International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 13 Issue 10, October-2024, Impact Factor: 8.375

Snort in Action: Tailoring Intrusion Detection to Combat Network Threats

Rathod Krunal Anilkumar[1], Kiran Dodiya[2], Khunt Akash[3], Divya Patel[4]

[1]M. Sc Cyber security, NSIT-IFSCS Jetalpur, Ahmadabad (Affiliated to National Forensic Sciences University) Gandhinagar, Gujarat, India
[2]Assistant Professor & Program Coordinator of DFIS (Cyber Security &Digital Forensics) NSIT-IFSCS (Affiliated to National Forensic Sciences University), Gandhinagar, Gujarat, India
[3]Assistant Professor & Program Coordinator of Cyber Security (Cyber Security &Digital Forensics) NSIT-IFSCS (Affiliated to National Forensic Sciences University), Gandhinagar, Gujarat, India
[4]Assistant Professor & Course Coordinator of DFIS (Cyber Security Digital Forensics) NSIT-IFSCS (Affiliated to National Forensic Sciences University), Gandhinagar, Gujarat, India

ABSTRACT

This paper presents the development and implementation of a network intrusion detection system using snort. The network intrusion detection system detects malicious activity in the enterprise network and alerts the console for immediate action against the intrusion. This paper also represents the custom rule creation of snort using Regular Expression. The primary objective of this paper is to detect and mitigate network-based threats through snorts and advanced detection techniques. It can detect the malicious payload, identify suspicious activity, and analyze the real-time traffic. The development of custom rules is included in this research paper to tailor snort capabilities and enhance performance and accuracy. These custom rules were designed to detect network-based attacks like cross-site scripting (XSS) attacks, SQL injection, command injection, unauthorized access attempts, and denial-of-service (DoS) attacks. An intrusion detection system's main role in a network is to help computer systems prepare for and deal with network attacks. Intrusion detection systems (IDS) have become a key component in ensuring the safety of systems and networks. Using snort, we can analyze the network traffic and response to real-time malicious activity and block the malicious activity using pre-defined rules. In this research paper, we have also implemented signature-based network intrusion detection using Snort and WinPcap.

Keywords: Network intrusion detection, Snort, payloads, Cross-Site Scripting, Denial of Services, WinPcap.

INTRODUCTION

Problem statement and challenges in IDS

With the faster advancements in computersand technology, the Internet has become asignificant part of daily life, bringing various network security-basedchallenges.Now, Network security has become an urgent concern due to the boost of threats that exploit vulnerabilities in the network. Tools such as firewalls and antivirus software offer a layer of protection, but sometimes, they fail to address all security risks in today's sophisticated network environments.[1]One critical solution in this landscape is the implementation of Intrusion Detection Systems (IDS), which can identify threats but need user assistance to stop them and create signatures based on network topology requirements.[2]. An intrusion detection system is a network security approach that observes and analyses network traffic without actively interfering with or alerting network flow. IDS captures real-time or nearly real-time traffic and gives alerts based on pre-defined rules. Tools like Snort work in the background and collect data from network packets, inspect the header and payload of incoming and outgoing packets, compare this data against predefined rules or signatures of well-known attacks, and give alerts based on that to detect abnormal and malicious activity.[3]When an anomaly is detected, the system generates alerts or logs this activity for an administrator to review. Network interfaces are typically settled down to promiscuous mode,which allows the system to capture and analyze all network traffic. Snort is a powerful tool that can inspect network-based and host traffic, detect malicious activities, and generate an alert.[4]. We must develop unique rulesets or signatures to detect and prevent network-based attacks like XSS, SQL injection, denial of service (DoS), and brute force attacks. Significant signatures provide a predefined patternset that represents well-known attack techniques. Various challenges arise when creating a custom ruleset

or signature, like False Positives,Maintaining Accuracy,Regex Syntax Errors, and human errors.Updating the custom rulesets is required to prevent the organizations from new attacks.[5].

