People's Democratic Republic of Algeria
Abou-Bakr Belkaid University - Tlemcen
Faculty of Sciences
Computer Science department

Master Dissertation

Submitted in partial fulfillment of the requirements for the Master degree in Computer Science
Track: Network and distributed system

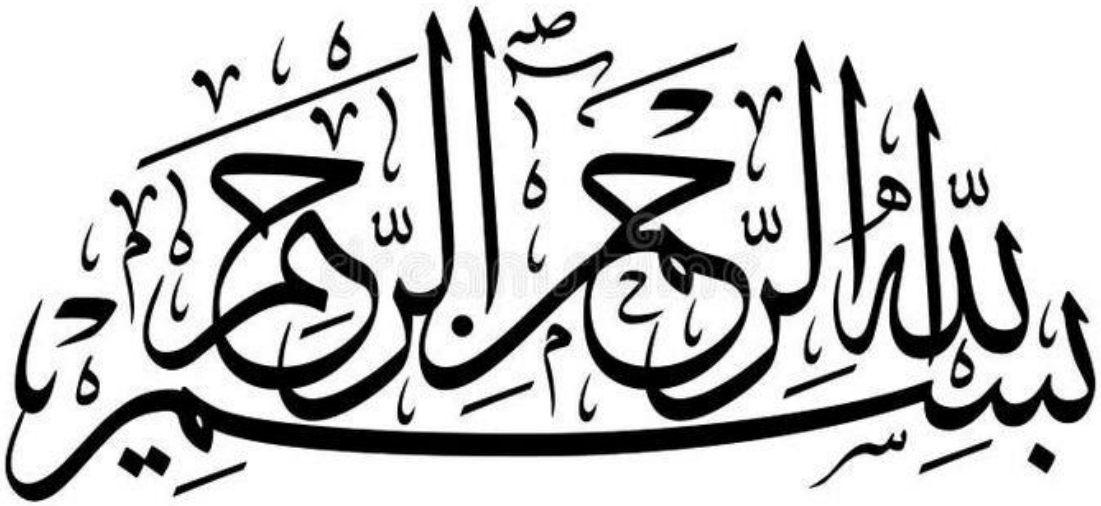
Presented by: BensenouciAbir

Intrusion Detection System (IDS)

Presented before the following committee:

Chair	Benamar Abdelkrim	MCA
Examiner	Benmouna Youcef	MCB
Supervisor	Ilyas Bambrik	MCB

Academic Year: 2024-2025


بسم الله الرحمن الرحيم

Abstract

As cyberattacks become stealthier and more sophisticated, detecting them becomes a major challenge for security systems. This project examines major cyber threats such as phishing, DDoS, and DNS tunneling, and then looks at the strengths and weaknesses of different types of IDS. The focus is on DNS tunneling - a covert method of data exfiltration. To address this problem, we developed a custom IDS in Python using Scapy. It monitors DNS traffic in real time, detects anomalies using whitelists, and notifies administrators of suspicious activity. The system was tested in a virtual lab using the Iodine tool and was proven to be effective in identifying DNS tunneling with a very low false positive rate. This work highlights the value of custom IDS solutions in combating modern cybersecurity threats.

ملخص

مع تزايد تعقيد الهجمات الإلكترونية وسريتها، أصبح اكتشافها تحديًا كبيرًا لأنظمة الأمن الحديثة. يتناول البرنامج أبرز التهديدات التي تواجه الشبكات، مثل التصيد الاحتيالي، وهجمات الحرمان من الخدمة الموزعة، وأنفاق نظام أسماء النطاقات(DNS) ، ثم يستعرض إيجابيات وسلبيات أنواع مختلفة من أنظمة اكتشاف التطفل .(IDS) يركز المشروع بشكل خاص على أنفاقDNS ، وهي طريقة سر

ية تستخدم لتهريب البيانات. لمحاربة هذا النوع من الهجوم، تم تطوير نظام IDS مخصص باستخدام Python ومكتبة .Scapyيقوم النظام بمراقبة حركة مرور DNS في الوقت الفعلي، ويستخدم القوائم البيضاء للكشف عن الأنشطة المشبوهة، وينبه المسؤولين عند اكتشاف تهديدات محتملة. تثبت التجارب التي أجريت باستخدام اليود في بيئة افتراضية قدرة النظام على اكتشاف هجمات أنفاق DNS بشكل فعال مع تقليل النتائج الإيجابية الخاطئة. يسلط هذا العمل الضوء على أهمية تطوير حلول IDS مخصصة لمواجهة التهديدات السيبرانية المعاصرة.

Résumé

Les cyberattaques devenant de plus en plus furtives et sophistiquées, leur détection constitue un véritable défi pour les systèmes de sécurité. Le projet décrit les principales cybermenaces telles que le phishing, les attaques DDoS et le tunneling DNS, puis examine les points forts et les limites des différents types de systèmes de détection d’intrusion (IDS). L’accent est mis sur le tunneling DNS – une méthode furtive d’exfiltration de données. Pour résoudre ce problème, nous avons développé un IDS personnalisé en Python en utilisant la bibliothèque Scapy. Il surveille le trafic DNS en temps réel, utilise des listes blanches pour détecter les anomalies et alerte les administrateurs en cas d'activité suspecte. En le testant avec l’outil Iodine dans un environnement virtuel, le système a démontré son efficacité à identifier les attaques DNS tout en limitant les faux positifs. Cet ouvrage souligne l’importance des solutions IDS personnalisées face aux menaces modernes de cybersécurité.

Dedication

I dedicate this work first and foremost to God, whose kindness, guidance and strength have sustained me every step of this journey. None of this would have been possible without His blessings.

I thank my dear parents for everything they have given me. Your unconditional love, endless sacrifices and constant encouragement have made me who I am today. I attribute this success to your leadership and the values you have taught me. This success is a reflection of your dedication and I dedicate it to you with all my heart.

I want to thank my dear sister Azza, my two brothers and my best friends Amel and abderahmen, who have always been there for me. Your presence, your support and your kind words have lifted me up during difficult moments and made this experience more meaningful.

I deeply appreciate the love, encouragement and comfort you have given me during this journey.

Gratitude

First, I would like to express my deepest gratitude to God for giving me the strength, perseverance, and guidance needed to execute and complete this project.

I would like to express my sincerest gratitude to my supervisor, **Mr. Bambrik Ilyas**, for his invaluable support, insightful guidance, and constructive feedback throughout the work. His encouragement and expertise were essential to the successful completion of this project.

I am also very grateful to the members of the jury: **Mr. Benamar Abdelkrim** and **Mr. Benmoana Youcef** for giving me the honor of judging my work and for their time and attention.

Finally, I would like to thank everyone who has supported, encouraged, or inspired me in one way or another during this journey.

My sincere gratitude – thank you very much.

Table of Contents

Table of figures:

Introduction

The frequency of cyberattacks has surged in recent years, presenting significant challenges for defenders, especially with the rise of advanced persistent threats (APTs). Attackers continuously adapt their strategies, exploring new vulnerabilities and employing obfuscation techniques to evade detection. In response, defenders need to be agile in identifying and mitigating malicious actions promptly. Many enterprises utilize Intrusion Detection Systems (IDS) to flag potential security threats, with security professionals tasked with evaluating the alerts generated. However, the shortcomings of existing IDS often result in a barrage of false positives, overwhelming security teams and, at times, allowing intrusions to remain unnoticed for extended periods. Cybersecurity research has aimed at enhancing detection rates and minimizing false alerts to mitigate the repercussions of APTs, safeguarding enterprises from potential financial losses and data breaches. Recent investigations into data provenance methods show promise, as they have been found to significantly improve detection capabilities while lowering the frequency of false alarms, thereby easing the cognitive burden on security professionals. Data provenance tracks system activities as a Directed Acyclic Graph (DAG), detailing the flow of information between entities like processes and objects such as files and network sockets. Nonetheless, despite advancements in provenance-based intrusion detection systems (PIDS), several challenges persist:

a) Data provenance in large enterprises can grow at an alarming rate, exceeding multiple terabytes, current detection methods depend on manually adjusted thresholds for specific situations.
b) There is a restricted exploration of contextual information that could enhance detection efficacy.
c) The existing benchmark datasets are either outdated or fail to capture genuine benign system activities.

Chapter 1: Network Threats

1. Introduction

In order to provide the security measures in computer network, the attacks should be clearly defined. In this chapter we define the types of cyberattacks and how they evolved with advances in computing, networks and user defined applications.

2. History of Cyberthreats

An attack is a dangerous or non-dangerous attempt to modify or use a resource accessible through the network in a way that was not intended. The network attacks can be classified into three general categories:

1- Unauthorized access to resources and information through the network.
2- Unauthorized manipulation of information on a network.
3- Attacks that lead to disruption of service delivery and are called Denial of Service [1].

Cyber-crimes have adapted to technological progress along with criminal methodology development during the previous decades. Computer crime did not exist during the 1940s because digital technology at the time had not advanced sufficiently. “Phone phreaking” emerged during the 1950s as a hacking activity that previewed later hacking methods which targeted telephone systems. The terms “hacking” and “vulnerabilities” started appearing in the 1960s indicating growing recognition of digital security threats during that period. Institutions began to establish computer security measures during the 1970s as they understood the necessity of system protection against emerging threats. Furthermore, the Internet's development through “ARPANET” expansion in the 1980s created a network of interconnected systems which provided criminals new ways to conduct attacks.

During the 1990s, computer viruses and worms became major tools for destructive attacks and data theft. The Internet experienced excessive growth throughout the 2000s which simultaneously created new opportunities alongside increased vulnerabilities. Cyber criminals perfected their methods during the 2010s by executing sophisticated attacks to exploit security breaches. Since then, cybercrime has developed into a complete modern industry, using organized networks and advanced technologies that threaten global digital infrastructure. The timeline illustrates how cybercrimes evolved from isolated

incidents into a continuous threat requiring uninterrupted cybersecurity innovation [2].

