write(header)
        file.write(content)
        file.write('\n\n') # Add extra new lines for separation between contents

def scrape_url(session, url, visited_urls, output_folder, root_folder, input_file_path):
    """Scrape a single URL and save its content."""
    if url in visited_urls:
        return
    try:
        response = session.get(url, timeout=10)
        response.raise_for_status()
        soup = BeautifulSoup(response.content, 'html.parser')
        text_content = soup.get_text(separator='\n', strip=True)
        save_content(url, text_content, output_folder, root_folder, input_file_path)
        visited_urls.add(url)
    except Exception as e:
        logging.error(f"Failed to scrape {url}: {e}")

def scrape_domain_and_links(domain, root_folder, input_file_path, visited_urls=None, output_folder=None):
    """Scrape the main page of a domain and its internal links."""
    if visited_urls is None:
        visited_urls = set()

    with requests.Session() as session:
        session.headers.update({'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4399.72 Safari/537.36'})

        domain_url = f"http://{domain}" if not domain.startswith(('http://', 'https://')) else domain
        try:
            response = session.get(domain_url, timeout=10)
            response.raise_for_status()
            soup = BeautifulSoup(response.content, 'html.parser')
            links = {urljoin(domain_url, a['href']) for a in soup.find_all('a', href=True) if urljoin(domain_url, a['href']) != domain_url}

            # Scrape main page
            scrape_url(session, domain_url, visited_urls, output_folder, root_folder, input_file_path)

            # Recursively scrape internal links
            for link in links:
                if link not in visited_urls:
                    scrape_url(session, link, visited_urls, output_folder, root_folder, input_file_path)
        except Exception as e:
            logging.error(f"Failed to scrape {domain_url}: {e}")

    # Clear memory after scraping a domain
    visited_urls.clear()
    gc.collect()

```

```
def scrape_domains_multithreaded(domains, root_folder, output_folder='scraped_content'):
    """Scrape multiple domains concurrently using threading."""
    with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
        futures = [executor.submit(scrape_domain_and_links, domain, root_folder, input_file_path)
                    for domain, input_file_path in domains.items()]
        for future in concurrent.futures.as_completed(futures):
            try:
                future.result()
            except Exception as e:
                logging.error(f"Error in concurrent scraping: {e}")

def get_input_files(root_folder):
    """Recursively search the root_folder for files."""
    input_files = []
    for subdir, _, files in os.walk(root_folder):
        for file in files:
            input_files.append((subdir, os.path.join(subdir, file)))
    return input_files

def read_domains_from_file(file_path):
    """Generator to read domains from a file line by line."""
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            for line in file:
                domain = line.strip()
                if domain:
                    yield domain
    except UnicodeDecodeError as e:
        logging.error(f"UnicodeDecodeError while reading {file_path}: {e}")
    except FileNotFoundError as e:
        logging.error(f"FileNotFoundError while reading {file_path}: {e}")

def main():
    root_folder = 'root'
    output_folder = 'domain_output'
    os.makedirs(output_folder, exist_ok=True)

    input_files = get_input_files(root_folder)
    domains = set()
    for subdir, file_path in input_files:
        print(f"Processing file: {file_path}") # Debugging line
        for domain in read_domains_from_file(file_path):
            domains.add((domain, file_path))

    scrape_domains_multithreaded(domains, root_folder, output_folder=output_folder)

if __name__ == '__main__':
    main()
```

Appendix R

Shallalist – merge script

```
import os
import string

def process_text(text):
    text = text.replace("'", '')
    text = text.replace('"', ', ').replace(" or ", " ")
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text

def concatenate_files(directory, remove_duplicates=False):
    concatenated_text = set() if remove_duplicates else ""
    for filename in os.listdir(directory):
        if filename.endswith(".txt"):
            with open(os.path.join(directory, filename), 'r', encoding='utf-8') as file:
                text = file.read()
                text = process_text(text)
                if remove_duplicates:
                    concatenated_text.update(text.split())
                else:
                    concatenated_text += text + "\n"
    if remove_duplicates:
        return " ".join(concatenated_text)
    else:
        return concatenated_text

def save_merged_file(directory, category_name, text, processing):
    if not os.path.exists(directory):
        os.makedirs(directory)
    filename = f"{category_name}_{processing}.txt"
    filepath = os.path.join(directory, filename)
```