Project Aim and Objectives (Attack Simulation, Custom Rules, Alerts, Analysis)

The main objective of this project is to develop and design an attack simulation model using network intrusion detection to effectively monitor, detect, and respond to network-based attacks. This project seeks to tailor the capabilities of snort by creating the custom rule set using regular expression, generating alerts using real-world attack scenarios on a hosted website, and performing live analysis to enhance the security of the organization's network and to make defenses before attacks. We can also check the accuracy and efficiency of custom rulesets created by us to detect various types of attacks: cross-site scripting (XSS) attacks, SQL injection, command injection, and denial of service (DoS) attacks, as well as remote code execution. Setup a web-based environment vulnerable to common attacks to detect various attacks. For detection, custom ruleset development comes first, so we must create signatures using regular expressions for specific network threats like cross-site scripting (XSS) injection, SQL injection, command injection, RCE (remote code execution), and denial of service (DoS) attacks. Ensure that the created snort signature can generate real-time alerts when suspicious activities or attacks are attempted. After implementing the rules, real-time traffic analysis of simulated attacks using Snort's packet inspection mode is used to monitor, log, and detect intrusions continuously**.**

Scope (Understanding IDS, Rule Development)

This research focuses on the practical implementation and enhancement of an intrusion detection system tailoring the capabilities of snort with custom rule development and detection of OWASP's top 10 network-based threats. Intrusion detection is a critical aspect of network security for identifying malicious or suspicious activity at both the host and network levels. It can be classified into two main types: signature-based and anomalies-based. Both have different methodologies for the detection of attacks. Much like antivirus signature-based IDS, it works on pre-defined signatures in databases that compare network data against databases; if they match, IDS gives an alert. An anomaly-based network intrusion detection system identifies abnormal traffic or deviation from normal traffic behavior by detecting anomalies in network traffic. It analyzes the packet header and the network protocol to find the abnormality and unusual patterns. This technique more efficiently detects unknown or zero-day attacks but generates more false positive alarms than a signature-based system**.**

NIDS (Network-based intrusion detection system)

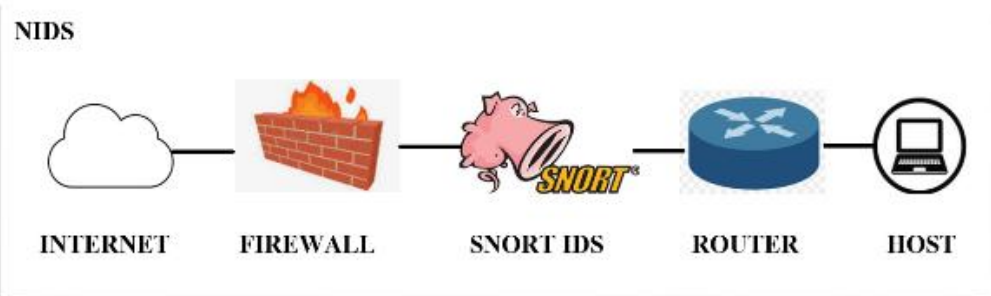

Figure 1NIDS (Network-based intrusion detection system)

This intrusion detection system captures and analyzes packets traveling over network mediums like cables and wires. It compares the traffic against the signature database to detect malicious payloads and network activities.

Advantages:

NIDS monitors traffic across the whole network, which can lead to early detection of attacks targeting the network rather than a specific host.

It is deployed strategically in the network for central monitoring and to reduce the need for individuals on every host.

Disadvantages:

Sometimes, NIDS cannot inspect encrypted traffic because it has no decryption capabilities, which limits the effectiveness of detection attacks in communication such as HTTPS.

NIDS needs high bandwidth and performance to analyze and capture the live traffic and has difficulty detecting malware activity at the host level.

HIDS (Host-based intrusion detection system)

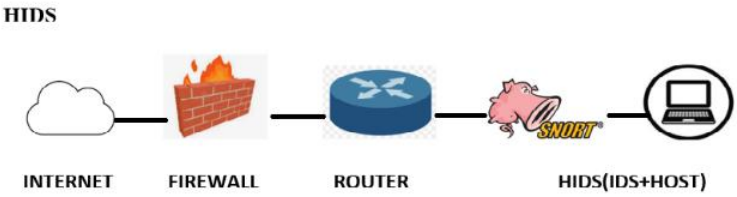

Figure 2HIDS (Host-based intrusion detection system)

Host-based intrusion detection systems (HIDS) are directly deployed at the host or endpoint level. They provide deep insight into host-level activities, including insider threats. HIDS monitors traffic directed toward hosts or endpoints and monitors application and system logs for detecting suspicious ongoing activities at the host level[6].

