International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.77, March 2025

Attack Information Gathering from Network Analysis Data during Scanning Activity

Stephane J. Tamafo
Department of Mathematics and Computer Science,
University of Dschang, Cameroon

Elie Fute Tagne
Department of Computer Engineering,
University of Buea, Cameroon
Department of Mathematics and Computer Science,
University of Dschang, Cameroon

Jaime C. Acosta
EVCOM Army Research Laboratory,
Network Security Branch, USA
jaime.c.acosta.civ@army.mil

Charles Kamhoua
DEVCOM Army Research Laboratory,
Network Security Branch, delphi, MD, USA

Rawat Danda
Department of Electrical Engineering and Computer Science,
Howard University, Washington, USA

ABSTRACT

The rise of cloud computing, remote work, and IoT has heightened the risk of cyberattacks, exposing sensitive data to advanced threats. Traditional security measures, such as cryptography and intrusion detection systems, often fail against zero-day exploits. This paper proposes a proactive approach to network security by identifying scanning tools and targeted services during the reconnaissance phase of an attack. By analyzing network scanning activities, it becomes possible to detect the tools, techniques, and targeted services used by attackers, enabling preemptive defense. The methodology involves capturing network traffic during scans, extracting key features, and using decision tree-based machine learning models to classify scanning tools, techniques, and services. Experiments conducted with the Weka tool demonstrate high accuracy in identifying scanning techniques (96.8%) and targeted services (98%). This approach provides critical insights into attackers' intentions, allowing for tailored defensive measures before an attack escalates. The results underscore the effectiveness of machine learning in enhancing network security by preemptively identifying and mitigating potential threats.

Keywords

Attack intention, cyber deception, decision tree, port scanning, scanning tools

1. INTRODUCTION

In recent decades, the rapid evolution of cloud computing, remote work, e-commerce, social networks, and the Internet of Things (IoT) has transformed how organizations operate and store sensitive information. The COVID-19 pandemic further accelerated these trends, pushing businesses to rely heavily on cloud-based services and remote infrastructures. While this shift offers significant advantages, it also exposes critical data—such as trade secrets, medical records, and financial information—to heightened risks, including human error, technical failures, fraud, espionage, and malicious attacks [5]. The cloud, in particular, has become a prime target for attackers, as breaches can originate from any direction and at any time, making robust security measures more critical than ever.

Traditional security mechanisms, such as cryptography, firewalls, antivirus software, and intrusion detection/prevention systems (IDS/IPS), play a vital role in protecting systems. However, these measures are often insufficient against advanced threats, particularly zero-day attacks, which exploit unknown vulnerabilities and can bypass conventional defenses [4]. As network technologies advance, attackers have developed increasingly sophisticated methods to conceal their activities, making it challenging to achieve comprehensive protection. This underscores the need for innovative strategies that can defend systems even after an attacker has gained access.

Cyber Deception is an advanced cybersecurity strategy that involves deploying honeypots and simulating fake systems, services, or information to attract, trap, and mislead attackers. By concealing critical assets behind carefully designed decoys, this approach not only protects infrastructures but also provides a unique opportunity to gather in-depth intelligence on the tactics, techniques, and procedures (TTP) used by cybercriminals [24**?** , 19]. This information enables defenders to better understand malicious intentions, anticipate attack patterns, and strengthen overall security posture [11].

However, the effectiveness of this approach relies on meticulous execution and a deep understanding of attackers' profiles. Poorly designed cyber deception, such as placing decoys on irrelevant or easily identifiable targets, can significantly reduce its impact and raise adversaries' suspicion, prompting them to adapt their strategies or avoid the traps [23]. Conversely, a well-orchestrated deception not only diverts attackers from critical systems but also prolongs their engagement, increasing the chances of identifying their motivations, command-and-control infrastructures, and even their affiliations with malicious groups [26, 27].
It is therefore essential to integrate cyber deception into a proactive, intelligence-driven security approach. The collection and analysis of actionable intelligence before an attack occurs allow defenders to tailor decoys to specific threats and maximize their effectiveness [9]. By combining this strategy with other mechanisms, such as early identification of attack objectives, it becomes possible to transform the digital environment into an unfavorable terrain for attackers, thus enhancing the resilience and robustness of systems against emerging cyber threats [22]. Since system scanning is the preliminary phase of attacks, it is advisable to perform this identification during this phase [21].
Port scanning is a common reconnaissance activity used in both defensive and offensive security. Network administrators frequently perform port scanning to identify and fix system vulnerabilities, enhancing security. However, attackers also use port scanning for the same reasons, but with the intent to exploit system weaknesses. Various scanning tools exist, each designed for specific objectives, whether assessing database security, analyzing web applications, evaluating network protocols, or auditing system infrastructure. Attackers select tools based on their intended targets, making it crucial to identify the scanning tool in use to anticipate the attacker's objectives and deploy effective countermeasures.
For example, an attacker aiming to exploit SQL injection vulnerabilities might use SQLMap, a tool designed to automate SQL injection testing on various database management systems (MySQL, PostgreSQL, etc.). Identifying SQLMap in use can alert defenders to potential database exploitation, allowing them to strengthen protections accordingly. Security-focused operating systems like Kali Linux and Parrot OS categorize scanning tools based on their functionalities, further highlighting their strategic importance in both attack and defense.
Based on their functions, scanning tools can be categorized as follows:

—**Database security assessment**: Identifies vulnerabilities in databases that could lead to data leaks, compliance violations, and major security breaches. Tools include SQLMap (automated SQL injection analysis), DbDat (database configuration auditing), NoSQLMap (NoSQL database security assessment), and AppSpider (web application database vulnerability detection).

—**Web application analysis**: Detects vulnerabilities in web applications, such as cross-site scripting (XSS) and misconfigurations. Tools include Unicorn (information collection and correlation), Nikto (weak configuration detection), and Burp Suite (comprehensive web application security analysis).

—**Network and protocol analysis**: Identifies vulnerabilities, analyzes data flows, and detects anomalies indicating security risks. Tools include Ike-scan, Ettercap, and RouterSploit.

—**Network auditing**: Evaluates IT infrastructure security, identifying misconfigurations and vulnerabilities. Popular tools include Nmap, Masscan, PortSpider, and Angry IP Scanner. While general-purpose, these tools can be tailored for specific tasks, such as analyzing network services (e.g., HTTP/HTTPS, MongoDB, MySQL, printer discovery).