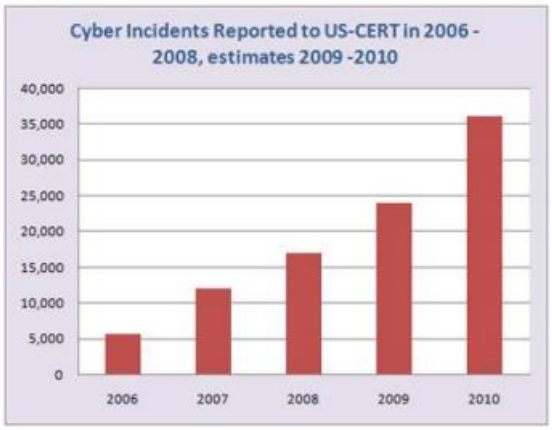


Figure 1: Number of cyber incidents reported

One example that illustrates the difficulty of detecting such sophisticated threats is the recent SolarWinds attack in 2020. The nation-state attackers got access to SolarWinds's software build system, which is meant to be used to manage enterprise networks and facilitate reporting. Afterwards, the attacks injected malicious code to a significant software update, and the update rollout distributed the malicious code to over 18,000 SolarWinds's customers' systems. Records state that over one thousand hackers must have been involved in the attack, and approximately 40 companies, including Fortune 500 companies and multiple US government agencies, have been compromised. The financial impact of the attack was extreme as "BITSIGHT" estimates the cost of incident response and forensic analysis at $90,000,000 in addition to the financial damage from data exfiltration [3].

3. Types of Network Attacks

3.1 Phishing

Phishing is a practice where cybercriminals send emails and other messages to unsuspecting people purporting to be representing reputable companies with the goal of tricking them into revealing passwords and other sensitive information.

For example, you receive an email that looks like it's from your bank, asking you to verify your account details. The email will even have the bank's logo and a perfect copy of the bank's email layout. But instead of your bank, it's a hacker trying to steal your information [4].

In order to reduce the risk of phishing attacks, critical thinking, hovering over the links, analyzing email headers and sandboxing must be used. Moreover, by raising awareness among the organization employees as well as for individuals, it is possible to prevent this type of attack to some extent [5].

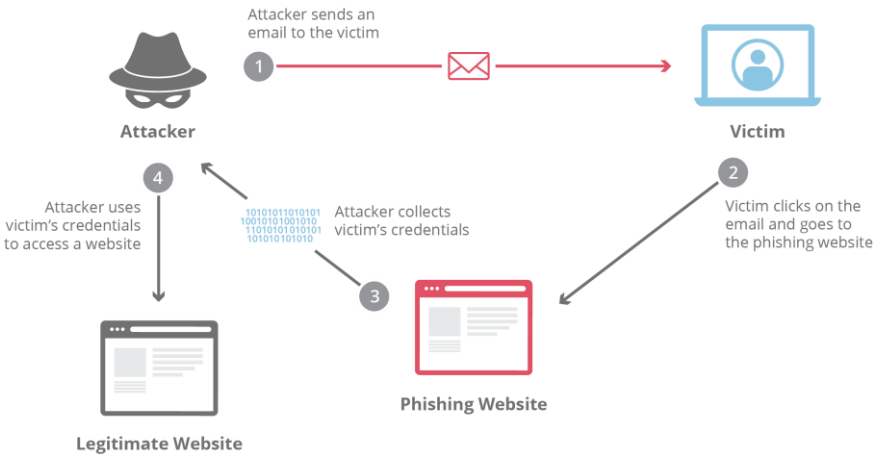


Figure 2: Phishing attack

3.2 DDoS (Distributed Denial of Service)

DDoS, which stands for Distributed Denial of Service, is one of the most concerning types of network attacks that network administrators must confront. These attacks are designed to cause disruption by overwhelming systems with a flood of requests. This influx of traffic consumes system or application resources, making it impossible for legitimate users to access the services.

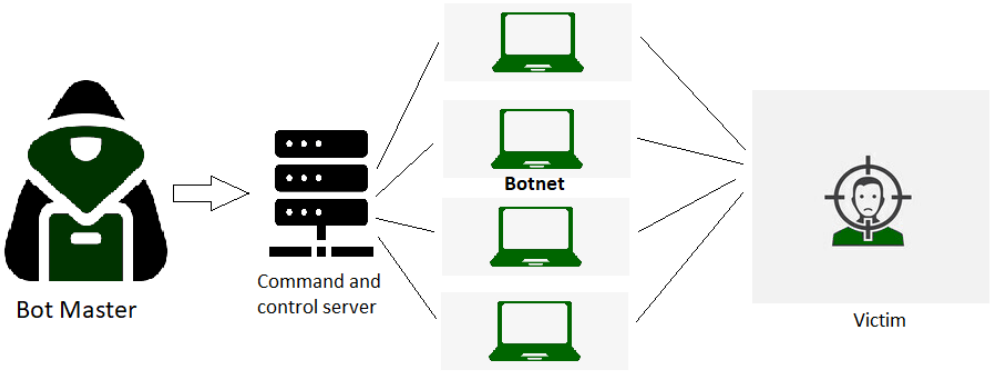


Figure 3: DDoS attack

A few years ago, a massive DDoS attack targeted Dyn, a company that manages domain name systems (DNS) for websites like Twitter, Netflix, and Reddit. The attackers used a botnet, which is a network of infected devices, to send an overwhelming amount of traffic to Dyn's servers. The result was that websites

that millions of people visited daily were suddenly inaccessible. It was like someone had pulled the plug on half the internet [4].

DDoS are usually conducted through a botnet, which is controlled by a malicious entity called the botmaster. A botmaster remotely controls the zombies and instructs them to perform malicious activities through commands. The way the bots are controlled depends on the architecture of botnet command and control mechanisms, which may be IRC, HTTP, DNS or P2P-based. These botnets are used to commit cyber-crimes such as sending spam mail, launching denial-of-service attacks or stealing personal data such as mail accounts or bank credentials. It is common knowledge that approximately 80% of all email traffic is spam and most such messages are sent through botnets [6].

3.3 Man-in-the-Middle

Man-in-the-Middle (MIM) attacks happen when malicious actors intercept and disrupt private communications between two unsuspecting parties. These cybercriminals, often referred to as “black hats”, place themselves between the victims, allowing them to monitor and control the information exchanged. As a result, they can disrupt files, intercept sensitive data, and engage in espionage [7].

One common tactic used in MITM attacks is to create fake Wi-Fi hotspots where you walk into a coffee shop and immediately get prompted to connect to "Free Café WIFI". The WIFI network will look legit but will actually be a rogue network set up by a hacker. Once connected, everything you do online can be monitored or manipulated.

Another tactic involves what’s called SSL stripping. Websites today often use HTTPS to encrypt data. However, an attacker can downgrade this connection to HTTP, which is not secure. Once they do that, they can then intercept everything sent to the website, such as login credentials or personal information. For instance, you might think you're securely logging into your bank account, but in reality, the attacker is capturing your username and password in plain text [4].

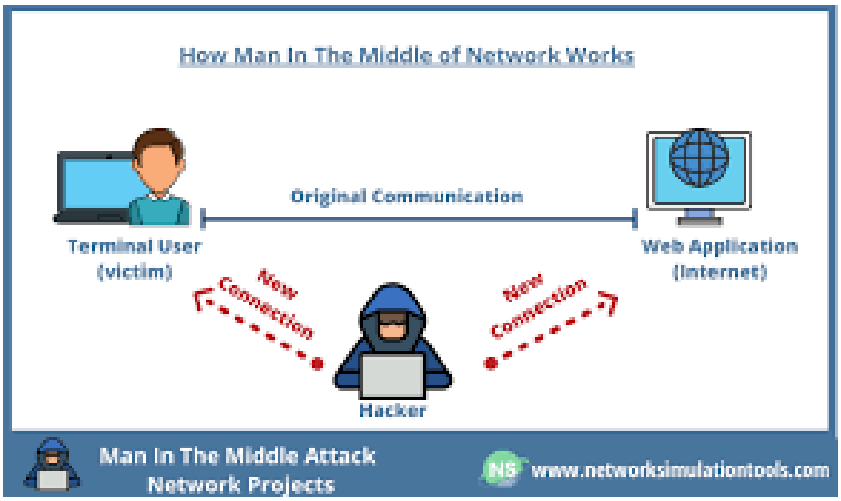


Figure 4: Man in The Middle attack

4. The Art of Deception: Obfuscation and the Hidden Layers of Cybersecurity

Obfuscation techniques are employed to evade detection by disguising malicious activities, making the attack more difficult to interpret. The concept of obfuscation involves modifying program code while preserving its original functionality, by making the code obscure and less readable, malware can effectively bypass modern IDSs.

Obfuscation exploits weaknesses in signature-based detection by leveraging limitations in the signature database and the way systems analyze data. However, code obfuscation remains a powerful tool for cybercriminals, enabling them to bypass IDS detection and execute attacks undetected.

To illustrate the impact of obfuscation techniques on IDS evasion, we will examine three specific attacks methods: Cross-site Scripting XXS, Tunneling, and Headless Browser Attacks. Each of these exploits leverage obfuscation to bypass traditional detection mechanisms, highlighting the challenges faced by modern security systems.

4.1 Cross-site Scripting (XSS)

Cross-Site Scripting (XSS) is a type of JavaScript code injection attack that allows an attacker to execute malicious JavaScript in a victim's web browser. This exploitation can lead to unauthorized access to sensitive information such as cookies, passwords, and credit card numbers. While XSS attacks target the client-side web browser, their effects can be felt on the web server side as well.

In order to exploit XSS vulnerabilities in web applications, an attacker carefully crafts and injects a malicious JavaScript payload into the application. This script is designed to appear as a benign component of the website, ultimately executing within the trusted domain of the site.

Figure 5: Scenario for a reflected or indirect XSS attack

A common scenario for a reflected Cross-Site Scripting (XSS) attack is illustrated in Figure5. To carry out this attack, all that is needed is a vulnerable web page and the ability to input malicious code through its search engine. For instance, if an attacker injects a malicious script into the application, as depicted in Figure 6, the script can trigger an alert in the user's web browser. In this example, the script is executed after accessing the domain pizza.com, making it particularly easy to exploit, especially when the targeted website is vulnerable to XSS attacks. This type of attack relies on the web application reflecting the injected script back to the user's browser without proper validation or sanitization.

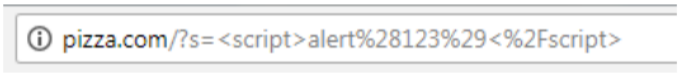


Figure 6: Sample XSS code reflected injected by URL

The Google Chrome browser detects and stop the script execution of the malicious scripts, and built-in security mechanisms detects as showed in Figure 7 [8].