Advantages
It provides deep insight into the host-level activities, including file integrity, log analysis, and process monitoring, and offers deeper insight into ongoing malicious activities at the host level.

It provides valuable data for forensic analysis after an incident and detects attacks even when the network traffic is encrypted.

Disadvantages
Protecting the host consumes more CPU, memory, and disk space on individual host systems.
It is more time-consuming and resource-intensive to install and maintain at each host, especially in large organizations.

Rules Development
Creating custom rules, such as signature-based snorts, is the most critical aspect of IDS. It involves creating specific patterns (signatures) that match the attack payload under which the alert was triggered and allow Snort to detect the potential threats in real-time. While making rules, the vendors or rule maintenance department has to ensure that regulations are accurate and minimize false positives. Rule creators must maintain the latest regulations based on the new attack signature. There is a specific pattern to make rules in snort, and deep network protocol knowledge is needed to create signatures.

Related work
As an Intrusion Detection and Prevention System (IDPS), Snort has been tailored to combat various network threats through innovative methodologies and integrations. Recent studies highlight implementing machine learning techniques, such as Decision Trees and Random Forests, to enhance intrusion classification accuracy, with Decision Trees outperforming in specific scenarios[7]. Additionally, the integration of Snort with Telegram allows for real-time alerts and immediate responses, such as IP blocking and password changes, thereby improving the system's responsiveness to threats.[8]. Customized rules and the incorporation of tools like Wazuh and Splunk further enhance Snort's capabilities, enabling effective monitoring and analysis of network traffic to identify and mitigate attacks like DDoS[9][10]. Furthermore, Snort's effectiveness in detecting malware traffic has been validated through comparative analyses, showcasing its strengths in identifying known signatures and its scalability in diverse environments.[9], [11]. Collectively, these advancements underscore Snort's adaptability and effectiveness in addressing contemporary network security challenges.

REQUIREMENTS

To implement effective IDS, we need some specific hardware and software requirements.

Hardware and software requirements.
Hardware
Processor: Intel Core i3 or higher processor for processing network traffic.
RAM: 4 GB or more than 4 GB is recommendable for large environments.
Storage: 20 GB of at least free space for logs, snort installation, and related files.
NIC: A network interface card that supports promiscuous mode for capturing packets.
Software: Operating system: Windows; you can also use Linux and MacOS.

Npcap: Library for Windows to capture live traffic. It operates in the background and allows Snort to analyze packets. It is a packet capture and network analysis architecture for Windows that gives applications direct access to network data. It will enable applications to capture, filter, and transmit raw packets to the network. It is made up of a software library and network driver. It includes a kernel-level packet filter, a low-level dynamic link library, and a high-level library. It also supports loopback packet capture, which allows it to sniff transmissions between services on the same machine.

Snort: IDS software is used to monitor and analyze the network traffic. It can analyze real-time traffic analysis and data flow in a network. It can check protocol analysis and can detect different types of attacks. Snort rules can be written in any language, its structure is also good, and it can be easily read.Rules can also be modified. Snort can detect this attack in XSS attacks by matching the previous pattern. In a signature-based IDS system, if patterns match, then attacks can be easily found, but IDS fails to detect them when new attacks come.

CMD (terminal): Command-line interface for managing snort for Windows or terminal for Linux/MacOS.
Configuration file: essential for setting up rules, logs, and network interface paths.

IMPLEMENTATION

Laboratory setup

This phase of the project's primary goal is to set up the laboratory environment, install and configure tools like Snort, and demonstrate how custom IDS rules are developed and implemented to detect network threats. This phase involves installing software, configuring snort to operate effectively, and creating custom rules specific to attack vectors, such as cross-site scripting (XSS), SQL injection, and denial of service (DoS) attacks.