—**IoT system analysis and exploitation**: Assesses security risks in IoT environments. Tools include Shodan, Thingful, and IoT-Seeker.

Attackers select their scanning tools based on their attack objectives, and identifying the tool used can provide crucial intelligence about the attack's intent. This information allows defenders to proactively secure targeted systems before an exploit occurs.
A key phase in cyberattacks is reconnaissance, where attackers analyze and gather information about a system. During this phase, attackers identify potential targets, open ports, vulnerabilities, and other compromising information about the system. Analyzing the tools and techniques used in this phase can help infer the attacker's goals and prepare tailored countermeasures. While previous studies have focused on analyzing attacker behavior during active attacks [20, 13], this reactive approach often allows attackers to achieve their objectives before any intervention. In contrast, this research proposes identifying the attackers' intentions during the reconnaissance phase by identifying the scanning tool, scanning technique, and target service during scanning from the scan traffic, enabling proactive defense and significantly reducing potential damage. This paper addresses this critical gap by making the following contributions:

—A comparative study of scanning tools and techniques, demonstrating the feasibility of identifying them from network traffic.

—A decision tree-based classification model to identify scanning tools from network traffic.

—A decision tree-based model to identify targeted network services during scanning activities.

The remainder of this paper is structured as follows: Section 2 provides a review of related studies on attack intent recognition. Section 3 outlines the methodology used for identifying scanning tools, techniques, and targeted services. Section 4 presents the experimental results, while Section 5 concludes the paper by discussing future research directions.

2. RELATED WORKS

The intent of an attack represents the primary objective that an attacker seeks to achieve by implementing various attack methods or techniques. However, predicting these intentions remains a complex task, even for cybersecurity experts [5]. Indeed, an attacker typically follows a sequence of steps while adopting concealment and stealth strategies to avoid detection. Understanding these intentions would allow security administrators to anticipate malicious activities and respond more effectively. In this regard, several research studies have been conducted to recognize attack intent, adopting various detection approaches.
One widely studied approach is based on **causal or Bayesian networks**, which model dependency relationships between different variables in the form of a directed acyclic graph. Several studies, such as [16], have explored the use of these networks to correlate and analyze attack scenarios, thus seeking to identify the tactics employed by attackers and predict their potential intentions. [17] combine Dempster-Shafer evidence theory with a probabilistic approach based on a causal network, allowing for the selection and prediction of real attack intentions while assessing the best possible responses.
Another extensively studied approach is that of **graphical methods**, which rely on representation in the form of directed graphs

$G = (S, E, S_I, S_F)$, where S denotes the set of states, E the transition relationships, SI the initial state, and SF the final state after a successful intrusion [15]. Several studies, such as those by [15], have used these graphs to model attack intentions, considering that the nodes represent system and attacker states, while the edges translate state transitions under the effect of malicious actions. [8] proposed an attack intent analysis method based on an attack path graph, integrating critical asset assessment to generate attack intent hypotheses aligned with the system's protection requirements. This approach thus facilitates the analysis of complex attack behaviors occurring in multiple stages.

The methodology based on **path analysis** aims to identify the attack plan from observations made by defense systems. The correlation of alerts and malicious actions is essential here to anticipate potential attacks and minimize their impact [5]. In this context, attackers can follow multiple possible paths represented in graph form, and these paths can be analyzed using attack trees to assess a system's vulnerability based on the attacker's objectives [5]. Among notable contributions in this field, [4] proposed an investigative approach based on attack intent recognition through a similarity analysis of attack characteristics with those in a recognition database. Their model relies on creating attack patterns as references and recognizing attack intentions in real-time by comparing detected attack signatures with those of pre-established models.

Although these approaches vary in their implementation, they all share a common point: attack intent identification generally occurs during the execution of the attack. This characteristic somewhat limits their relevance, as knowing the attack intent during the attack helps mitigate damage, but earlier anticipation would be even more beneficial in strengthening system resilience against cyber threats.

3. DATA AND METHODOLOGY

3.1 Data gathering

Collecting information about the attacker's intent before they take action proves crucial for defense, as such information could reveal the target, making it easier to develop an adequate and tailored defense strategy against the attack in preparation. Training a machine learning model capable of classifying scan traffic to identify the scanning tool and target service becomes just as important. However, there is no publicly available dataset for training such a model. Hence, the creation of a synthetic dataset becomes essential.

For the creation of the dataset, three physical computers were used to collect data in order to build the dataset. The first machine had the following specifications: i7 Core processor, 10th generation, 2.6 GHz frequency, 4 cores, 8 logical processors; 4 GB of dedicated memory; 32 GB of RAM; and a 1 TB SSD hard drive. On this first machine, three virtual machines were deployed (two with Ubuntu versions 18.04 and 20.04, and the last one with a Windows 10 operating system). The second machine had the following specifications: i5 processor from the 8th generation with a 2.2 GHz frequency, 2 cores, 4 logical processors, 2 GB of dedicated graphics memory, 8 GB of RAM, and a 128 GB SSD hard drive. On this machine, two virtual machines were deployed: one with Linux Mint and the other with Windows 10. The third machine had the following specifications: i5 processor from the 11th generation with a frequency of 2.44 GHz, 4 cores, 8 logical processors, 512 MB of dedicated graphics memory, 16 GB of RAM, and a 256 GB SSD hard drive. On this machine, a single operating system was installed: Kali Linux, to perform audits across the network. After deploying these virtual machines, they were connected to the network using the GNS3 network simulator. The architecture of the network is represented in Figure 1. Once the machines were networked, different analyses were performed, using one machine as the source (Kali Linux) and the others as targets. The process was repeated several times, changing the scanning strategy each time.

Several analysis tools were studied, with the most popular being Nmap (Network Mapper), Nessus, Masscan, AngryIpScanner, Unicorn, PortSpider, and others like IkeScan, SqlMap, and NiktoScan. Some of these tools, such as Nmap, Unicorn, and PortSpider, use several network analysis techniques that work in more or less different ways and achieve varying goals. A Python script using the T-shark library was written to intercept traffic passing through the source machine's network card. Several network analyses were then performed on the target virtual machines at different time intervals using each of the scanning tools considered, of course, taking into account their different scanning techniques for tools that implement more than one. This data collection also includes the 20 most commonly used services on computer networks, and analyses were performed on each of these services across the network. Among these services are:

—**DHCP**: Used for automatically assigning IP addresses to devices on a network.