```
with open(filepath, 'w', encoding='utf-8') as file:
    file.write(text)

def main():
    categories_directory = 'shallalist'
    merged_directory = 'merged'

    for category_folder in os.listdir(categories_directory):
        category_path = os.path.join(categories_directory, category_folder)
        if os.path.isdir(category_path):
            subfolder_name = None
            for subfolder in os.listdir(category_path):
                subfolder_path = os.path.join(category_path, subfolder)
                if os.path.isdir(subfolder_path):
                    subfolder_name = subfolder
                    break

            if subfolder_name:
                output_folder = os.path.join(subfolder_path, 'output')
                category_output_directory = os.path.join(merged_directory, category_folder)

                merged_text_lang = concatenate_files(subfolder_path)
                save_merged_file(category_output_directory, category_folder, merged_text_lang, "

                merged_text_stopwords = concatenate_files(subfolder_path, remove_duplicates=True)
                save_merged_file(category_output_directory, category_folder, merged_text_stopwor

if __name__ == "__main__":
    main()
```

Appendix S

Shallalist cleaner Script - for TF-IDF and Cosign

```
import os
import chardet
import multiprocessing
import re
import nltk

from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')

WORD = re.compile(r'\w+')
STOPWORDS = set(stopwords.words('english'))

def text_to_vector(text):
    words = WORD.findall(text)
    return Counter(words)

def detect_encoding(file_path):
    with open(file_path, 'rb') as f:
        rawdata = f.read()
    encoding = chardet.detect(rawdata)['encoding']
    return encoding
```

```

def clean_text(text):
    text = text.lower()
    text = re.sub(r'[~a-zA-Z0-9\s]', '', text)
    text = re.sub(r'\s+', ' ', text)
    return text

def tokenize_text(text):
    tokens = word_tokenize(text)
    return tokens

def lemmatize_tokens(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return lemmatized_tokens

def extract_keywords(text):
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens if token not in STOPWORDS]
    cleaned_text = " ".join(lemmatized_tokens)
    return cleaned_text

def clean_file(input_file, output_directory):
    file_name = os.path.basename(input_file)
    output_filename_cleaned = os.path.join(output_directory, os.path.splitext(file_name)[0] + '_cleaned.txt')

    encoding = detect_encoding(input_file)
    with open(input_file, 'r', encoding=encoding, errors='ignore') as txtfile:
        content = txtfile.read()
        cleaned_text = clean_text(content)
        keywords = extract_keywords(cleaned_text)
        with open(output_filename_cleaned, 'w', encoding='utf-8') as txt_out:
            txt_out.write(keywords)

def clean_files_in_directory(input_directory, output_directory):
    files_to_clean = []
    for root, dirs, files in os.walk(input_directory):
        for file in files:
            if file.endswith('_lang.txt'):
                files_to_clean.append(os.path.join(root, file))

    num_processes = multiprocessing.cpu_count()
    with multiprocessing.Pool(processes=num_processes) as pool:
        pool.starmap(clean_file, [(file, output_directory) for file in files_to_clean])

if __name__ == "__main__":
    #The path of the folder to be cleaned

```

```
input_directory_path = "#####"
output_directory_path = './cleaned_tf_IDF'
clean_files_in_directory(input_directory_path, output_directory_path)
```

Appendix T

Shallalist cleaner Script – for Mini-LM

```
import os
import chardet
import multiprocessing
import re
import nltk

from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')

def text_to_vector(text):
    words = WORD.findall(text)
    return Counter(words)

def detect_encoding(file_path):
    with open(file_path, 'rb') as f:
        rawdata = f.read()
    encoding = chardet.detect(rawdata)['encoding']
    return encoding

def clean_text(text):
    text = text.lower()
    text = re.sub(r'^a-zA-Z0-9\s', '', text)
    text = re.sub(r'\s+', ' ', text)
    return text
```