Software Installation and Configuration

During this implementation, we can evaluate and review Snort's working principle and assess its ability to analyze network traffic in real-time, detect malicious activities, and generate alerts. This laboratory setup involves implementing and creating custom rules tested in simulated network environments. The overall objective is to evaluate the effectiveness of IDS functionality in detecting and reporting security breaches by implementing predefined and custom rules.

Install snort

Download and install Snort from https://snort.org/ on the system. This core IDS setup is responsible for monitoring and analyzing the network traffic.

Install Npcap

Download and install the Npcap packet-capturing library. This library allows Snort to capture live packets for analysis. It works at the network layer and provides Snort with the necessary access to the data packet for inspection.

Configure Snort. conf

In snort. Conf to configure network variables and rule paths; configuration is the main part, so using this, the snort can detect and work efficiently. It will not work as expected if you mistakenly misconfigured one of them. Defining the network variable is a crucial part of IDS. Using this variable, snort will monitor the traffic. It includes two network variables, HOME_NET and EXTERNAL_NET. HOME_NET refers to the configuration of the local network segment that Snort will focus on for detection. Sometimes, it is a local range of IP addresses like 192.168.1.0/24. EXTERNAL_NET is considered an external network for monitoring. The IP ranges are usually defined, but the best practice is to determine the HOME_NET variable. This indicates that any traffic not originating from the HOME_NET should be treated as external. This distinction aids Snort in effectively differentiating between internal and external traffic. Another critical aspect of snort is configuring the rule paths for snort's rule file. Setting up this variable designates the directory of Snort's rule file. Custom rules are defined in the local. The rules file indicates that the custom rules are loaded into Snort's operation. using the command include $RULE_PATH/local. rules ensure that custom rules are defined locally. Rules files are loaded into Snort's operational environment. This functionality will adapt and enhance the capabilities of snort by adding a custom rule set for detection.

The next step is to display the list of network interfaces that Snort can utilize to capture the packets for analysis. Command Snort-W provides valuable information about each network interface, including name, description, and index number. Interface identification in IDS is critical to ensuring snort captures traffic from the correct source. For initiating traffic after identification of network interface snort, use command snort -i <interface_index>. This command is also used for diagnostics of the status of network interfaces and to ensure that they are ready for operational use. After all the identification and configuration, the command snort -i 5 -c c: Snort\etc\ snort.conf commands Snort to monitor the specific

network interface using the rules and settings defined in the snort—conf file. -I specify the network interface that snorts will listen to the incoming traffic, and 5 represents the index number of the network interface. The -c option allows Snort to specify the path to the configuration file. Make some custom rules for testing purposes; for example, if you ping to some IP, does the network interface capture or not the ping? If snort can capture, the traffic or ping configuration is made successfully.

Snort working overview

Snort is an open-source and widely used intrusion detection system that works on real-time traffic to monitor and analyzeit. It operates primarily in two modes: network sniffer mode and network intrusion detection mode.

Network sniffer mode

Snort only logs packets into a log file when it works in network sniffer mode. After storing the packets in the log file, they can be later analyzed for further analysis and operation. No intrusion detection activity is done during this mode; using Snort for this mode is not very useful, as many tools are available for packet sniffing. It only reads the packets as they move across the network and gives output in a human-friendly format. Commonly, it is used in basic traffic analysis and troubleshooting.

Network intrusion detection mode.

It is the most powerful and most used mode of snorting. It monitors traffic from various sources and analyses it against a pre-defined rule and signature. If traffic matches the signature or rule set, the console is alerted or performs pre-defined actions. This mode detects various network-based attacks like SQL injection, cross-site scripting (XSS), denial of service (DoS), and more.