—**FTP**: Used for transferring files between a client and a server over a network.

—**IMAP**: Allows retrieving and managing emails from a mail server.

—**HTTP/HTTPS**: Used to transfer web pages, images, and other content on the web.

—**DNS**: Enables domain name resolution.

—**MySQL**: Open-source relational database management system.

—**SSH**: A cryptographic network protocol used to securely access remote machines.

—**SMTP**: Used for sending emails from a client to a mail server or between servers.

Once the scans were completed, the captured traffic was saved in JSON format. This approach facilitates efficient analysis and processing of network data, as JSON is a widely used format for representing structured information. For instance, tools like Wireshark allow users to export packet dissections in JSON format, enabling detailed examination of network traffic.

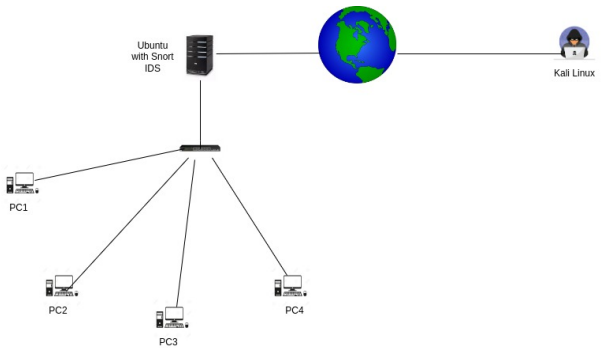

Fig. 1. Network architecture deployed on GNS3.

A comparative study of packets captured during scans performed with various scanning tools and techniques was conducted to determine which information encapsulated in the packets identifies

the scanning tool, the technique used, and the target services. Figures 2, 3, and 4 show how the values of a frame's attributes can vary from one scanning tool to another. Following this data analysis, the JSON files were consolidated into a single folder according to a specific naming convention. A Python script was then developed to analyze all the JSON files in the folder. Within each file, the script traverses each frame to extract the values of the aforementioned properties, to which a class is assigned based on the scanning tool, scanning technique, or target service, in order to construct a CSV file. The pseudocode for this script is described in Algorithm 1. It is important to note that two separate files were created: one for the identification model of scanning tools and techniques, and the other for the identification of target services. In this collection, only the frames sent by the source were considered, not the response frames, because it is assumed that in the network monitoring module, the interest lies solely in the incoming flow and not the outgoing flow of the network; therefore, the response frames are not relevant. These response frames cannot help identify the port scanning tool used, the scanning technique employed, or the target service. After executing the first script, the CSV file is passed to a second script that processes it line by line to remove duplicates. Since the first script processes file by file, and since the files come from the same tool and the same technique, or scan data from the same network service, samples of the same class are grouped based on their names. The CSV file is then passed to a third script to randomize its content. After this stage, the Weka data mining tool was used to transform the CSV file into ARFF format, thus allowing the machine learning models to be directly stored in Weka.

Algorithm 1 Extraction of frame characteristics

```
Define the list of fields to extract (FIELDS)
Add the JSON files to a list json_files
Open the CSV file in write mode
for each JSON file in json_files do
    Open the JSON file in read mode
    Load the packets from the JSON file into packets
    for each packet in packets do
        Initialize an empty dictionary to store the extracted features
        for each field in FIELDS do
            Split the field into a list of keys (in case of nested fields)
            Initialize a variable to store the value
            if the packet contains the key corresponding to the first key of the field then
                Extract the value by following the nested keys
                if the value is a number (int or float) then
                    Assign the value to the corresponding feature
                else
                    Set the feature to 0
                end if
            else
                Set the feature to 0
            end if
        end for
        Write the extracted features to the CSV file
    end for
end for
```

3.2 Fields description

After analyzing the various files captured within the network, several frame properties have been identified that, when combined, allow for the determination of the scanning tools used, as illustrated in Figures 2, 3, and 4. These properties have enabled the establishment of multiple characteristics for the dataset, described as follows:

—**Protocol**: Lists the transport protocol used, which is valuable because, depending on the analysis tools and techniques, the protocol may vary more or less between UDP, TCP, ICMP, and many others. These values are made of characters and are heavier to handle than numeric values, each protocol was replaced with its number. For example, if the protocol is ICMP, the field will have the value 1 (6 for the TCP protocol and 25).

—**IP.len**: Denotes the size of the IP layer within the frame. Based on the comparative study, this value varies between tools and scanning techniques. Contrary to common assumptions, this size does not necessarily depend on the IP address itself.

—**IP.flags:** Identification, flags, and fragment displacement are fields that allow the fragmentation of datagrams. This field contains three different properties; This field contains three different properties; the two most commonly used are: DF (don't fragment), which indicates whether the datagram can be fragmented or not. If a datagram has this bit set to one and the router cannot route it without fragmenting it, then the datagram is rejected with an error message; MF (More Fragments) indicates whether the datagram is a data fragment or not depending on whether this bit is set or not. If the indicator is zero, it indicates that the fragment is the last one (ie. the router should have all the previous fragments) or that the datagram has not been fragmented.

—**TCP.flag**: Represents the TCP flag that the frame carries. It is valid only for TCP frames, specifies the type of frame (synchronization frame, urgent frame, etc.), and can also identify the TCP scanning technique used. There are about different TCP flags, each with a different meaning: SYN, to request synchronization; ACK, which indicates that the connection request has been accepted; RST, which is the breaking or denial of the connection; FIN, which requests the end of the connection; URG, which specifies the urgency of the frame; NS, which signals the presence of congestion; CWR, which indicates that the congestion has been dealt with; and PSH, which indicates that the data should be sent immediately. Because all these values are strings, to facilitate manipulation, they were transformed into numerical values by representing them as 8 different features that take the value 0 if it is not the flag of the frame and 1 otherwise. For transport protocol frames other than TCP, all these fields have the value 0.

—**DST.port**: A destination port, which represents the target port in the network and allows the identification of the target service. This attribute is important because some tools, like PortSider, implement modules that scan the network on a particular port to identify weaknesses in a given service.

—**UDP.length**: The size of the information block associated with the transport layer for a UDP frame. This is only for UDP frames; for TCP, this value will be 0.