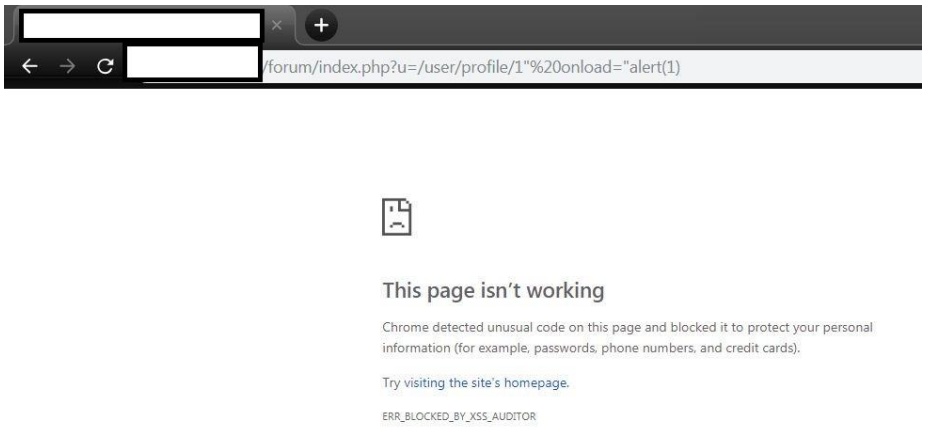


Figure 7: Chrome browser response to XSS attack

4.2 Protocol Tunneling

Tunneling attacks represent a typology of network attacks where the attacker initiates a covert communication tunnel by embedded traffic within another protocol. It is involved in virtual network, which allows the attacker to evade traditional network security measures (firewalls, ids...), and potentially extract sensitive data [9].

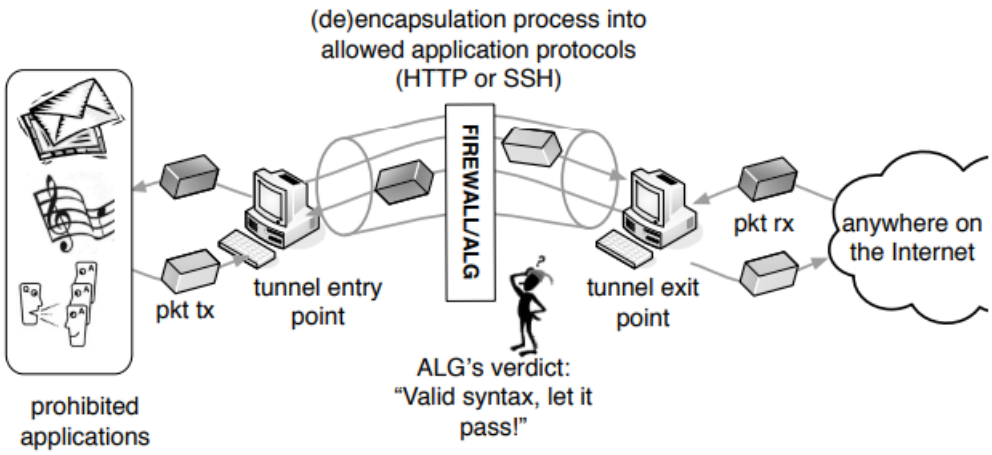


Figure 8: How protocol tunneling work

One of the most widely used techniques in this category is DNS tunneling: Which is a security threat that involves the covert transmission of data and commands by exploiting DNS. As one of the most prominent DNS-based attacks, DNS pose a serious risk to network security. Its ability to evade detection make it a top concern for cybersecurity professionals.

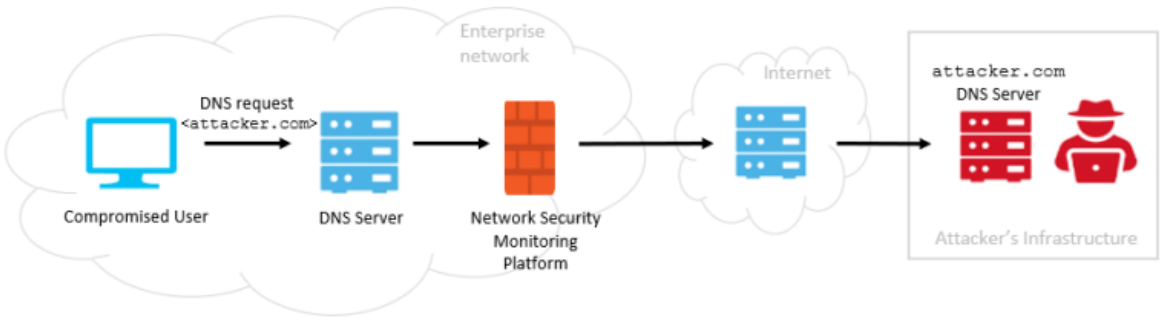


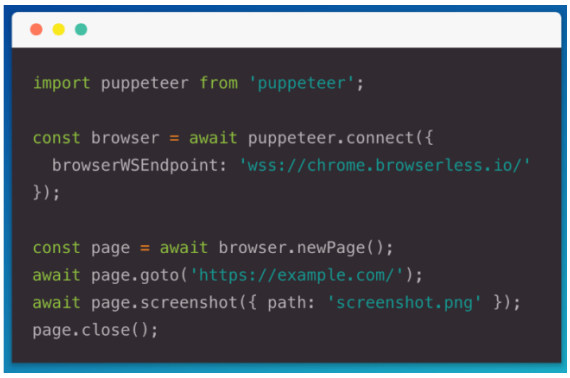
Figure 9:DNS tunneling

Figure 9 illustrates a typical DNS tunneling scenario [9]:

1. First, the attacker registers a malicious domain (e.g.: attacker.com, accessed on 02/03/2025) hosted on a command-and-control (C&C) server under their control.
2. Once the attacker gains control of a machine within the targeted network and breaches its security perimeter, the compromised device sends a DNS query to the malicious domain.
3. Since DNS requests are generally permitted to traverse network boundaries, the query passes through the DNS resolver and reaches the attacker's C&C server, where the tunneling program is hosted.

4.3. Headless Browser

Headless browsers are web browsers that run without a graphical user interface (GUI). This means that it runs entirely in the background, without displaying any visual elements like windows, tabs, or toolbars. Instead, they can perform tasks like: loading websites, clicking buttons, filling out forms, logging into accounts, make HTTP requests, render content, visit URLs,**capture a Screenshot (Figure 10),** etc., which is difficult or impossible with a traditional graphic browser.



```
import puppeteer from 'puppeteer';

const browser = await puppeteer.connect({
  browserWSEndpoint: 'wss://chrome.browserless.io/'
});

const page = await browser.newPage();
await page.goto('https://example.com/');
await page.screenshot({ path: 'screenshot.png' });
page.close();
```

Figure 10: Example of using Puppeteer to capture a screenshot with a Headless Browser

While headless browsers are a powerful tool for developers and security professionals, they also pose a serious risk to the security and vulnerabilities of the web applications, including ad fraud owing to it's capability of simulating human-like interactions including mouse movements and clicks.

For example, a headless browser could automate a brute force attack on a login form, rapidly testing thousands of password combinations.

Additionally, they can be used to execute more advanced attacks, such as exploiting cross-site scripting (XSS) vulnerabilities or other security weaknesses in web applications.

For instance, if a headless browser is configured to accept any URL:

1. First, the attacker implements a malicious website in our example: **attacker.com**.
2. The attacker sends an initial request to **victim.com**specifying the URL set to **attacker.com**.
3. The website forwards a POST request to its server.
4. The victim's server starts a headless browser and directs it to **attacker.com**.
5. The headless browser navigates to **attacker.com**.
6. **attacker.com** requests sensitive information that should only be accessible to the server.
7. The headless browser sends this request to the victim's server, resulting in **a Server-side Request Forgery (SSRF).**
8. Since the headless browser originals from the server, the server approves the request and sends the requested resources back to the headless browser.

9. The headless browser the forwards the stolen data to **attacker.com.**
10. **attacke.com** sends the data to the attacker's server, successfully exfiltrating private information.

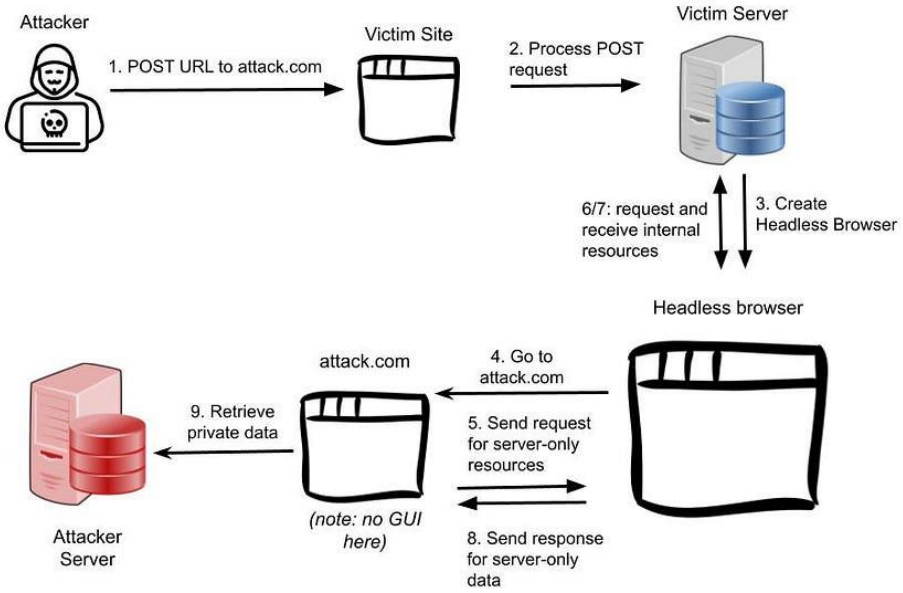


Figure 11: Exploitation of a Headless Browser for Server-Side Request Forgery (SSRF) and Data Exfiltration

5. Conclusion

Cybercriminals are increasingly using advanced techniques to target computer systems, incorporating social engineering tactics as well as advanced malware. Because many of these attackers are skilled and motivated, they often succeed in encrypting their communications, concealing their identities, distancing themselves from the proceeds of crime, and leveraging strong infrastructure that is difficult to compromise.