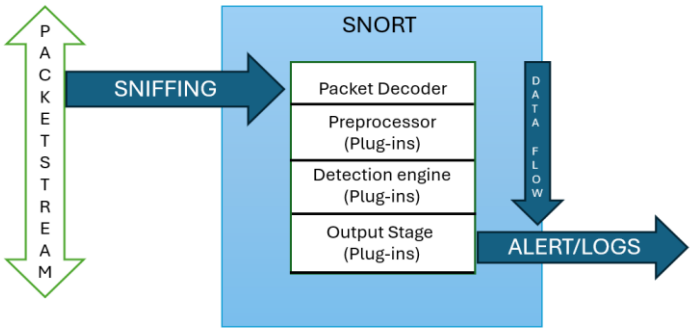

Figure 3 Network intrusion detection mode.

This image illustrates an open-source network intrusion detection system (NIDS) workflow. It shows how NIDS works, how packets are processed and filtered in multiple stages, and how it detects malicious activities. A packet from the Internet enters the packet decoder and goes through several phases. Snort takes the required action at every phase. If the detection engine finds any miscellaneous content in the packet, it will alert or log that packet. A module packet is logged in, or an alert is generated.

1. Packet Stream

Firstly, Snort works in promiscuous mode to capture packet streams via sniffing. This means it intercepts all traffic passing through a network, not just packets addressed to the system. Sniffer refers to a network capture process where Snort collects raw packets for deep analysis.

2. Packet Decoder

The packet decoder is used to understand the packet structure and fields in the coming packets after sniffing. This component breaks down each packet to examine whether it is valid. It examines the protocol headers, such as ETHERNET, IP, TCP, and UDP, and after reviewing that, it ensures whether the packet format is legitimate. It is a critical stage that relies on accurately intercepting and properly decoding the packet format.

3. Preprocessor

This component includes combining fragmented packets into complete streams for protocols like TCP, identifying open port scanning attempts, and ensuring protocols are processed accurately to avoid evasion techniques. The main

responsibility of this component is accurately normalizing the various packets and inspecting them before they reach the detection engine. It also matches the pattern of whole strings in the packet.

4. Detection Engine
This is the core of snort. That detects malicious activities against the defined rule or signature. If the packet matches the rules, the detection engine triggers an action. The detection engine evaluates the packet based on the rule. Each rule defines the patterns that might indicate the malicious activity attempt.

5. Output Stage
As the name suggests, if the packet matches the patterns, it will log or generate an alert. This stage can be customized to log or alert to a specific destination. In the output stage, snort handles the packets or gives an alert based on pattern matching.
6. Alerts/Logs

Here, the final output will come, depending on whether to give an alert or log, depending on the creation of the rule. Logs could be sent to the Security Event Information Management tool for further correlation and analysis. Alert notifications can be sent to the administrator for immediate response.

Custom Rule Creation and Examples
Creating custom rules in a snort allows you to detect the malicious payloads in packets and tailor the behavior of the IDS system to detect threats or suspicious activity in an enterprise network. Creating custom rules for the organization depends on the organization's needs and network defense policy.Rules varyfrom organization to organization; rules are not the same for all organizations. Some general rules come with a snort rule file, but you can do it if we want to add or modify some specific rules.