—**FRAME.encap.type**: There are three types of encapsulation, (namely, member variable encapsulation, function encapsulation) and class encapsulation, each of which has its meaning concerning the member variables of a class. Some scanning tools among those considered can be uniquely identified from their encapsulation type, like Unicorn.

—**TCP option**: The TCP provides optional header fields. The options are taken into account by the checksum. This field has several properties (no operation, sack permitted, MSS (maximum segment size), etc.). This field is not only optional but also only for the TCP; therefore the values of the different fields related to the TCP options for packets that do not have them have the value 0.

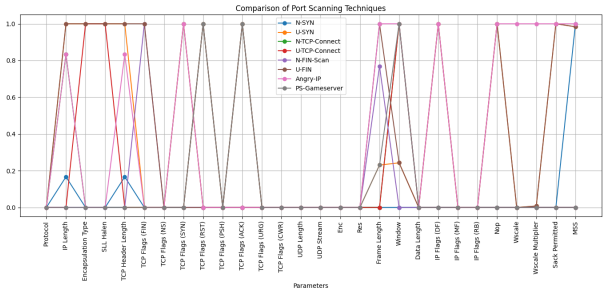

Fig. 2. Comparison of port scanning techniques across different tools. * N: Nmap, U: Unicorn, PS: PortSpider. Variable values indicate dynamic or context-dependent parameters.

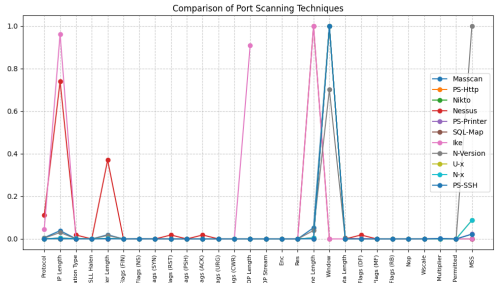

Fig. 3. Comparison of port scanning techniques across different tools. * PS: PortSpider, N: Nmap, U: Unicorn. Variable values indicate dynamic or context-dependent parameters.

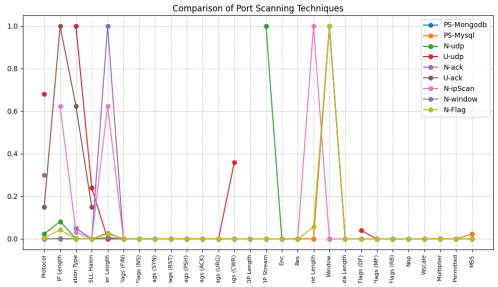

Fig. 4. Comparison of port scanning techniques across different tools.* PS: PortSpider, N: Nmap, U: Unicorn. Variable values indicate dynamic or context-dependent parameters.

In figures. 2, 3, 4, N refers to Nmap, U to Unicorn, PS to PortSpider, and Ike to IkeScanner.

3.3 Architecture and methodologie

There are many different types of attacks, each differing in technique, target, propagation method, and objective. However, a common stage among all these attacks is the system reconnaissance phase. During this phase, attackers gather information about the system, identify targets, and detect potential vulnerabilities. This stage is referred to as the reconnaissance phase [6, 14]. Depending on the type and objective of the attack, as well as its target, specific scanning tools and techniques may be employed. Identifying the tools and techniques used in network port scanning can help infer the attacker's intent or at least narrow down the list of possible attacks, as some scanning methods are tailored for particular tasks. Anomaly detection and intrusion detection and prevention systems for port scanning have been extensively studied in the literature [12, 10]. This work focuses solely on port scanning detection. The proposed architecture considers only external attackers and is limited to analyzing incoming network traffic from external sources (Internet).

Since all communication with the external environment passes through a gateway—acting as the interface through which data enters and exits the network—the proposed architecture assumes the presence of a reliable port scanning detection module (IDS/IPS) at this critical point. This module continuously monitors and filters incoming traffic, effectively identifying any network scanning activity. Normal traffic flows uninterrupted through the network, ensuring minimal disruption to legitimate communications. Upon detecting scanning activity, malicious traffic is redirected to an identification module for further analysis. This identification module is composed of two specialized detection components: one dedicated to recognizing the scanning tools and techniques used in the traffic, and another focused on identifying the targeted network service (if an attack on a specific service is detected), as illustrated in Fig. 5. Both sub-modules within the identification module leverage machine learning to classify scanning tools and techniques, as well as to identify targeted services. Given the imbalanced nature of the dataset, undersampling was applied to the majority classes. Decision trees were selected for model training, as they are known to be less sensitive to class imbalance, as highlighted in [**?**]. Before the learning phase, data normalization was performed to ensure all variables were on a similar scale, preventing certain features from disproportionately influencing the model. The chosen machine learning algorithm for this model is J48, a decision tree-based classifier. This choice was motivated by its demonstrated effectiveness in network packet classification, as supported by multiple studies [7, 3]. To validate the model, cross-validation with k=100 was employed. Stratified cross-validation was used to mitigate the impact of class imbalance, as it preserves the original class proportions within each validation fold, ensuring fair representation of both majority and minority classes during training and evaluation [18].

4. EXPERIMENTS AND RESULTS

4.1 Weka

The experiments were conducted using Weka, a machine learning software developed in Java at the University of Waikato in New Zealand. This data mining environment provides a comprehensive suite of visualization tools and machine learning algorithms for data analysis and predictive modeling. Additionally, its graphical user interface facilitates ease of use, making it accessible for various applications. Initially developed in 1997, Weka has since be-