```

def tokenize_text(text):
    tokens = word_tokenize(text)
    return tokens

def lemmatize_tokens(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return lemmatized_tokens

def extract_keywords(text):
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    cleaned_text = " ".join(lemmatized_tokens)
    return cleaned_text

def clean_file(input_file):
    file_name = os.path.basename(input_file)
    output_directory = './cleaned_minilm'
    os.makedirs(output_directory, exist_ok=True)
    output_filename_cleaned = os.path.join(output_directory, os.path.splitext(file_name)[0] + '_'

    encoding = detect_encoding(input_file)
    with open(input_file, 'r', encoding=encoding, errors='ignore') as txtfile:
        content = txtfile.read()
        cleaned_text = clean_text(content)
        keywords = extract_keywords(cleaned_text)
        with open(output_filename_cleaned, 'w', encoding='utf-8') as txt_out:
            txt_out.write(keywords)

def clean_files_in_directory(input_directory, output_directory):
    files_to_clean = []
    for root, dirs, files in os.walk(input_directory):
        for file in files:
            if file.endswith('_lang.txt'):
                files_to_clean.append(os.path.join(root, file))

    num_processes = multiprocessing.cpu_count()
    with multiprocessing.Pool(processes=num_processes) as pool:
        pool.starmap(clean_file, [(file, output_directory) for file in files_to_clean])

def main():
    #the path of the folder to be cleaned
    input_directory = "#####"

    output_directory = './cleaned_minilm'
    clean_files_in_directory(input_directory, output_directory)

```

```
if __name__ == "__main__":  
    main()
```

Appendix U

Elastic Connector

```
from elasticsearch import Elasticsearch

def elkConnector():
    connected = False
    try:
        elasticsearch_host = 'https://172.20.102.84:9200'
        username = 'elastic'
        ELASTIC_PASSWORD = 'n8u0WneEI180KQhkr0qm'
        CERT_FINGERPRINT = '0c3ee72217818266f6a2df36087afd989bcea4217fb1807d811bb8838ec1c355'
        client = Elasticsearch(
            elasticsearch_host,
            ssl_assert_fingerprint=CERT_FINGERPRINT,
            basic_auth=(username, ELASTIC_PASSWORD)
        )
        if client.ping():
            connected = True

    except Exception as e:
        client = None
        print(f"An error occurred: {e}")
    return client, connected

def elkdirconnect(client):
    if client:
        client.close()
```

Appendix V

Forward result script to Elasticsearch

```
import json
import logging
from concurrent.futures import ThreadPoolExecutor
from elasticsearch import Elasticsearch
from elkConnectionHandler import elkConnector, elkdisconnect

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

ELASTICSEARCH_INDEX = 'processed_results'
CHUNK_SIZE = 1000

def create_index_if_not_exists(es):
    if not es.indices.exists(index=ELASTICSEARCH_INDEX):
        mapping = {
            "mappings": {
                "properties": {
                    "_ID": {"type": "keyword"},
                    "timestamp": {"type": "keyword"},
                    "source_ip": {"type": "ip"},
                    "destination_ip": {"type": "ip"},
                    "source_port": {"type": "keyword"},
                    "destination_port": {"type": "keyword"},
                    "protocol": {"type": "keyword"},
                    "TLS_version": {"type": "keyword"},
                    "source_mac": {"type": "keyword"},
                    "destination_mac": {"type": "keyword"},
                    "src_country_iso_code": {"type": "keyword"},
```

```

        "src_country_name": {"type": "keyword"},
        "dst_country_iso_code": {"type": "keyword"},
        "dst_country_name": {"type": "keyword"},
        "dns_resolution": {"type": "keyword"}
    }
}

es.indices.create(index=ELASTICSEARCH_INDEX, body=mapping)
logging.info(f"Created index: {ELASTICSEARCH_INDEX} with mapping")

def forward_to_elasticsearch(processed_results):
    try:
        client, connected = elkConnector()
        if connected:
            create_index_if_not_exists(client)
            logging.info("Forwarding data to Elasticsearch...")
            with ThreadPoolExecutor() as executor:
                results = executor.map(send_to_elasticsearch, [client]*len(processed_results), p
                for result in results:
                    pass
            logging.info("Data forwarding completed.")
        else:
            logging.error("Failed to connect to Elasticsearch.")
    finally:
        elasticsearch.disconnect(client)