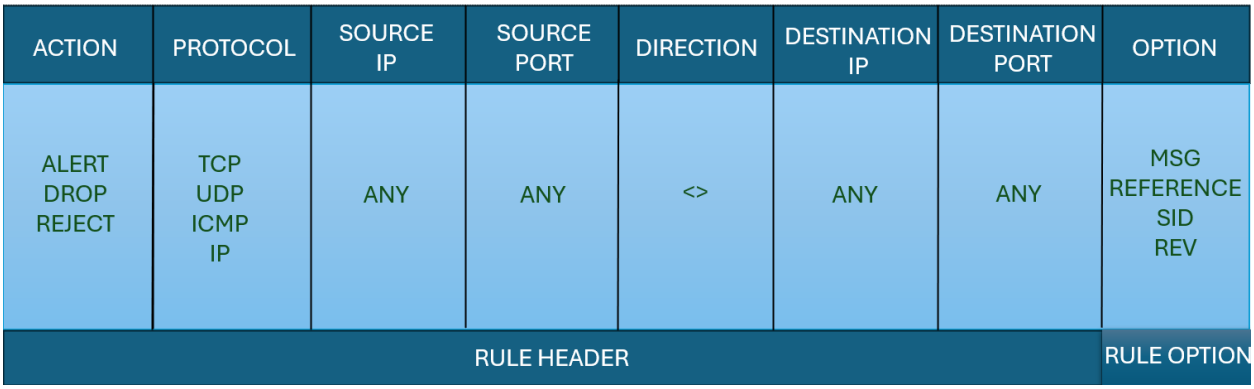

Figure 4 Custom Rule Creation and Examples

The rule header and Rule option are the two main components of the Snort rule structure. The rule header consists of seven fields and includes fundamental conditions for triggering the rules. Each field works together to fulfill the rule's structure and helps Snort tailor its ability to detect malicious activity in the organization's network.

Rule Header
1. Action
This action field specifies if the rule condition matches or specifies the action when a packet matches the rule condition defined in the custom rule. This field indicates the specific action if the pattern matches. Actions like alert, drop, and reject. Before writing the rule action, you must be aware of its functionality.

Alert: send an alert message to the console if some conditions are met or rule conditions are true for a particular packet. (Used in IDS mode).
Drop: If certain conditions are met,the packetand the eventare logged. (This is used in IPS mode.)
Reject: It is like the drop but sends an error message to the console (used in IPS mode).

2. Protocol
The protocol part applies the rule to packets for a particular protocol only. This is the first criterion mentioned in the rule. Some examples of protocols used are IP, ICMP, UDP, etc.

IP: Snort will examine the Ethernet header in this protocol to determine the packet type. This allows Snort to detect the IP without the transport layer protocol like TCP or UDP.
ICMP: This protocol is used for network diagnostics,such as a ping request. Snort uses the IP header to identify ICMP packets.
TCP: Snort can inspect specific headers such as source/destination IP addresses, ports, flags, and sequence numbers.
UDP: Snort looks at the header fields,such as IP addresses and ports in the UDP packet.

3. Source Ip
Define the source IP address where the rules apply. Here, we define the specific IP range and use ANY that defines the rule to apply to any source IP address.

4. Source Port
This field specifies the source port, where we can write a single port number or a range of the port number. We can use ANY that indicates that any source port can trigger the rule. It is useful when we want to monitor specific services.

5. Direction
The direction field determines the source, destination addresses, and port numbers in a rule. The following rules apply to the direction field.
-> (right arrow) indicates traffic moving from source to destination.
<- (Left arrow) indicates that traffic is moving from destination to source.<> (both ways arrow) indicates that the rule applies to traffic in both directions. This symbol is useful when you want to monitor data packets for both client and server.

6. Destination Ip
Indicate the destination IP address where the rule applies, like the source IP.

7. Destination Port
Indicate the destination port number where the rule applies, like the source port. We can use ANY to indicate that any destination port can trigger the rule. It is useful when we want to monitor specific services.

8. Rule Option
1. MSG: This option indicates the message on the console when the alert is triggered. This message helps us quickly identify which attack type has been attempted.

2. REFERENCE: This option is used when the creator wants to link to external sources that provide more information about the vulnerability or attack, such as CVE-2021-44228.
3. SID: This option helps to track and manage the rule. It is called a snort ID, a unique identifier assigned to each rule.
4. REV: This option specifies a revision number for a certain rule. As rules are updated or refined, the revision number helps track changes.

9. Example Of Rule
Here, we take an example of a rule when someone tries to ping our computer: how it is identified as a source and destination, and the focus is on making rules. alert ICMP any <> any any (msg:"ICMP Packet found"; reference: CVE-XXXX; sid:1000001; rev:1)
ACTION: Snort will generate an alert when the packet matches the rule.
PROTOCOL: This rule is focused on the ICMP protocol.
SOURCE IP AND SOURCE PORT (ANY-ANY): This rule applies to the packet from any source IP address to any source port.