Thus, it has become most imperative to have advanced IDS capable of detecting contemporary as well as subtle threats. This means that the design of such systems should be based on an understanding of the state-of-the-art intrusion detection research level checking out their advantages and disadvantages.

Chapter 2: Intrusion detection system IDS

1. Introduction

The openness and dynamic nature of today's online communities have significantly amplified security concerns surrounding our digital assets. Intrusions and attacks are purposefully designed to compromise information and disrupt network communications. Recently, the frequency of these threats has escalated dramatically, propelled by the widespread reliance on computer networks. A recent report from Semantic highlights that they monitor events from 123 million attack sensors worldwide, successfully block 142 million threats each day, and track threat activities in over 157 countries [10]. To robustly defend computer networks, we utilize Intrusion Detection Systems (IDS), which effectively identify intrusions and attacks in real-time by analyzing network activities through statistics, rules, and sophisticated machine learning techniques.

2. Definition

Intrusion refers to any unauthorized activities that cause damage to an information system. This includes any actions that could threaten the confidentiality, integrity, or availability of information. For instance, activities that render computer services unresponsive to legitimate users are considered intrusions.

An IDS is a software or hardware tool designed to detect malicious actions on computer systems, in order to help maintain system security [11]. The primary goal of an IDS is to identify various types of malicious network traffic and computer usage that a traditional firewall may not detect. This functionality is crucial for providing robust protection against activities that could compromise the availability, integrity and confidentiality of computer systems, and intrusion prevention techniques (such as encryption, authentication, access control, secure routing, etc.).

The IDS that are being designed should satisfy the following requirements [12]:

- not introduce new weaknesses to the system.
- need little system resources and should not degrade overall system performance by introducing overheads.
- run continuously and remain transparent to the system and the users.
- use standards to be cooperative and open.

- be reliable and minimize false positives and false negatives in the detection phase.

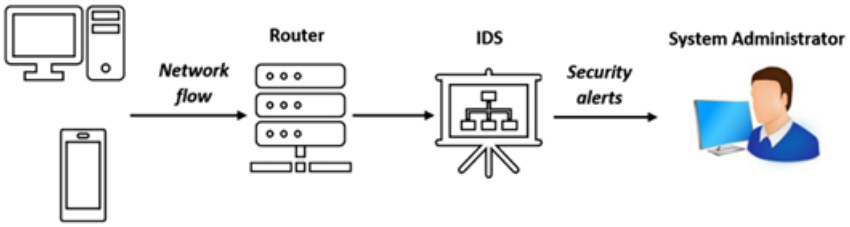


Figure 12: Intrusion detection system (IDS).

3. Classification of intrusion detection systems

Intrusion detection system IDS can be classified into various types based on their monitoring and analysis methodologies. Each type offers unique benefits and limitation. Additionally, all these approaches can be understood within the framework of a general model for IDSs.

3.1.Detection method:

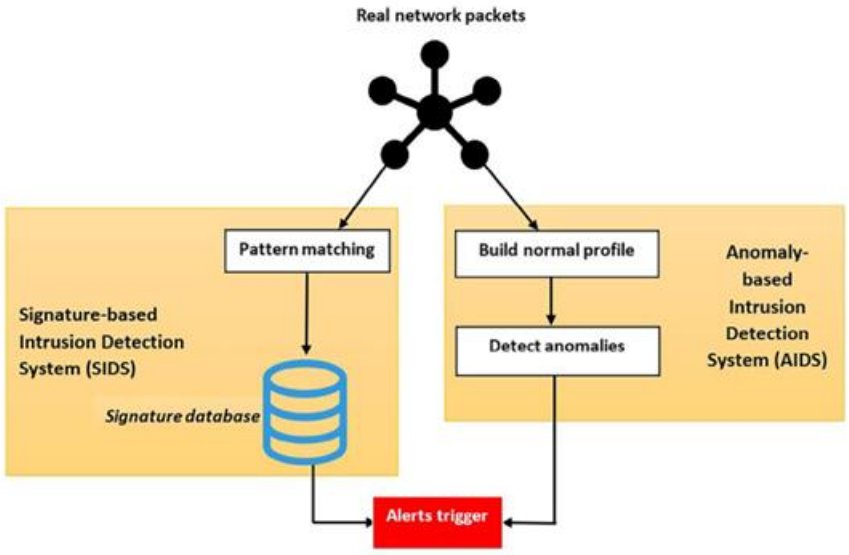


Figure 13: SIDS vs AIDS.

3.1.1. Anomaly-Based IDSs:

Anomaly-based intrusion detection system also called Behavior-based IDS,is advanced security mechanisms designed to identify unusual or suspicious user behavior or activities by comparing observed traffic patterns to a predefined baseline of normal behavior, it is able to detect not only known intrusion but also unknown intrusion.

AIDSs are particularly effective in detecting previously unknown or new attacks, as they do not rely on predefined signatures of known threats. However, their effectiveness can be challenged by the dynamic nature of network environments, such as evolving user behaviors, diverse traffic patterns, and the introduction of new applications or technologies. These changes can cause legimate activities to deviate from the established profile, leading to false positives or high false alarm rate.

Several researchers have proposed various techniques to enhances security in the IoT environment through anomaly detection. In 2020, Fitni and Ramli [13] proposed an anomaly-based IDS system using a mix of Decision Tree, Logistic Regression, and Gradient Boosting classifiers. The model was tested on the CSE-CIC-IDS2018 dataset, which include 80 features, with 80% of the dataset size used for training and 20% for validation [14].

Fu et al. [15] introduced an IDS designed to identify different types of attacks using data mining techniques. Their approach leverages intrusion semantic techniques to detect device misbehavior within an IoT network. The proposed system employs a slice time window method for intrusion detection, where device information is categorized based on time analysis. Anomalies are identified by comparing real-time data with a predefined normal profile. If any deviation is detected, the data are inconsistent and classified as an intrusion [16].

The Pseudocode of find normal pattern is shown in Figure 13:

```
1: Function [substring, time]=FindNormalPattern (T, δ)
2: start_time = 1
3: window_start =start_time + T
4: sax1 = SAXSymbolization ( S_start_time,…, S_window_start)
5: While (! find_normal_pattern)
6:        sax2 = SAXSymbolization ( S_window_start,…,
          S_window_start+T)
7:        window_start = window_start+1
8:        dist = Distance (sax1, sax2)
9:        If (dist < δ)
10:               find_normal_pattern = true;
11:               normal_parttern = (S_start_time, …,
                  S_window_start)
12:               cycle = window_start – start_time
13:       End
14: End
15: Return [normal_parttern, cycle]
```

Figure 14: finding normal pattern algorithm

For example: a bank client with a regular credit card who typically spends between 500$ and 1500$ per month, his transactions usually occur in New York, where he lives. However, one evening the bank detects a suspicious transaction: multiplehigh-value transactions in a short period, transaction from a foreign country, or purchases from unusual merchant categories.

The bank anomaly intrusion detection system flags this behavior as suspicious because it significantly deviates from the client usual spending patterns.

The system can take one of the following actions:

- ✓ Alert the user: by a SMS or an Email.
- ✓ Temporarily freeze the card: if the client did not respond.
- ✓ **Decline suspicious transactions.**

If the client confirms the transactions, the system updates his profile and learns his new behavior. If the client denies the transactions, the bank immediately blocks the card.

3.1.2. Signature-Based IDSs :

Signature-based intrusion detection system also called Misuse-based IDS, or Knowledge-based Detection, is the technique used by most commercial systems, because of the best defense against the known various network attacks.

This method is based on pattern matching techniques to find a known attack. Therfore, it depends on the receiving of regular updates of patterns. In misuse detection the user's activities are compared with the known patterns behaviors stocked in a large attack signatures database, if the activity match with any previous malicious detected attack, the IDS notifies the network administrators and triggers an alert.

This is why the database should be updated continuously in order to include all the new attacks patterns, which make it less effective against the zero-day attacks, which is a type of cyberattacks that exploits a previously unknown vulnerability in software or systems before developers can create and release a fix. Since these attacks are not yet documented in security databases, signature-based detection intrusion struggles to identify them. Figure 15 illustrate the mechanism of signature-based IDS

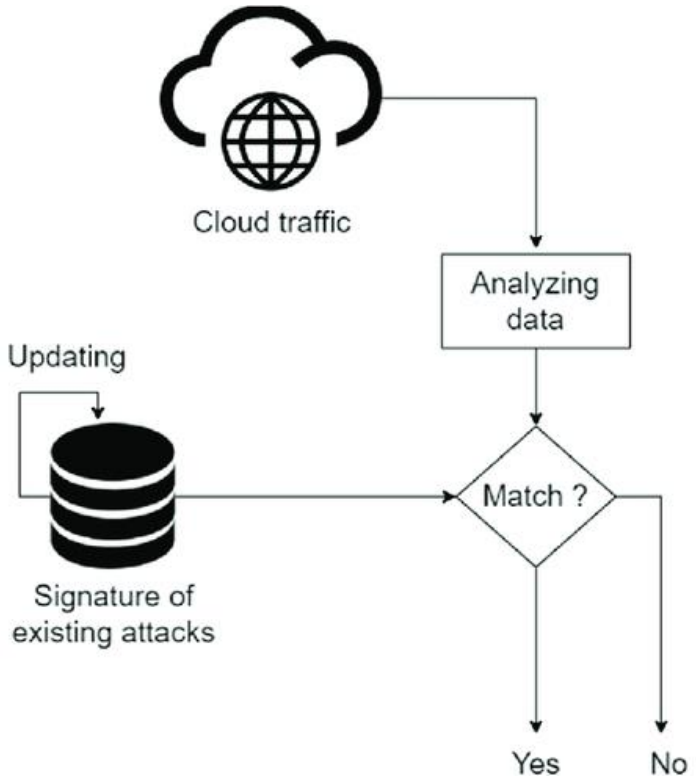


Figure 15: Signature-based intrusion detection.

.

Amin et al [17] used signature IDS to implement a mechanism for securing the IP-USN (IP-based Ubiquitous Sensor Network).in this IDS, the signature database is stored as an array and kept in the bloom filter. After sending the data packets to the bloom filters, if any match is found after the pattern checking, the packets stop processing and triggers an alarm signal [16].