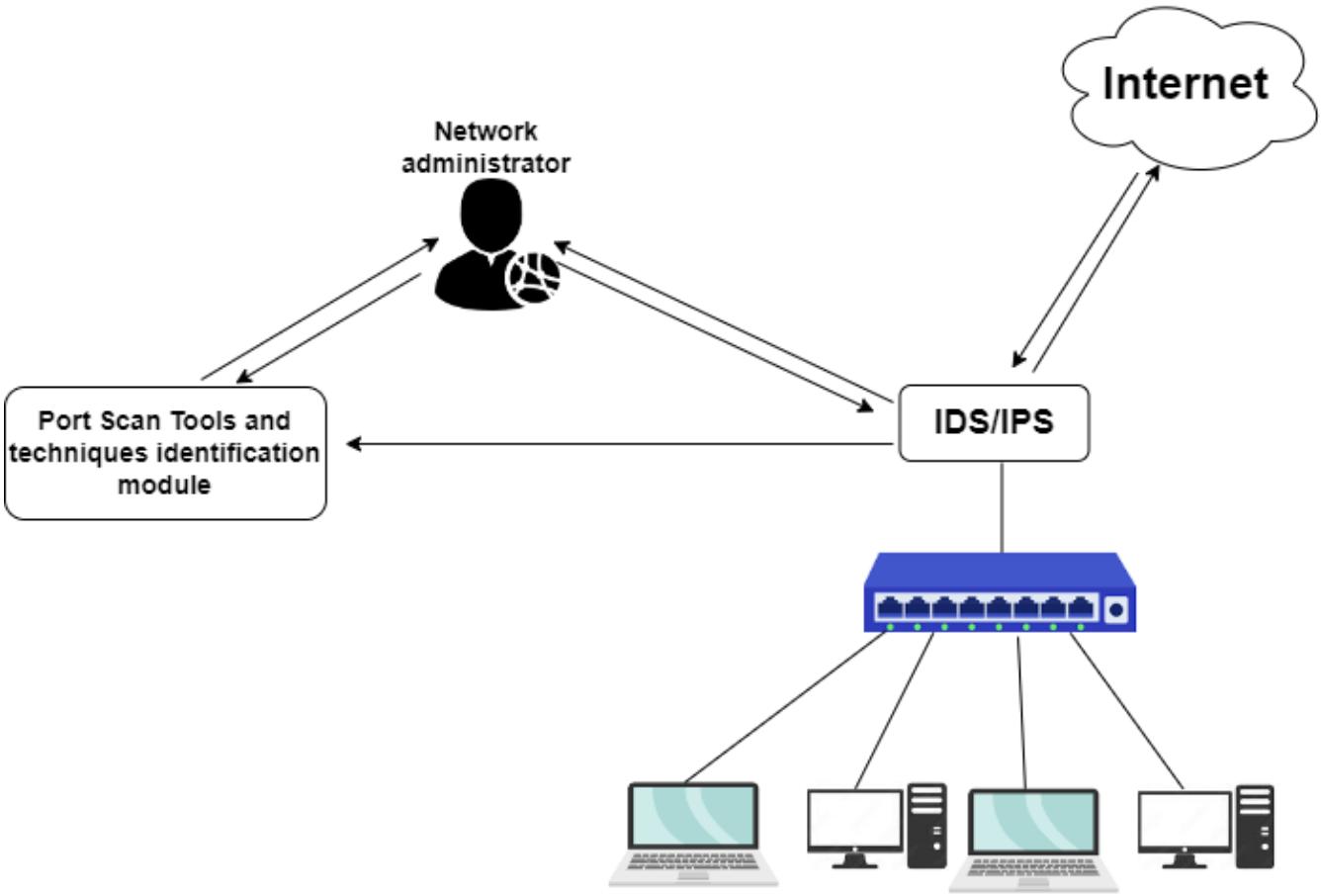

Fig. 5. Proposed system architecture

come widely adopted in multiple domains, particularly in education and research, due to its versatility and extensive functionalities.
The experimentation process followed the scheme illustrated in Fig.10. The final module in Fig.10 pertains to the experimental results, which will be detailed in the subsequent sections of this document.

4.2 Port scan tool and technique identification

4.2.1 True positive and False positive rate. In classification, a dataset can be categorized into four distinct outcomes. Given two classes, A and B, the classification results are as follows: instances of class A correctly classified as A are termed true positives (TP), while instances of class A incorrectly classified as B are called false negatives (FN). Conversely, instances of class B correctly classified as B are true negatives (TN), and those misclassified as A are false positives (FP). True positives (TP) and true negatives (TN) represent correct classifications, whereas false positives (FP) and false negatives (FN) represent misclassifications. Two commonly used evaluation metrics in classification models are the true positive rate (TPR) and the false positive rate (FPR). The TPR is calculated as the number of true positives divided by the total number of actual positives (TP + FN), while the FPR is the number of false positives divided by the total number of actual negatives (FP + TN) [25].

Figures Fig. 7, Fig. 8, and Fig. 9 illustrate the TP-rate and FP-rate curves for each scanning technique analyzed in this study.