DIRECTION: The bidirectional symbol shows that the rule applies to any direction, whether an ICMP request or an ICMP response.
Destination IP AND DESTINATION PORT (ANY ANY): The rule applies to any destination IP address and port.
MSG: ICMP PACKET FOUND": This specific message will be displayedwhen this particularrule triggers an alert.
Reference: CVE-XXXX: This specifies the specific CVE related to ICMP vulnerabilities.
SID:1000001: This indicates the specific unique snort identifier.
REV:1: The rev indicates the rule revision number. This helps track changes to the rule over time.

Rule explanation and demonstration.

Rule for XSS detection and demonstration

XSS is a cyberattack in which a malicious actor injects malicious code into a trusted website and executes it when the user visits it. XSS attacks allow attackers to execute arbitrary JavaScript on other users' browsers, leading to session hijacking, data theft, or other malicious actions.

alert tcp any any -> any any (msg: "Possible script tag in HTTP request"; content:" <script"; nocase; pcre:"/(?i)(<|%3C)\s*script/"; sid:1000005; rev:1;)

This rule tries to catch the <script> tag in an HTTP request. This rule triggers an alert when malicious JavaScript code with the <script> tag attempts to inject into a website. This rule contains both simple content <script> and PCRE regular expression to ensure that this rule can catch different forms of tags,Including variations with different capitalization or encoded characters (like %3C for <). This rule helps to identify suspicious activity that could involve client-side code injection, which is a common attack vector for XSS attacks.

SQL injection

An SQL injection attack is a code injection technique that allows an attacker to insert malicious SQL statements into an entry field for execution. These attacks can be used to attack any SQL database but are most commonly used against websites.

alert tcp any -> any (msg: "SQL Injection Attempt Detected"; content: "SELECT"; nocase; pcre:"/(?i)\bSELECT\b\s+(?:[^'"\\]+|'[^']*'|"[^"]*"|(?:--[^'\r\n]*)? (?:#|\/*|%[a-fA-F0 9]{2}|$)*?\bFROM\b/"; sid:1000001; rev:1;)

This snort rule is designed to detect SQL injection attempts. It finds SQL queries in HTTP traffic. Particular focus should be placed on words like SELECT query followed by FROM, which indicates a common pattern in SQL injection attacks. It also looks for patterns indicating malicious input in web traffic, such as quotes, comments, and encoded characters.

Brute force attack

A brute force attack is a cyberattack where a hacker uses trial and error to guess a user's login credentials. It is a simple yet reliable tactic for gaining unauthorized access to individual accounts and organizations' systems and networks.

Alert TCP any -> any 22 (msg: "Potential SSH Brute Force Attack"; flow:to_server, established; content: "Failed password"; nocase; detection_filter: track by_src, count 5, seconds 60; sid:1000001; rev:1)

This rule is specific for detecting a potential SSH brute force attack by monitoring failed password attempts on SSH port 22. This rule looks for the particular content “failed password” in the packet, which is common in unsuccessful SSH login attempt logs. An alert is triggered if five or more failed attempts occur from the same source within 60 seconds.

Command injection

Command injection is an attack that aims to execute arbitrary commands on the host operating system via a vulnerable application. In Command Injection, the attacker extends the application's default functionality, which performs system commands without injecting code.

alert tcp any -> any 80 (msg:"Command Injection detected"; content:"cat"; pcre:"/(?i)(cat(\s*[\w-]+)*$)/"; sid:1000004; rev:2;)

This snort rule detects a possible command injection where an attacker tries to use the cat command over an HTTP request on port number 80. (?i) makes the match case-insensitive and cat(\s*[\w-]+)*$)/"; Matches the cat command. It checks if the string “cat” is followed by a potential file name or arguments and triggers an alert based on that rule.

Denial of Service (DoS)

A DoS attack is a cyberattack that makes a network resource or machine temporarily or permanently unavailable to its intended users. The attacker does this by flooding the target with unnecessary requests to overload the system and prevent legitimate requests from being fulfilled. alerttcp any -> any (msg:"Potential DoS attack detected"; flags:S; threshold:type both, track by_dst, count 10, seconds 10; sid:1000026; rev:1)This rule is designed for the detection of the denial-of-service attack by monitoring for repeated SYN packets. An alert is generated if a single destination receives ten or more SYN packets within 10 seconds. S indicatesthat the TCP SYN packet initiates a TCP connection in these rule flags.