Oh et al. [18] developed an intrusion detection system that detect various attack patterns using a matching engine with auxiliary shifting for early attack identification and termination. The system offers high detection accuracy, improved computational efficiency, scalability, and optimized memory usage. However, its limitations include reliance on predefined attacks signatures, restricting it to detect only known threats, and inability to operate in real time [16].

Sun et al [19]proposed an IDS for IoT-based cloud environment using cloud eye, which detects malicious activities through an intelligent lightweight agent scanner on devices. The server side maintains a large database of predefined attack patterns, updated periodically. The system uses Suspicious Bucket Cross Filtering SBCF to detect malicious data [16].

❖ Table 1: Comparison Between Anomaly-Based IDS and Signature-Based IDS

Feature	Anomaly-Based IDS	Signature-Based IDS
Detection method	Detects deviations from normal behavior	Matches activity with known attack signatures
New attackdetection	Can detect zero-day attacks	Ineffective against zero-day attacks
False Positives	Higher (may flag normal activities as anomalies)	Lower (detects only known threats)
False negatives	Lower (can detect unknown threats)	Higher (misses new attacks)
Database updates	Requires training on normal behavior	Need constant updates
Performance	Slower (requires learning and analysis)	Faster (compares with stored signatures)
Complexity	More complex (needs ML and behavior profiling)	Simpler (relies on predefined rules and patterns)

3.2. Information Sources

Intrusion Detection System IDS are commonly classified based on the type of data source they monitor. Some IDSs detect attacks by analyzing network packets captured from network backbones or LAN segments. Other identify intrusions by examining data generated by the operating system or application software.

3.2.1. Network-Based IDSs:

Network-based intrusion detection observe and analyze the network packets to detect attacks or any suspicious activity. NIDS has the ability to monitor the whole traffic owing to his position on the network devices such as routers and switches.

The detection of intrusion on the network-based IDS is operated through three stages [20]:

- ✓ Monitoring stage: it works to store and collect all the transmission-related data such as; the number of packets sent by devices, packet headers, content, size, port numbers, and established connection.

- ✓ Detection stage: in this stage NIDS use Misuse IDS and Anomaly IDS are used to collect data, and identify malicious network activities.
- ✓ Response stage: any detected attack or suspicious network activity is reported to the network administration for appropriate action.

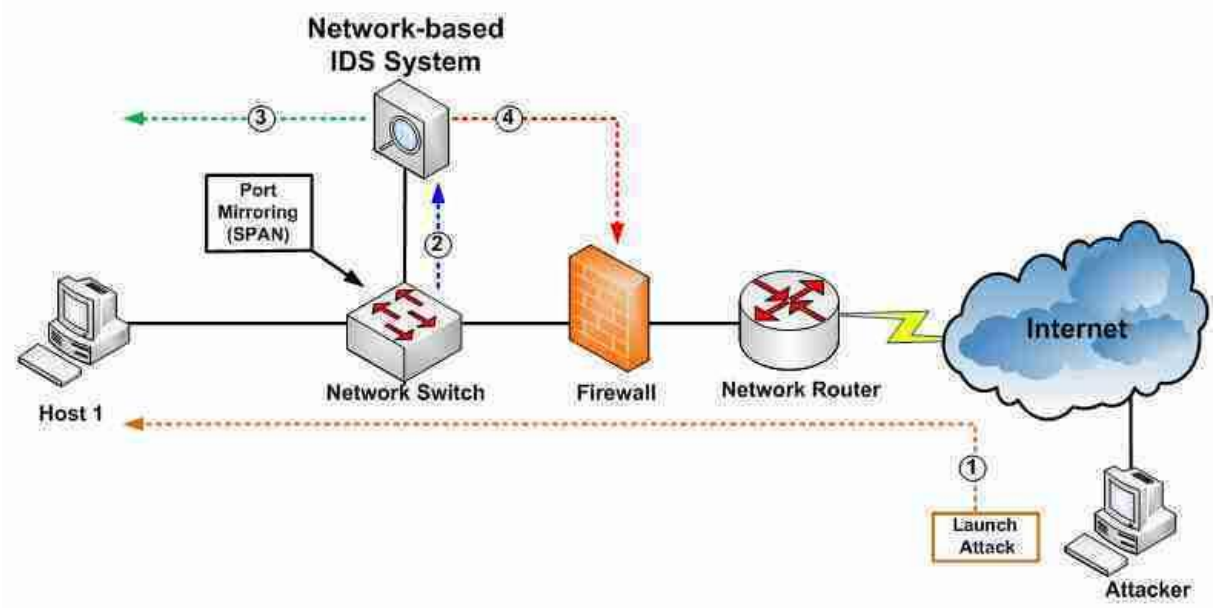


Figure 16: Network-based intrusion detection system.

Figure 16illustrate the concept of NIDS. The attacker lunch his attack to host 1 in step1 using the internet. The IDS uses promiscuous monitoring to see the attack via the port mirroring SPAN of the network switch which is connected to (step2), because it is not directly in the physical traffic path between the attack and the host, if any intrusion activity is detected the NIDS triggeres an alert (step3) and instructs the firewall to block the traffic via dynamic ACL (step4) [21].

Network-based IDS face challenges in high-traffic environment, as they may struggle to process all packets, potentially missing attacks during peak loads. Modern switch-based networks further reduce their effectiveness, as network segmentation restricts the IDS's monitoring scope, and even when switches provide monitoring ports, they may not capture all traffic. Additionally, network-based IDSs cannot analyze encrypted data, making them less effective as VPN usage increases. They also cannot confirm if an attack was successful, requiring manual investigation after detection. Lastly, some IDSs are vulnerable to fragmented packet attacks, which can cause instability and system crashes.

Mobile ad-hoc networks MANET are highly vulnerable to various routing attacks due the absence of central management. To address this issue,

Sivaneshet al [22] proposed an IDS called Accurate and cognitive IDS ACIDS, designed to detect black hole attacks, that has the ability to identifies anomalies by analyzing the normal behavior of key network parameters such as the destination sequence number (DSN) and route reply (RREP) [23].

In 2018, Belouch et al. [24] evaluated the performance of four classification algorithms: SVM, Naïve bayes, Decision Tree, and Random Forest for intrusion detection in network traffic using Apache Spark. The study utilized the UNSW-NB15 dataset, which contains 42 features for model training. Experimental results showed that random forest outperformed the other classifiers, achieving 97.49% accuracy, 93.53% sensitivity, and 97.75% specificity [14].

3.2.2. Host-Based IDS:

Host-based intrusion detection system is a software component positioned on the system, run on individual hosts/devices, which allows it to analyze activities with high reliability and accuracy, they can precisely determine which users are involved in an attack on the operating system. Their role includes managing system configuration files, monitoring recorded system logs, collects host-specific data such as system calls, DLL files, and registry keys, also analyzing potential integrity changes, and preventing unauthorized or malicious activities, any modifications to these files are compared against the existing security policy, ensuring an immediate response in case of an attack. HIDS provides real-time logging and reactions, with some of them able to monitor port events and restrict access to specific private ports.

ZarpelÃco et al [25] highlighted the importance of deploying host-based IDS, as they offer deeper monitoring of connected devices, enabling the detection of attacks that might go unnoticed using network-based alone.

Various HIDS, such as OSSEC [26] and Sagan [27], have been designed for traditional systems, offering comprehensive monitoring, alert correlation, and active response mechanisms.

Unfortuanatlly, HIDS are more difficult to manage, as they require configuration on each monitored host. Since they reside on the targeted system, they are vulnerable to attacks that could disable them. They are also ineffective at detecting network-wide threats, such as scans or reconnaissance, as they only monitor traffic received by their host. Additionally, certain denial-of-service DoS attacks can disable them. When relying on operating system audit trails, they generate large volumes of data, requiring significant storage capacity.

Lastly, they consume system resources, which may impact the performance of the monitored host. Figure 17 illustrates how Host-based intrusion detection system work.

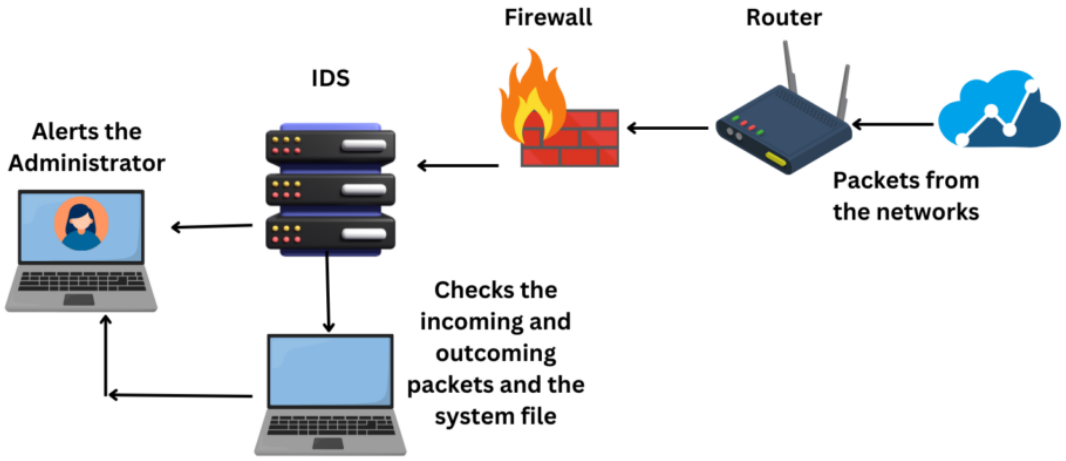


Figure 17: Host-based intrusion detection system.

Appu et al [28] proposed preprocessing system call patterns for machine learning-based Host-based IDS using the Bag-of-Words (BoW) technique, including BoW, BoW Boolean, BoW Probability, and BoW TF-IDF. These processed system calls patterns were then utilized to evaluate the classification and detection performance of various machine learning models, including Random Forest RF, J48, RIPPER, and KNN methods using two datasets namely the ADFA-LD dataset and the VMM malware dataset. Experimental results on the ADFA-LD dataset showed that the RF model with BoW TF-IDF achieved the best performance, with 98.4% accuracy (ACC) and 1.7% false alarm rate (FAR). Similarly, experiment on the VMM dataset demonstrated that J48, RF, and RIPPER with BoW TF-IDF were the most effective, achieving 100% accuracy and 0% false alarm rate [23].