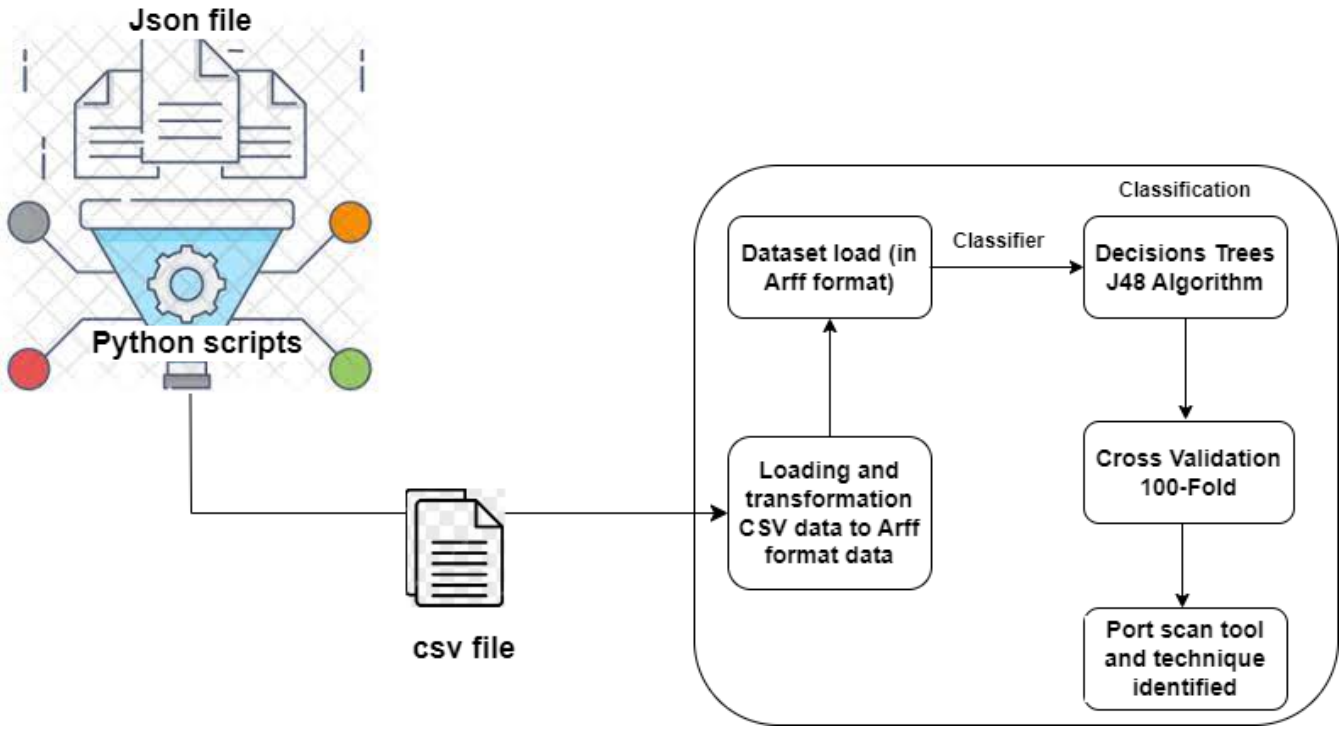

Fig. 6. Path of experimentations

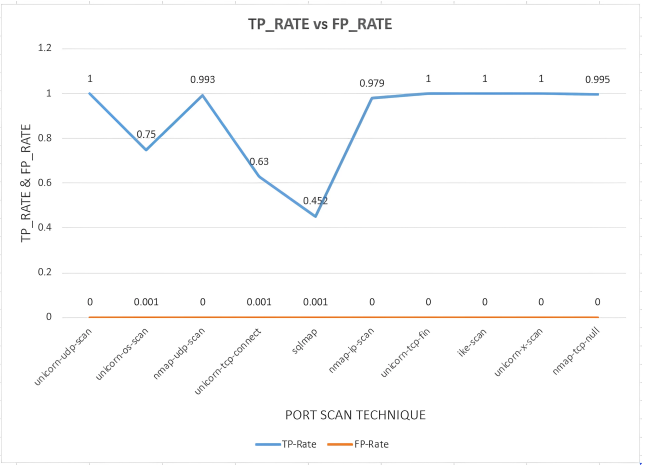

Fig. 7. The true positive rate for port scanning technique 1.

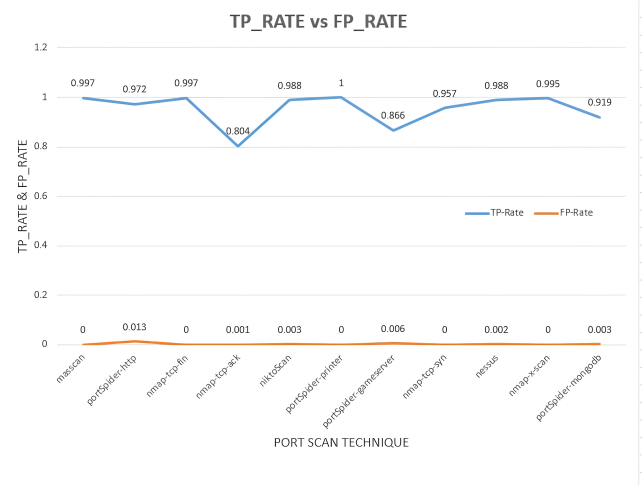

Fig. 8. The true positive rate for port scanning technique 2.

4.2.2 Precision and ROC cuve. Precision and ROC are measures of the relevance of pattern recognition models, automatic classification models, and many others. Precision represents the proportion of relevant items among all proposed items. It is calculated using the formula (1).

$$Precision = \frac{TP}{(TP + FP)}. \quad (1)$$

The curves in Fig. 10, Fig. 11, and Fig. 12 represent the precision and ROC curve of each scanning technique considered in the dataset. A ROC curve represents the TVP and TFP values for different classification thresholds. Decreasing the classification threshold results in more items being classified as positive, increasing the number of false positives and true positives. The details of the model results are shown in table. 1. The confusion matrix of our experiments is available on a GitHub repository of [2]

4.3 Targeted services identification

The target service identification module, similar to the scanning technique identification module, employs a decision tree-based machine learning algorithm. The model was trained on the second dataset using the J48 algorithm. Figure 13 presents a diagram illustrating the average values of various evaluation metrics for the model, while Table 2 provides a detailed breakdown of the classification results.

After training and cross-validation, the model's hyperparameters are exported and tested in a Python script for identifying scanning tools and target services in the previously mentioned GNS3 network. In each scan, the scanning tool used was correctly identified.

Port scan Techniques	True positive Rate	False positive Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
niktoScan	0.988	0.003	0.984	0.988	0.986	0.984	1.000	0.998
nessus	0.988	0.002	0.979	0.988	0.983	0.982	0.999	0.995
portSpider-gameserve	0.866	0.006	0.908	0.866	0.887	0.880	0.991	0.940
nmap-tcp-null	0.995	0.000	0.992	0.995	0.994	0.994	0.999	0.997
portSpider-ssh	0.854	0.003	0.843	0.854	0.848	0.845	0.990	0.892
masscan	0.997	0.000	0.999	0.997	0.998	0.998	1.000	1.000
portSpider-http	0.972	0.013	0.939	0.972	0.955	0.946	0.996	0.982
portSpider-mongodb	0.919	0.003	0.883	0.919	0.900	0.898	0.994	0.956
nmap-tcp-ack	0.804	0.001	0.945	0.804	0.869	0.870	0.998	0.954
sqlmap	0.452	0.001	0.644	0.452	0.531	0.538	0.995	0.617
nmap-udp-scan	0.993	0.000	0.992	0.993	0.993	0.993	0.998	0.995
nmap-tcp-fin	0.997	0.000	0.999	0.997	0.998	0.998	0.999	0.998
nmap-x-scan	0.995	0.000	0.998	0.995	0.997	0.996	0.998	0.996
angry-ip	0.863	0.000	0.963	0.863	0.910	0.910	0.979	0.881
nmap-v-scan	0.994	0.000	0.971	0.994	0.982	0.982	0.998	0.983
nmap-ip-scan	0.979	0.000	0.992	0.979	0.986	0.986	0.994	0.967
portSpider-printer	1.000	0.000	0.996	1.000	0.998	0.998	1.000	0.998
nmap-w-scan	0.954	0.003	0.830	0.954	0.887	0.888	0.998	0.933
nmap-scan-flag	0.994	0.000	0.997	0.994	0.995	0.995	0.998	0.997
nmap-tcp-syn	0.957	0.000	0.991	0.957	0.974	0.973	0.993	0.974
nmap-tcp-connect	0.848	0.000	0.972	0.848	0.906	0.906	0.990	0.921
nmap-m-scan	0.997	0.000	0.998	0.997	0.997	0.997	0.998	0.996
portSpider-mysql	1.000	0.000	0.999	1.000	1.000	1.000	1.000	0.998
unicorn-tcp-ack	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
unicorn-os-scan	0.750	0.001	0.672	0.750	0.709	0.709	0.999	0.84
unicorn-x-scan	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
unicorn-tcp-fin	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
ike-scan	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
unicorn-tcp-connect	0.630	0.001	0.716	0.630	0.670	0.671	0.999	0.848
unicorn-tcp-null	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
unicorn-udp-scan	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Weighted Average	0.966	0.003	0.966	0.966	0.966	0.963	0.997	0.982