Threshold: type both monitors the rate and total count of matching packets and track by_dst tracks the packets based on the destination IP.Count 10, seconds 10: The rule triggers if 10 SYN packets are seen going to the same destination within 10 seconds.

CONCLUSION AND FUTURE WORK

In enterprise networks, security is a big issue in today's environment. Hackers and intruders have found many successful techniques to disrupt the enterprise network using successful attempts. Sometimes, they only target the big company network and infiltrate the data. They have powerful tools and techniques; they can do real-time traffic analysis using these. So, to detect this type of attack, we need an intrusion detection system. Once snorts identify any intrusion, they alert the console for immediate action on intrusions. Snort is a strong intrusion detection system, but sometimes it gives a configuration fault in the Windows environment. In this paper, signature-based network intrusion detection system snort has been implemented in the Windows environment. The result shows that it can be configured in a Windows environment and configured as a firewall. Future work will develop more accurate signatures and parallel techniques. Using these techniques, we can discover the intrusion attempts and improve the performance of a signature-based network intrusion detection system. Implement the AI and ML algorithms to discover intrusions and gaps in detection techniques. Using AI and ML, more accurate defensive measures can be made in the enterprise network. Integration of Snort with AI-ML will give a more precise result. Use algorithms like random forest, K-Nearest Neighbors, and SVM (support vector machine) for precise detection.

REFERENCES

[1]. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," Electronics 2023, Vol. 12, Page 1333, vol. 12, no. 6, p. 1333, Mar. 2023, doi: 10.3390/ELECTRONICS12061333.

[2]. M. Macas, C. Wu, and W. Fuertes, "A survey on deep learning for cybersecurity: Progress, challenges, and opportunities," Computer Networks, vol. 212, Jul. 2022, doi: 10.1016/j.comnet.2022.109032.

[3]. M. A. Hossain and M. S. Islam, "Ensuring network security with a robust intrusion detection system using ensemble-based machine learning," Array, vol. 19, p. 100306, Sep. 2023, doi: 10.1016/J.ARRAY.2023.100306.

[4]. "What is an Intrusion Detection System? - Palo Alto Networks." Accessed: Oct. 28, 2024. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids

[5]. M. Mazzini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," Journal of King Saud University - Computer and Information Sciences, vol. 31, no. 4, pp. 541–553, Oct. 2019, doi: 10.1016/J.JKSUCI.2018.03.011.

[6]. "What is HIDS (Host-Based Intrusion Detection System)? | Sysdig." Accessed: Oct. 28, 2024. [Online]. Available: https://sysdig.com/learn-cloud-native/what-is-hids/

[7]. R. E. Febrita, L. Hakim, and A. P. Utomo, "The Implementation of Machine Learning for Optimizing Network-Based Intrusion Detection in the Snort Application," 6th International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2023 - Proceeding, pp. 141–147, Dec. 2023, doi: 10.1109/ISRITI60336.2023.10467566.

[8]. J. A. Dharma and Rino, "Network Attack Detection Using Intrusion Detection System Utilizing Snort Based on Telegram," bit-Tech, vol. 6, no. 2, pp. 118–126, Dec. 2023, doi: 10.32877/BT.V6I2.943.

[9]. "Intrusion Detection System Using Customized Rules for Snort," International Journal of Managing Information Technology, vol. 15, no. 3, pp. 01–14, Aug. 2023, doi: 10.5121/IJMIT.2023.15301.

[10]. A. R. AL Waili, "Analysis of Traffic Using the Snort Tool for the Detection of Malware Traffic," International Journal of Information Technology and Computer Engineering, no. 33, pp. 30–37, May 2023, doi: 10.55529/IJITC.33.30.37.

[11]. S. Abdulrezzak and F. A. Sabir, "Enhancing Intrusion Prevention in Snort System," Proceedings - International Conference on Developments in eSystems Engineering, DeSE, vol. 2023-January, pp. 88–93, Jan. 2023, doi: 10.1109/DESE58274.2023.10099757.