Basant et al [29] proposed a real-time Host-based IDS for detecting abnormal system processes. This work represented system calls using n-gram feature vectors and computed TF-IDF values with the TFIDF Vectorizer method, applying Truncated Singular Value Decomposition SVD to reduce dimensionality. This HIDS used three detection methods: Support Vector Machine SVM, Neural Network NN, and Decision Tree DT. Performance evaluation on the ADFA-LD and ADFA-WD showed that:

- the best binary and multiclass classification performance using the SVM method with a 3-gram feature vector, obtaining FPR values of 3.34% and 9.12%, respectively.
- ADFA-WD demonstrated that the HIDS containing the NN method with a 5-gram feature vector exhibited the best performance. In binary and multi-class classification, this HIDS achieved FPR values of 8.63% and 15.11%, respectively.

Ribeiro, José, et al [30] proposed a HIDS for detecting malicious activities on Android mobile devices using a combination of statistical and machine learning ML methods. The system analyzes log files to identify intrusions. Two real-time datasets were generated using an android device which are:

- Statistical methods using the Univariate Gaussian Model UGM or Multivariate Gaussian Model MGM achieved 99.83% to 100% accuracy.
- ML-based HIDS (using classifiers like RF, DT, BN, OneR, LR …) reached 100% accuracy.

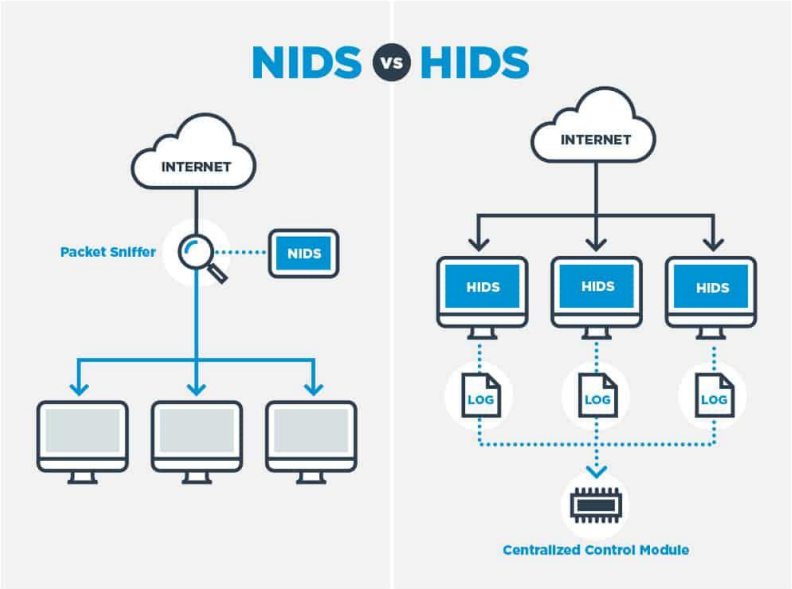


Figure 18: HIDS vs NIDS

Table 2: Comparison Between Host-Based IDS and Network-Based IDS

Feature	**Host-Based IDS**	**Network-Based IDS**
Data Source	Logs of operating system or application programs	Network traffic
Placement	Installed on individual servers or endpoint	Deployed on network devices (routers, switches...)

Response	Weak real-time response Good for long-term attacks	Strong response against outsider's attacks
Effectiveness against	Insider threats, malwares, unauthorized file modification	External threats, network intrusions, large-scale cyberattacks
Evasion risk	Attackers can disable or bypass it, if they gain access to the system	Encrypted traffic or segmented networks can limit the visibility
Limitation	Cannot analyze network behaviors	Monitor only the traffic passing through a specific network segment

4. Conclusion

Intrusion detection systems (IDS) play an important role in the overall defense strategy of computer and network systems and complement other security measures such as firewalls and unified threat management (UTM) solutions. Its main function is to monitor system activities, identify potential attack signatures or abnormal behaviors, and trigger appropriate alerts. Based on the detection method, IDS can be roughly divided into abuse-based (signature-based) systems and anomaly-based systems. From a deployment perspective, IDS are generally divided into network-based (NIDS) systems and host-based (HIDS) systems. Modern IDSs usually combine data from host and network sources to improve detection capabilities. In terms of effectiveness, an IDS is considered more reliable if it can accurately identify a variety of attacks while minimizing false positives to ensure a balance between security and availability.

Chapter 3: DNS Tunneling Detection Implementation

1. Introduction

The Domain Name System (DNS) one of the internet's core protocols. This protocol allows users to access websites and services by utilizing human-readable domain names instead of IP addresses. For example, it is far easier to access Google.com than to memorize its IP address, which is "8.8.8.8"(in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).

Emailing, web browsing, and a host of other internet activities are made easier by this mechanism. Nevertheless, DNS was not initially intended to manage data transit securely, despite its crucial function. It is therefore a desirable target for malevolent use.

DNS tunneling, a technique where attackers encapsulate data within DNS queries and responses to circumvent security measures and exfiltrate sensitive information, is one new threat that takes advantage of DNS.

Attackers can create secret communication channels by taking advantage of DNS traffic because it is frequently disregarded in conventional security solutions. Malicious uses for these channels include remote command execution, file exfiltration, and even complete remote control of infiltrated systems. By comprehending how DNS works and how it might be exploited, this work attempts to analyze DNS traffic and create an intrusion detection system that can spot unusual patterns suggestive of DNS tunneling.

2. DNS query/response form

The DNS operates as a structured naming system designed to manage domain name resolutions across various levels, to allow user to interact with device on internet by two types of DNS messages: Query message, and Response message. The format is similar for both types of messages, they consist of five sections:

- Header.
- Question.
- Records.
- Answer records.
- Authoritative records.
- Additional records.

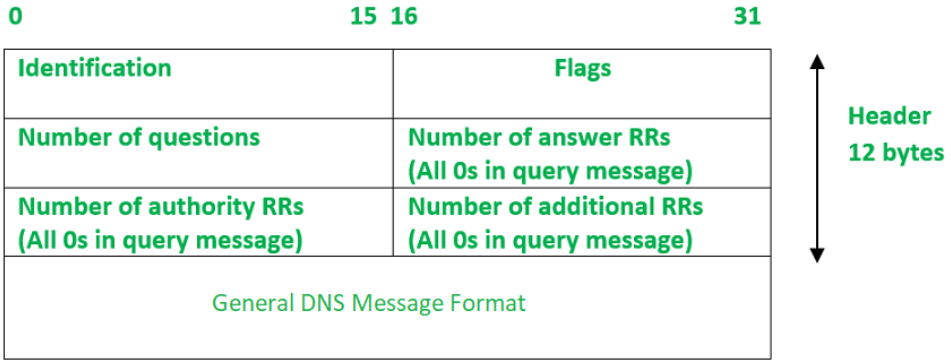


Figure19: DNS message format

At the highest point of this structure lies the DNS root, responsible for overseeing the resolutions of Top-Level Domains (TLDs). Below the DNS root, Second-Level Domains are assigned to resolvers for management. A Fully Qualified Domain Name (FQDN) comprises labels from all levels of the domain, pinpointing its precise location within the DNS hierarchy, starting from the root down to the lowest level. For instance, in the case of www.example.com, the TLD is com, and the SLD is example.com.

When a client seeks to resolve a domain, the process typically begins with a recursive DNS resolver. This resolver, which can be designated by the Internet Service Provider (ISP) or chosen by the user, initiates the resolution process. As depicted in Figure 19, the recursive resolver sequentially communicates with the root, TLD, and SLD name servers to resolve the domain name before furnishing the answer to the client. Hence, attempting to intercept DNS traffic directed towards a recursive resolver directly would disrupt this resolution process.

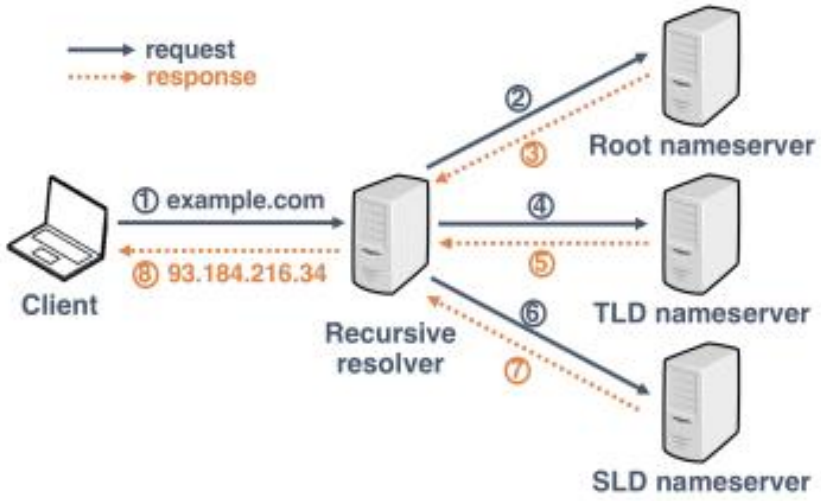


Figure 20: Domain resolution process with a recursive resolver.

3. DNS Tunneling

The first known discussion of DNS tunneling was by Oskar Pearson on the Bugtraq mailing list in April 1998. Since that time, a number of DNS tunneling utilities have been developed: most of the utilities use the same techniques with differences in encoding.

DNS tunneling is based on a client-server architecture. Typically, the client is located on the organization's internal network, while the server is hosted outside on the Internet. Communication between the two occurs through the organization's existing DNS infrastructure. The tunnel itself does not provide encryption; the encapsulated traffic is transmitted via DNS packets. To increase data confidentiality, attackers can implement additional layers within the DNS tunnel (such as VPN, often referred to as double tunneling), or use SSH connections with port forwarding.

4. Implementation Environment and Tools used:

4.1. Oracle VirtualBox:

Oracle VM VirtualBox is a cross-platform, open-source virtualization software that can run multiple operating systems simultaneously on a single machine. Developers can use VirtualBox to run and test different operating systems on their computers, allowing them to deliver code faster. IT teams and solution providers use VirtualBox to reduce operating costs and accelerate the secure deployment of applications on-premises and in the cloud. Designed for IT professionals and developers, Oracle VM VirtualBox runs on Windows, macOS, Linux, and Oracle Solaris systems. It is ideal for testing, developing,

demonstrating, and deploying solutions across multiple platforms on a single machine.