Table 1.
Detailed accuracy for different scanning techniques

Service	True positive Rate	False positive Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
m-SQL	0.997	0.001	0.992	0.997	0.995	0.994	1.000	1.000
Auth	0.990	0.005	0.979	0.990	0.984	0.981	1.000	1.000
DHCP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Netbios	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
LDAP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
MySQL	0.972	0.001	0.992	0.972	0.982	0.980	1.000	0.999
FTP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
QMTP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
IPP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
ssh	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Shell	0.868	0.003	0.833	0.868	0.850	0.847	0.999	0.948
DNS	0.997	0.000	0.999	0.997	0.998	0.998	1.000	1.000
HTTP	0.545	0.002	0.507	0.545	0.525	0.524	0.998	0.697
login	0.545	0.001	0.733	0.545	0.625	0.631	0.999	0.769
IMAP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
PostgreSQL	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
POP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Exec	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Printer	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000
SMTP	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 2.
Detailed accuracy for different services

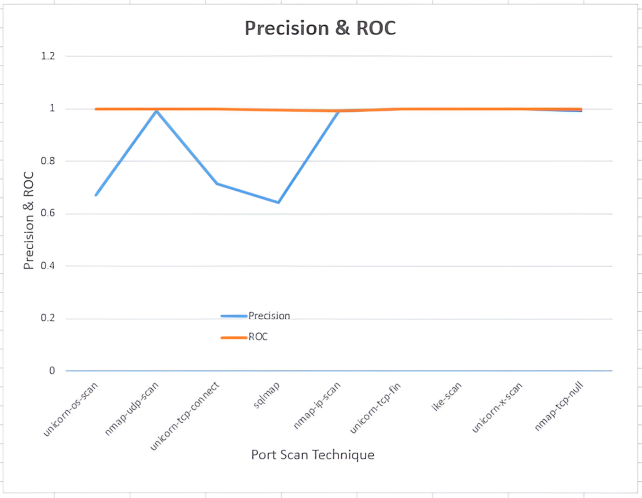

Fig. 10. Precision and ROC curve for each port scan technique

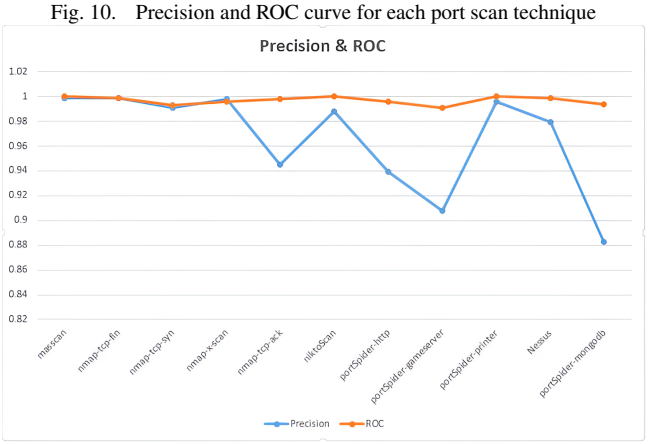

Fig. 11. Precision and ROC curve for each port scan technique

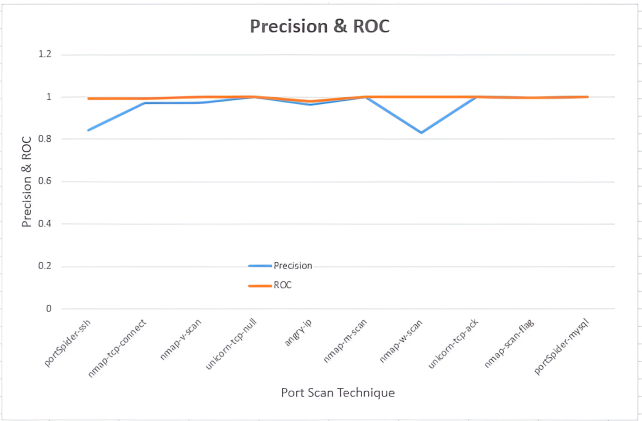

Fig. 12. Precision and ROC curve for each port scan technique

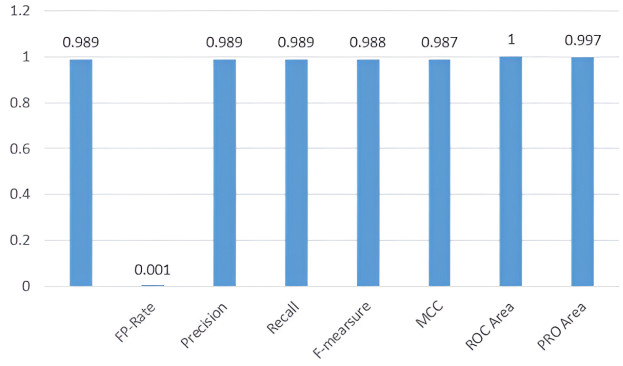

Fig. 13. Diagram of the mean values of the different evaluation criteria

The dataset we designed for this work, as well as the scripts for cleaning the raw data, are available on our GitHub repository [1].