def send_to_elasticsearch(client, result):
    try:
        doc = {
            "_ID": result["_id"]["$oid"],
            "timestamp": result["timestamp"]["$date"],
            "source_ip": result["source_ip"],
            "destination_ip": result["destination_ip"],
            "source_port": result["source_port"],
            "destination_port": result["destination_port"],
            "protocol": result["protocol"],
            "TLS_version": result.get("TLS_version", None),
            "source_mac": result["source_mac"],
            "destination_mac": result["destination_mac"],
            "src_country_iso_code": result["src_country_iso_code"],
            "src_country_name": result["src_country_name"],
            "dst_country_iso_code": result["dst_country_iso_code"],
            "dst_country_name": result["dst_country_name"],
            "dns_resolution": result["dns_resolution"]
        }

        for section, data in result["processed_results"].items():
            for key, value in data.items():

```

```
        for i, item in enumerate(value[:3], start=1):
            doc[f"{section}_{i}_label"] = item.get("label")
            doc[f"{section}_{i}_score"] = item.get("similarity")

        client.index(index=ELASTICSEARCH_INDEX, body=doc)
    except Exception as e:
        logging.error(f"Failed to send data to Elasticsearch: {e}")

def main():
    try:
        with open('processed_result.json', 'r') as combined_file:
            processed_results = json.load(combined_file)

        forward_to_elasticsearch(processed_results)
    except FileNotFoundError:
        logging.error("Combined_Success.json file not found.")
    except Exception as e:
        logging.error(f"An unexpected error occurred: {e}")