4.2. Ubuntu 20.04.2 LTS (Long Term Support):

Ubuntu is a Linux distribution based on Debian and composed primarily of free and open-source software. developed by the British company Canonical and a community of contributors under a meritocratic governance model [31], It was first released in 2004 when Mark Shuttleworth and a small team of Debian developers founded Canonical and then launched the Ubuntu project. Canonical released its first official version of the OS -- Ubuntu 4.10 -- in October 2004 [32]. And Ubuntu 20.04.2 LTS, one the latest version of ubuntu, also includes a number of security improvements for those developing and distributing software within the Ubuntu ecosystem.

4.3. Wireshark:

Wireshark is a powerful, open-source network protocol analyzer that allows users to captured and interactively browse the traffic running on the computer network, providing deep inspection of hundreds of protocols. This tool is freely available for use, with its source code accessible under the GNU General Public License (GPL). The Wireshark project is supported by a vibrant community of developers and users who contribute to its ongoing development and enhancement [33].

4.3. Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum, and released in 1991 [34].It is used for web development (server-side), software development, mathematics and system scripting. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance [35]. Python supports modules and packages like scapy wich is a powerful interactive packet manipulation library able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. Scapy runs natively on Linux, macOS, and on Windows [36].

4.4. Iodine:

Iodine is a DNS tunneling program first released in 2006 last updated in 2010. It was developed by Bjorn Andersson and Erik Ekman. Iodine is written in C and

it runs on Linux, MacOS and Windows. It creates a network interface on each client and connects them as if they were using the same network. This feature is unique to Iodine as other DNS tunneling tools focus on tunneling specific ports instead of all IPv4 traffic. This allows computers to ping each other and access all UDP/TCP ports and all other protocols encapsulated by the IP header. Iodine can tunnel IPv4 data through a DNS server. This tool runs on Linux, Mac OS X, FreeBSD, NetBSD, OpenBSD and Windows [37].

Two main components are required to setup the DNS tunneling:

- The server: a virtual machine running ubuntu 24.04.2 LTS, connected to the network with an IP address from the internal LAN IP address range (192.168.1.13). Iodine was installed on this machine using:

```
abir@ubuntu:~$ apt install iodine
```

Figure 21: Iodine installation command.

Then start the DNS tunneling with the command:

```
abir@ubuntu:~$ sudo iodined -f -P password 10.0.0.1 tunnel.test
[sudo] password for abir:
Opened dns0
Setting IP of dns0 to 10.0.0.1
Setting MTU of dns0 to 1130
Opened IPv4 UDP socket
Listening to dns for domain tunnel.test
```

Figure 22: Iodine tunnel established from the server.

- ✓ -f option: keeps iodine running in the foreground.
- ✓ -P option: specifies the password to be used between the client and the server.
- ✓ 10.0.0.1: is a virtual IP address that will assigned for the tunnel interface on the server side.
- ✓ Tunnel.test: virtual DNS domain.

Once the command above is entered on the server, Iodine creates a virtual tunnel interface and start listening for DNS queries on UDP port 53.

```
abir@ubuntu:~$ ifconfig
dns0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1130
        inet 10.0.0.1  netmask 255.255.255.224  destination 10.0.0.1
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 2  bytes 96 (96.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 48 (48.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fd00::1ecd:189b:9f5e:7a64  prefixlen 64  scopeid 0x0<global>
        inet6 fd00::a00:27ff:fea9:cdd4  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::a00:27ff:fea9:cdd4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a9:cd:d4  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 4770 (4.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 117  bytes 17585 (17.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.13  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::2d5:b7f0:11e9:9c1e  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b1:a8:77  txqueuelen 1000  (Ethernet)
        RX packets 1868  bytes 194807 (194.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1265  bytes 126943 (126.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4526  bytes 352204 (352.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4526  bytes 352204 (352.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 23: Output of ifconfig on the Iodine server.

- The client: is a virtual machine too like the server machine, connected to the network with an IP address, iodine installed on this machine too, Then the DNS tunneling start with the command:

```
victim@victim:~$ sudo iodine -f -P password -r 192.168.1.13 tunnel.test
[sudo] password for victim:
Opened dns0
Opened IPv4 UDP socket
Sending DNS queries for tunnel.test to 192.168.1.13
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Setting IP of dns0 to 10.0.0.2
Setting MTU of dns0 to 1130
Server tunnel IP is 10.0.0.1
Skipping raw mode
Using EDNS0 extension
Switching upstream to codec Base128
Server switched upstream to codec Base128
No alternative downstream codec available, using default (Raw)
Switching to lazy mode for low-latency
Server switched to lazy mode
Autoprobing max downstream fragment size... (skip with -m fragsize)
768 ok.. 1152 ok.. ...1344 not ok.. ...1248 not ok.. ...1200 not ok.. 1176 ok..
1188 ok.. will use 1188-2=1186
Setting downstream fragment size to max 1186...
Connection setup complete, transmitting data.
```

Figure 24: Iodine tunnel established from the client.

- ✓ -f option: keeps iodine running in the foreground.
- ✓ -P option: specifies the same password used in the server.
- ✓ 192.168.1.13: the IP address of the server.
- ✓ Tunnel.test: the same virtual DNS domain used in the server.

Once the command above is entered on the client, Iodine creates a virtual tunnel interface and start listening for DNS queries on UDP port 53, and the IP address 10.0.0.2 is assigned to the tunnel interface on the client side.

```
victim@victim:~$ ifconfig
dns0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1130
        inet 10.0.0.2  netmask 255.255.255.224  destination 10.0.0.2
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 48 (48.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 25: Output of ifconfig on the Iodine client.

To test the connection between the server and the client, a simple test of the connection was done by pinging the server IP address 10.0.0.1 from the client. Then, the results was captured with the Wireshark.

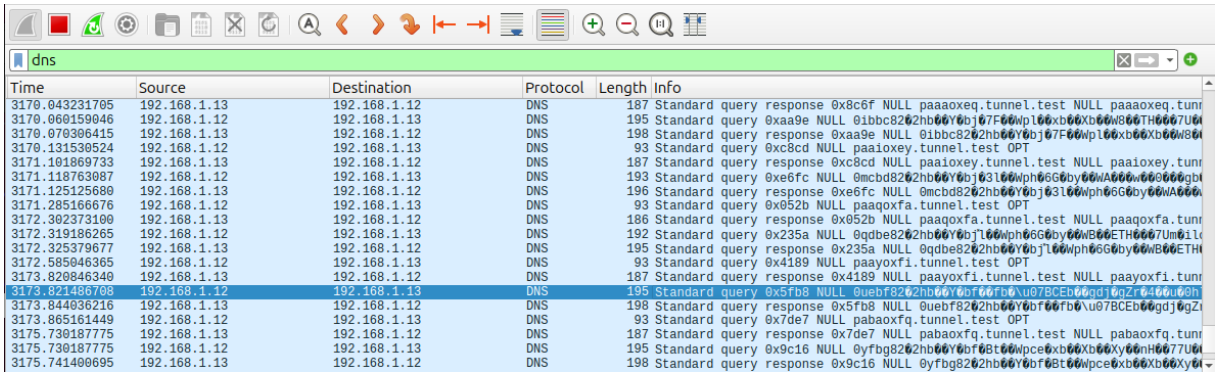


Figure 26: packets captured by Wireshark

Iodine breaks up the tunneled traffic into small packets that fit into the NULL field of the DNS packet. The encapsulated packets are sent individually through the tunnel and reassembled at the other end.

Detecting this type of activity would be difficult for a simple firewall. However, this does not pose a challenge for the IDS proposed here.

In the next section, the proposed IDS is introduced with its architecture, and show how it can effectively detect DNS tunneling activities generated using Iodine.

To evaluate the effectiveness of the IDS, we implemented a custom attack scenario. This attack simulated the behavior of malware written in Python and encoded with Base64 to bypass basic content inspection mechanisms. The encoded data was then broken up and transmitted through multiple DNS packets, simulating real-world exfiltration or command and control (C2) communications through DNS tunnels. This approach enables realistic and controllable simulation of DNS-based data transmission and provides a reliable test case for validating the detection capabilities of the IDS.

Additionally, the DNS packets used in this attack targeted a spoofed domain name(tunnel.test). This was intentionally chosen so that it would not appear on the whitelist of authorized domains that the company normally accesses. This anomaly in domain name usage is an important indicator for an IDS. By inspecting DNS queries and comparing the domain name to a predefined whitelist, an IDS can flag unauthorized or suspicious traffic. Therefore, the presence of unknown and unlisted domain names, coupled with the unusual volume and pattern of DNS requests, allowed the IDS to successfully detect the attack.

The IDS immediately creates a thorough incident report when it detects such an attack. Important details like the attacker IP address, the suspicious domain that was searched, the timing of detection is all included in this report, also a graph of detection time per suspicious DNS packets. The system administrator is then emailed the report, enabling a rapid response and more research. This notification technique ensures that security professionals are instantly aware of potential DNS tunneling activity within the network.

In order to create the Intrusion Detection System (IDS), various Python tools and libraries were intergrated. These tools enabled the system to perform real-time analysis of network traffic and to automatically generate alerts.

The main component of the system is Scapy, a robust library used for manipulating network packets. With Scapy, the IDS was able to monitor DNS traffic on port 53 in real-time and extract important information such as source IP address, destination IP address, and the queried domain.

also, to validate domains and generate alerts, the following tools were used in the system:

- Network analysis:

Scapy.all (sniff, DNSQR, DNS, conf, get_if_list): sniff DNS packets in real-time on port 53 and extrat query fields such as domain names and Ips.

- Data management and configuration:

Csv whitelist: loads whitelist file which contains trusted domains.
Socket: retrieves the local IP address.

- Detection performance tracking:

Time: tracks how long it takes to analyze each DNS packet and store detection latency.
Matplotlib.pyplot: plots a graph of detection time per suspicious DNS packets.