5. CONCLUSION

This article presents an approach based on supervised learning to identify port scanning techniques, the tools used, and the targeted services during a network attack. The data was collected from a real network to build the dataset, leveraging the essential properties of the extracted packets. The experiments were carried out on Weka using the J48 decision algorithm, which proved to be particularly suitable for identifying scanning techniques with good accuracy (96.8%) and targeted services with an accuracy of 98%. This method is especially effective for gathering information on attacks involving tools specialized in well-defined actions, such as PortSpider or SqlMap, but it shows limitations when dealing with more generic blind scans.

Several improvement perspectives can be considered, such as integrating other advanced learning algorithms, including neural networks or deep learning models, to enhance accuracy and the model's generalization capability. Expanding theSeveral improvement perspectives can be considered, such as integrating other advanced learning algorithms, including neural networks or deep learning models, to enhance accuracy and the model's generalization capability. Expanding the dataset by incorporating more diverse scenarios and recent tools would help better cover the evolution of attack techniques. Generalizing the analyses to collect information on attacks involving versatile scanning tools like Nmap would also be beneficial. dataset by incorporating more diverse scenarios and recent tools would help better cover the evolution of attack techniques. Generalizing the analyses to collect information on attacks involving versatile scanning tools like Nmap would also be beneficial.

References

[1] Datasets of different scanning tools and techniques. `https://github.com/stephane65133/Data`.

[2] Result confusion matrix. `https://github.com/stephane65133/Data/blob/main/confusion%20matrix.PNG`. Accessed: 2023-03-26.

[3] Kamarularifin Abd Jalil, Muhammad Hilmi Kamarudin, and Mohamad Noorman Masrek. Comparison of machine learning algorithms performance in detecting network intrusion.

In *2010 International Conference on Networking and information technology*, pages 221–226. IEEE, 2010.

[4] Abdulghani Ali Ahmed. Investigation approach for network attack intention recognition. In *Digital Forensics and Forensic Investigations: Breakthroughs in Research and Practice*, pages 185–208. IGI Global, 2020.

[5] Abdulghani Ali Ahmed and Noorul Ahlami Kamarul Zaman. Attack intention recognition: A review. *Int. J. Netw. Secur.*, 19(2):244–250, 2017.

[6] Mohammad Almseidin, Mouhammd Al-Kasassbeh, and Szilveszter Kovacs. Detecting slow port scan using fuzzy rule interpolation. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pages 1–6. IEEE, 2019.

[7] Ch Ambedkar and V Kishore Babu. Detection of probe attacks using machine learning techniques. *International Journal of Research Studies in Computer Science and Engineering*, 2(3):25–29, 2015.

[8] Liu Y. Li S. Gao X. Chen, B. Attack intent analysis method based on attack path graph. In *In Proceedings of the 2019 the 9th International Conference on Communication and Network Security*, pages 97–102, 2018.

[9] U. Franke and M. Anderson. Using cyber deception to enhance early detection of threats. *Computer Security Journal*, 5(1):72–84, 2016.

[10] Jayant Gadge and Anish Anand Patil. Port scan detection. In *2008 16th IEEE international conference on networks*, pages 1–6. IEEE, 2008.

[11] D. Huth and L. Brown. Honeypot systems and their role in cybersecurity intelligence. *International Journal of Information Security*, 16(5):355–367, 2018.

[12] Cynthia Bailey Lee, Chris Roedel, and Elena Silenok. Detection and characterization of port scan attacks. *Univeristy of California, Department of Computer Science and Engineering*, 2003.

[13] Han Liu, Dezhi Han, and Dun Li. Behavior analysis and blockchain based trust management in vanets. *Journal of Parallel and Distributed Computing*, 151:61–69, 2021.

[14] Talha Ongun, Oliver Spohngellert, Benjamin Miller, Simona Boboila, Alina Oprea, Tina Eliassi-Rad, Jason Hiser, Alastair Nottingham, Jack Davidson, and Malathi Veeraraghavan. Portfiler: Port-level network profiling for self-propagating malware detection. In *2021 IEEE Conference on Communications and Network Security (CNS)*, pages 182–190. IEEE, 2021.

[15] WANG Zhigang et CHEN Junhua PENG, Wu. Research on attack intention recognition based on graphical model. In *Fifth International Conference on Information Assurance and Security. IEEE,*, pages 360–363. IEEE, 2009.

[16] Xinzhou Qin and Wenke Lee. Attack plan recognition and prediction using causal networks. In *20th Annual Computer Security Applications Conference*, pages 370–379. IEEE, 2004.

[17] Jantan A. Rasmi, M. Attack intention analysis model for network forensics. In *In Software Engineering and Computer Systems: Second International Conference, ICSECS 2011, Kuantan, Pahang, Malaysia, June 27-29, 2011, Proceedings, Part II 2*, pages 403–411. Springer Berlin Heidelberg., 2011.

[18] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009.

[19] N. C. Rowe. Cyber deception: Enhancing the effectiveness of honeypots in network security. *Journal of Cybersecurity and Privacy*, 1(2):98–112, 2019.

[20] Farhan Sadique and Shamik Sengupta. Analysis of attacker behavior in compromised hosts during command and control. In *ICC 2021-IEEE International Conference on Communications*, pages 1–7. IEEE, 2021.

[21] J. Schneider and A. Herrmann. Early identification of system scanning activities in cybersecurity. *Cybersecurity Journal*, 3(2):102–115, 2017.

[22] S. Sengupta and A. Choudhury. Proactive cyber defense through deception: A framework for emerging threats. *Journal of Network and Computer Applications*, 45:12–22, 2019.

[23] A. B. Smith and P. Williams. Challenges in deploying cyber deception mechanisms: Avoiding common pitfalls. *Journal of Cyber Threats*, 7(3):204–216, 2017.

[24] Stout W. Luc-Watson J. Grim C. Liebrock L. Merza M. Urias, V. Technologies to enable cyber deception. In *In 2017 International Carnahan Conference on Security Technology (ICCST) IEEE*, pages 1–6, 2017.

[25] M Vidhya. Efficient classification of portscan attacks using support vector machine. In *2013 International Conference on Green High-Performance Computing (ICGHPC)*, pages 1–5. IEEE, 2013.

[26] J. Villar and R. Perez. Strategic use of deception in cyber defense. *Cyber Defense Review*, 9(4):38–50, 2015.

[27] T. Zhao and Q. Liu. Deception-based defense systems: A review of current research and applications. *IEEE Transactions on Information Forensics and Security*, 15:150–160, 2020.