if __name__ == "__main__":
    main()
```

Appendix W

Configuration and installation

1 Wireshark installation guide

TABLE W.1: Wireshark installation guide

1. Update the package repository to ensure the latest software versions are available:
sudo apt update
2. Install Wireshark along with its command-line counterpart, TShark, by running the following command: **sudo apt install wireshark**
3. During the installation process, a dialog box prompts to choose whether non-superusers should be able to use Wireshark. Selecting "Yes" granted this permission.
4. To allow non-superusers to capture packets, add the user to the "wireshark" group:
sudo usermod -aG wireshark username
5. Log out and log back in for the changes to take effect.
6. Launch Wireshark with superuser privileges using the following command: **sudo wireshark**
7. Wireshark's graphical user interface (GUI) opened, allowing network traffic analysis and packet capturing on the Ubuntu virtual machine.

Following these steps, Wireshark was successfully installed on the Ubuntu virtual machine, enabling the capture and analysis of network traffic.

2 Configuring Tor Exit Node

Setting up a Tor exit node requires careful consideration and adherence to several steps to ensure proper and secure configuration. Below is a general outline of the steps involved in configuring a Tor exit node:

1. **Install Tor:** Begin by installing the Tor software on your server. You can do this using the package manager of your operating system or by downloading the Tor software directly from the official Tor Project website.

2. **Configure Tor:** Edit the Tor configuration file (`torrc`) to enable the exit node functionality. You must set the `ExitRelay` option to 1 in the configuration file. You can also configure other settings related to your Tor exit relay, such as bandwidth limits and exit policies.
3. **Port Forwarding:** Make sure that the necessary ports for Tor (such as port 9001 for TCP traffic and port 9030 for directory requests) are open on your server. You may need to configure port forwarding on your router or firewall to ensure Tor traffic can pass through.
4. **Static IP Address:** It is recommended to have a static IP address for your server hosting the Tor exit node. This ensures the node's IP address remains consistent, making it easier for users to connect to your exit node reliably.
5. **Exit Policy:** Configure your exit policy to specify which types of traffic are allowed to exit through your node. You can specify which IP addresses and ports are allowed or blocked based on your preferences and security considerations.
6. **Security Considerations:** Ensure that your server hosting the Tor exit node is secure and updated with security patches. Regularly monitor your server for any unusual activity or potential security vulnerabilities.

TABLE W.2: Tor exit node configuration guide

1. **Tor Package installation:** Ensure you update the packages database before installing the package, then call apt to install it:
 - **apt update**
 - **apt install tor**
2. **Configuration file:** change the following parts of the configuration file `/etc/tor/torrc`
 - **Nickname EastAfricanRelay** : Change the "myNiceRelay" part
 - **ContactInfo eastafricarelay@proton.me:** Wrote the e-mail address
 - **ORPort 9001** :Change the port number
 - **ExitRelay 1** : To become an exit relay change ExitRelay from 0 to 1 in the torrc configuration file.
 - **SocksPort 0**
 - **RelayBandwidthRate 100 Mbytes** : Define these to limit how much relayed traffic is allowed.
3. **Exit Notice HTML page:** To make it even more obvious that this is a Tor exit relay you should serve a Tor exit notice HTML page. Tor can do that for you: if your **DirPort** is on TCP port 80, you can make use of tor's :
 - **DirPort 80**

DirPortFrontPage : feature to display an HTML file on that port. This file will be shown to anyone directing their browser to your Tor exit relay IP address. If you didn't set this up before, the following configuration lines must be applied to your torrc:

 - **DirPortFrontPage /etc/tor/tor-exit-notice.html**
4. **Restart the service:** Restart the tor daemon, so your configuration changes take effect:
 - **systemctl restart tor@default**

3 Elasticsearch installation guide

Moreover, the installation of Elasticsearch and Kibana is shown in Table [W.3](#) following steps

TABLE W.3: Elasticsearch configuration

Elasticsearch configuration**Download and install archive for Linux**

- **wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.12.2-linux-x86_64.tar.gz**
- **wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.12.2-linux-x86_64.tar.gz.sha512**
- **shasum -a 512 -c elasticsearch-8.12.2-linux-x86_64.tar.gz.sha512**
- **tar -xzf elasticsearch-8.12.2-linux-x86_64.tar.gz**
- **cd elasticsearch-8.12.2/**

Run Elasticsearch from the command line

- To run elastic foreground within a terminal: **./bin/elasticsearch**
- To run elastic as a daemon use: **./bin/elasticsearch -d -p pid to exit pkill -F pid**

When starting Elasticsearch for the first time, security features are enabled and configured by default. The following security configuration occurs automatically:

- Authentication and authorization are enabled, and a password is generated for the elastic built-in superuser.
- Certificates and keys for TLS are generated for the transport and HTTP layer, and TLS is enabled and configured with these keys and certificates.
- An enrollment token is generated for Kibana, which is valid for 30 minutes. To regenerate if it expires use **bin/elasticsearch-create-enrollment-token -s node** for enrollment token for the node and **bin/elasticsearch-create-enrollment-token -s kibana** for kibana token