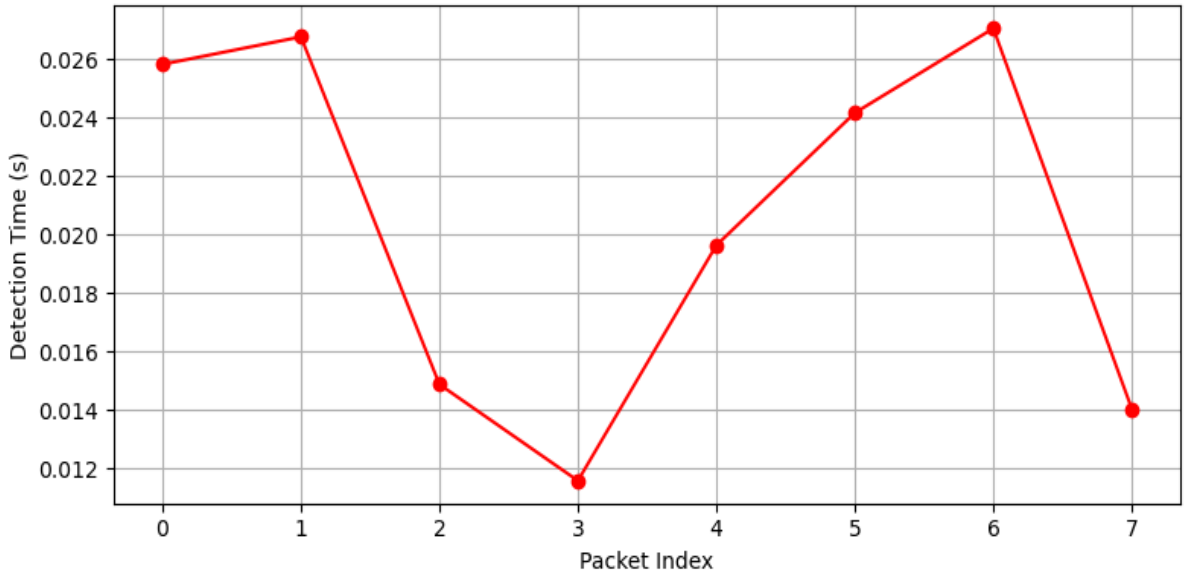


Figure 27: Detection time per suspicious DNS packets

5. Conclusion

The system is capable of effectively identifying any efforts to extract data via DNS, even in cases where advanced methods such as using fake domains or dividing base64-encoded data into multiple packets are employed. Its adaptable structure enables simple scalability to tackle new attack techniques or to merge with more comprehensive security platforms.
This tool provides a simplified and automated method for enhancing security, encompassing both detection and response features. These functionalities play a vital role in contemporary network defense strategies.

General Conclusion

With rapid technological advances and the explosive growth of digital data, cybercriminals are increasingly using sophisticated attack techniques and social engineering tactics to compromise computer systems. Attackers are increasingly able to hide their identities, obfuscate communications, and exploit powerful infrastructure that is difficult to track or disrupt. Therefore, it is more important than ever to protect systems with advanced intrusion detection systems (IDS).

As part of this project, we investigated various types of cyberattacks, including denial of service (DoS), phishing, malware, and DNS tunneling - an advanced technique for data exfiltration via legitimate DNS traffic. These examples highlight the need for detection mechanisms that can adapt to evolving threats.

Given the critical role that IDS plays in protecting network and system resources, it is increasingly important for organizations to implement them as an early defense mechanism against malicious activity. IDS research continues to attract significant interest from both academia and industry, resulting in the continued development of experimental and commercial systems.

As part of this final project, a custom IDS programmed in Python was proposed using the Scapy library specifically designed to detect DNS tunneling attacks between two virtual machines. The tool successfully identified covert file transfers hidden in DNS packets, demonstrating the real-world effectiveness of a custom intrusion detection system in a controlled environment.

Bibliography

[1] M. S. Roza Dastres, " A Review in Recent Development of Network Threats and Security Measures," International Journal of Information Sciences and Computer Engineering, 2021.

[2] S. S. A. M. O.-O. A. Y. A. Ömer Aslan, "A Comprehensive Review of Cyber Security Vulnerabilities,," Wojciech, Ankara, Turkey, 2023.

[3] F. G. E. C. a. T. D. Michael Zipperle, "Provenance-based Intrusion Detection Systems: A Survey," Michael Zipperle, Canberra, Australia, 2022.

[4] R. Gargan, "network-attacks," 12 August 2024. [Online]. Available: https://www.netmaker.io/resources/network-attacks. [Accessed 14 02 2025].

[5] N. G. A. J. P. Jibi Mariam Biju, "CYBER ATTACKS AND ITS DIFFERENT TYPES," Kerala, India, 2019.

[6] D. K. B. a. J. K. K. Nazrul Hoque, "Botnet in DDoS Attacks: Trends and Challenges," 2015.

[7] S. Marshall, "Lepide," 23 December 2024. [Online]. Available: https://www.lepide.com/blog/common-types-of-network-attacks/.

[8] E. B. T. German Rodriguez, "Cross-Site Scripting (XSS) Attacks And Mitigation: A Survey," German Rodriguez, Faculty of Systems Engineering Escuela Politecnica Nacional,Quito,Ecuador, 2019.

[9] B. C. ,. D. U. *. a. F. B. Filippo Sobrero *, "Towards a Near-Real-Time Protocol Tunneling Detector Based on Machine Learning Techniques," *Cybersecurity and privacy,* p. 14, 6 November 2023.

[10] Broadcom, "Internet Security Threat Report (ISTR) 24,," Symantec Enterprise Division, 2019.

[11] C.-H. R. L. Y.-C. L. a. K.-Y. T. H.-J. Liao, "Intrusion detection system: comprehensive review," in *J Netw Comput Appl*, 2013.

[12] S. D. M. a. R. S. Ismail Butun, "A Survey of Intrusion Detection Systems in Wireless Sensor Networks," , FIRST QUARTER 2014.

[13] Q. R. S. F. a. K. Ramli, ""Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems," IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 2020, 2020.

[14] A. M. A. Azar Abid Salih, "Evaluation of Classification Algorithms for Intrusion Detection System: A Review," Universiti Tun Hussein Onn Malaysia Publisher’s Office, Duhok, Kurdistan Region, IRAQ , 2021.

[15] K. Z. Z. Y. Rongrong Fu, "An Intrusion Detection Scheme Based on Anomaly Mining in Internet of Things," in Proceedings of the 4th IET International Conference on Wireless, Mobile and Multimedia Networks, Beijing 100876, China, June 2011..

[16] M. ,. K. S. V. N. Santhosh Kumar, " A Comprehensive Survey on Machine Learning-Based Intrusion," p. 24, 27 january 2023.

[17] M. S. C. S. H. a. J. C. S. O. Amin, "A novel coding scheme toimplementsignature basedIDS inIPbased sensor networks," p. 269–274, june 2009.

[18] D. K. ,. W. R. Doohwan Oh, "A Malicious Pattern Detection Engine for Embedded Security Systems in the Internet of Things," p. 14(12), 21 december 2014.

[19] X. W. R. B. J. S. Hao Sun, "CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained internet of things (IoT) devices," in *Software: Practice and Experience*, vol. 47, 2016, pp. 421-441.

[20] H. Satilmiş, "A Systematic Literature Review on Host-Based Intrusion Detection Systems," IEEE Access, Turkey, 2024.

[21] "Intrusion Detection and Prevention – IDS/IPS," How to network, [Online]. Available: https://www.howtonetwork.com/technical/security-

technical/intrusion_detection_and_prevention/#Chapter_Summary. [Accessed 13 march 2025].

[22] S. S. &. V. R. S. Dhulipala, " Accurate and Cognitive Intrusion Detection System (ACIDS):," *A novel Black Hole Detection Mechanism in Mobile Ad Hoc Networks,* p. 9, 2021.

[23] S. A. ,. A. Z. Y. T. HAMİ SATILMIŞ, "A Systematic Literature Review on Host-Based Intrusion Detection Systems," IEEE Access , Tubitak Turkey, 2024.

[24] S. E. H. M. I. Mustapha Belouch, "Performance evaluation of intrusion detection based on machine learning using Apache Spark," in *Procedia Computer Science,*, ISSN 1877-0509, 2018, pp. 1-6.

[25] R. S. M. C. T. K. S. C. d. A. Bruno Bogaz Zarpelão, "A survey of intrusion detection in Internet of Things," in *Journal of Network and Computer Applications*, ISSN 1084-8045, 2017, pp. 25-37.

[26] D. A. H. a. R. B. Cid, "host-based intrusion detection guide," Syngress, OSSEC, 2008.

[27] N. E.-J. ,. J. M. F. D. A. a. M. R. D. Robin Gassais, "Multi-level host-based intrusion detection system for Internet of things," 2020.

[28] G. J. W. K. S. P. V. S. R. N. Appu Alfred Raja Melvin, "An AI powered system call analysis with bag of word approaches for the detection of intrusions and malware in Australian Defence Force Academy and virtual machine monitor malware attack data set," Expert systems, India, 18 may 2022.

[29] P. G. Basant Subba, "A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes,," ISSN 0167-4048, India, 2021.

[30] J. S. F. B. M. G. R. J. S. S. J. &. A.-A. R. A. Ribeiro, "An Autonomous Host-Based Intrusion Detection System for Android Mobile Devices," in *Mobile Networks and Applications*, 2019, p.

pages 164–172.

[31] W. contributors, "Ubuntu," Wikipedia, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ubuntu&oldid=1294356019. [Accessed 07 06 2025].

[32] R. Sheldon, "Definition Ubuntu," TechTarget, 08 Aug 2023. [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/Ubuntu. [Accessed 2025].

[33] "The world's leading network protocol analyzer," Wiresharrk, [Online]. Available: https://www.wireshark.org/. [Accessed 08 06 2025].

[34] "w3schools," [Online]. Available: https://www.w3schools.com/python/python_intro.asp. [Accessed 10 06 2025].

[35] "python," [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed 10 06 2025].

[36] "Scapy," [Online]. Available: https://scapy.net/. [Accessed 10 06 2025].

[37] M. A.-k. a. T. Khairallah, " Winning tactics with DNS tunnelling," Princess Sumaya University for Technology,, Jordan, December 2019.

[38] B. B. G. Shashank Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms:classification and state-of-the-art," Lulea University of Technology, India , Sweden, 28 November 2014.

[39] Roberto, "How Malicious Hackers Can Takeover Your Headless Browser:Examining Headless Browser + Server-Side Request Forgery (SSRF)," 21 july 2022. [Online]. Available: https://infosecwriteups.com/how-malicious-hackers-can-takeover-your-headless-browser-part-1-bcab9e3a2f9c. [Accessed 06 03 2025].

[40] Broadcom, "Internet Security Threat Report (ISTR) 24," Symantec Enterprise Division, [Online]. Available: , 2019.