The password for the elastic user and the enrollment token for Kibana are output to your terminal Elasticsearch automatically generates certificates and keys in the following directory:

config/certs

You can test that your Elasticsearch node is running by sending an HTTPS request to port 9200 on localhost: **curl -cacert /config/certs/http_ca.crt -u elastic:\$ELASTIC_PASSWORD https://localhost:9200**

4 Kibana installation guide

TABLE W.4: Kibana Configuration

Kibana Configuration

Download and install the Linux 64-bit package

- The Linux archive for Kibana v8.12.2 can be downloaded and installed as follows: `curl -O https://artifacts.elastic.co/downloads/kibana/kibana-8.12.2-linux-x86_64.tar.gz`
- `curl https://artifacts.elastic.co/downloads/kibana/kibana-8.12.2-linux-x86_64.tar.gz.sha512 --shasum -a 512 -c -tar -xzf kibana-8.12.2-linux-x86_64.tar.gzcd kibana-8.12.2/`
- Start Elasticsearch if it is not up and generate an enrollment token for Kibana

Run Kibana from the command line

- Run Kibana from the command line: `./bin/kibana`

By default, Kibana runs in the foreground, prints its logs to the standard output (stdout), and can be stopped by pressing **Ctrl-C**.

When starting Kibana for the first time, this command generates a unique link in the terminal to enroll the Kibana instance with Elasticsearch.

- In your terminal, click the generated link to open Kibana in your browser.
- In your browser, paste the enrollment token generated in the terminal and then click the button to connect your Kibana instance with Elasticsearch.
- Login to Kibana as the Elastic user with the password generated when you started Elasticsearch.

Appendix X

Shallalist web categories

TABLE X.1: Shallalist web categories

No.	Category name	Category Descriptions
1.	Advertisements	All about advertising: This includes sites offering banners and banner creation as well as sites delivering banners to be shown on webpages and advertising companies.
2.	Aggressive	Sites with aggressive content, such as racism and hate speech.
3.	Alcohol	Sites of breweries, wineries, and distilleries. This category also covers sites that explain how to make beer, wines, and spirits.
4.	Anonymous VPN	Sites providing VPN services to the public. The focus is on VPN sites used to hide the origin of the traffic, f.e. tor nodes.

5.	Automobile	<ul style="list-style-type: none">• All sites related to cars. Included are automobile companies and automotive suppliers.• All sites related to motorcycles. Included are vendor sites, resellers, fan and hobby pages, and suppliers. Scooters included.• All sites related motorboats. Included are vendor sites, resellers, fan and hobby pages, and suppliers.• All sites related to planes ranging from small one and two seaters up to the large traffic planes, old and new, private, commercial, and military. Vendors and suppliers are included (airports are not). Helicopter sites are included as well.
6.	Chat	Site for real-time chatting and instant messaging.
7.	Cost Traps	Sites that lure you with free-of-charge services but then give you a costly subscription (written somewhere in tiny letters nearly unreadable).
8.	Dating	Sites to contact people for love and living together. He seeks her, she seeks him, and so on.
9.	Downloads	This covers mostly filesharing, p2p, and torrent sites. Other download sites (for software, wallpapers, etc.) are included as well.
10.	Drugs	Sites offering drugs or explain how to make drugs. Covers alcohol and tobacco as well as viagra and similar substances.
11.	Dynamic	All domains where people login obtaining a dynamic IP address.

12.	Education	Home pages of schools, colleges and universities.
13.	Finance	<ul style="list-style-type: none"> • Home page of banking companies are listed here. This is not restricted to online banking. • Sites of insurance companies, information about insurances and link collections concerning this subject. • Sites one can apply for loans and mortgages or can obtain information about this business. • Sites about all types of real estate, buying and selling homes, finding apartments for rent. • Sites about and related to the stock exchange and Finance in general.
14.	Fortunetelling	All sites about astrology, horoscopes, numerology, palm reading, and so on; sites that offer services to foretell the future.
15.	Forum	Discussion sites. Covers explicit forum sites and some blogs. Sites where people can discuss and share information in a non interactive/real-time way.
16.	Government	Sites belonging to the government of a country, county, or city.
17.	Gambling	Sites offering the possibility to win money. Poker, Casino, Bingo, and other chance games, as well as betting sites.
18.	Hacking	Sites with information and discussions about security weaknesses and how to exploit them. Sites offering exploits are listed as well as sites distributing programs that help to find security leaks.

19.	Hobby	<ul style="list-style-type: none"> • Sites concerning food and food preparation. • Sites related to games. This includes descriptions, news, and general information about games. No gambling sites. • Sites about online games. The games are for fun only (no gambling). • Sites about gardening, growing plants, fighting bugs, and everything related to gardening. • All topics concerning pets: description, breed, food, looks, fairs, favorite pet stories, and so on.
20.	Homestyle	Sites about everything need to create a cozy home (interior design and accessories).
21.	Hospitals	Sites of hospitals and medical facilities.
22.	Image hosting	Sites specialized in hosting images, photo galleries, and so on.
23.	Internet Service Provider	Home pages of Internet Service Providers and Sites of companies offering webspace only are now being added, too
24.	Jobsearch	Portals for job offers and job seekers, as well as the career and work-for-us pages of companies.
25.	Library	Online libraries and sites where you can read e-books.
26.	Military	Sites of military facilities or related to the armed forces.
27.	Models	Model agency, model and supermodel fan pages, and other model sites presenting model photos. No porn pictures.

28.	Movies	Sites offering cinema programs and information about movies and actors. Sites for downloading video clips/movies (as long as it is legal) are also included.
29.	Music	Sites that offer the download of music, information about music groups, or music in general.
30.	News	Sites presenting news. Homepages from newspapers, magazines, journals, and some blogs.
31.	Podcasts	Sites offering podcasts or podcast services include audio-books.
32.	Politics	Sites of political parties, political organizations, and associations; sites with political discussions.
33.	Porn	Sites about all kinds of sexual content ranging from bare bosoms to hardcore porn and sm.
34.	Radio and TV	Domains and URLs of TV and radio stations.

35.	Recreation	<ul style="list-style-type: none">• Humorous pages, comic strips, funny stories, everything that makes people laugh.• Sites dedicated to martial arts such as karate, kung fu, taek won do as well as fighting sports sites like ufc.• Sites of restaurants as well as restaurant descriptions and commentaries.• All about sports: sports teams, sports discussions, as well as information about sports people and the various sports themselves.• Sites with information about foreign countries, travel companies, travel fares, accommodations, and everything else that has to do with travel.• Sites about treatments for feeling internally and externally healthy and beautiful again.
36.	Redirector	Sites that actively help to bypass URL filters by accepting URLs via the form and play a proxying and redirecting role.
37.	Religion	Sites with religious content: all kinds of churches, sects, religious interpretations, etc.
38.	Remote control	Sites offering the service to remotely access computers, especially (but not limited to going) through firewalls. This does not cover traditional VPNs.
39.	Ringtones	Sites that offer the download of ringtones or present other information about ringtones.

40.	Science	<ul style="list-style-type: none"> • Sites of institutions as well as of amateurs about all topics of astronomy. • Sites of institutions as well as of amateurs about all topics of chemistry.
41.	Search engines	Collection of search engines and directory sites.
42.	Sex	<ul style="list-style-type: none"> • Sites selling and presenting sexy lingerie. • Sites explaining the biological functions of the body concerning sexuality as well as sexual health.
43.	Shopping	Sites offering online shopping and price comparisons.
44.	Social Networking	Sites bringing people together (social networking) be it for friendship or for business.
45.	Spyware	Sites that try to actively install software or lure the user into doing so in order to spy the surfing behavior (or worse). The home calling sites where the collected information is sent, are listed too.
46.	Tracker	Sites that keep an eye on where you surf and what you do in a passive manner. It covers web bugs, counters, and other tracking mechanisms in web pages that do not interfere with the local computer and collects information about the surfing person for later analysis.
47.	Update Sites	List to allow necessary downloads from vendors.
48.	URL shortener	Sites offering short links for URLs.
49.	Violence	Sites about killing and harming people. Covers anything about brutality and bestiality.

50.	Warez	Collection of sites offering programs to break license keys, license keys themselves, cracked software, and other copyrighted material.
51.	Weapons	Sites offering all kinds of weapons or accessories for weapons: Firearms, knives, swords, bows, etc. Armory shops are included, as well as sites that hold general information about arms (manufacturing and usage).
52.	Webmail	Sites that offer web-based email services.
53.	Webphone	Sites that enable users to phone via the Internet. Any site where users can voice-chat with each other.
54.	Webradio	Sites that offer to listen to music and live radio streams.
55.	Web TV	Sites offering TV streams via the Internet.

Appendix Y

Random sample distribution

TABLE Y.1: Sample data distribution

No.	Captured date	Number of Pcap file (Sampling distribution)
1.	17/11/2023	20
2.	18/11/2023	75
3.	19/11/2023	80
4.	20/11/2023	25
5.	21/11/2023	20
6.	22/11/2023	23
7.	23/11/2023	27
8.	24/11/2023	25
9.	25/11/2023	33
10.	26/11/2023	22

11.	27/11/2023	25
12.	28/11/2023	38
13.	29/11/2023	40
14.	30/11/2023	2
15.	05/12/2023	24
16.	06/12/2023	33
17.	07/12/2023	30
18.	08/12/2023	32
19.	09/12/2023	42
20.	10/12/2023	37
21.	11/12/2023	23
22.	14/12/2023	15
23.	15/12/2023	38
24.	16/12/2023	53
25.	17/12/2023	34
26.	18/12/2023	43
27.	19/12/2023	37
28.	22/12/2023	12
29.	23/12/2023	68
30.	24/12/2023	42
31.	25/12/2023	34

32.	26/12/2023	11
	Total	1063

Assurance

This thesis is submitted as a partial fulfillment of the Master of Science in Cyber Security (School of Information Technology and Engineering), Addis Ababa Institute of Technology, Addis Ababa University. The author carried out the work included in this thesis. Wherever other people's contributions are involved, every effort is made to make this clear, with an appropriate citation from the source.

Tariku Eshetu

Signature

.....

Date: 16 August, 2024