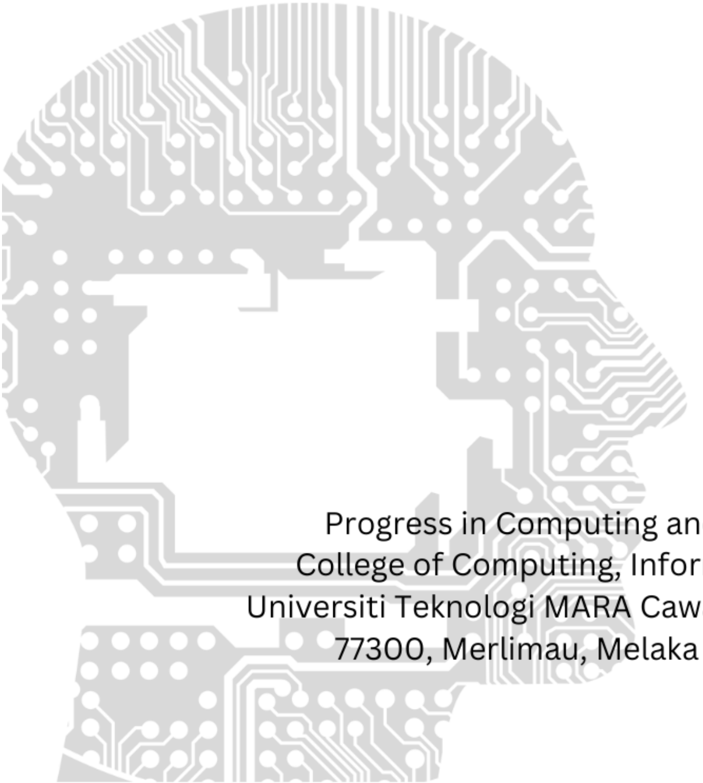
اونيۏرسيتي تيكنولوڬي مارا
UNIVERSITI
TEKNOLOGI
MARA
Cawangan Melaka

PCMJ

Progress in Computing and Mathematics Journal

volume 1

https://fskmjebat.uitm.edu.my/pcmj/



Progress in Computing and Mathematics Journal
College of Computing, Informatics, and Mathematics
Universiti Teknologi MARA Cawangan Melaka, Kampus Jasin
77300, Merlimau, Melaka Bandaraya Bersejarah

PCMJ

Progress in Computing and Mathematics Journal

volume 1

Progress in Computing and Mathematics Journal (PCMJ)
College of Computing, Informatics, and Mathematics
Universiti Teknologi MARA Cawangan Melaka, Kampus Jasin
77300, Merlimau, Melaka Bandaraya Bersejarah

EDITORS

PCMJ
Progress in Computing and Mathematics Journal
volume 1

PREFACE

Welcome to the inaugural volume of the **Progress in Computing and Mathematics Journal (PCMJ)**, a publication proudly presented by the College of Computing, Informatics, and Mathematics at UiTM Cawangan Melaka.

This journal represents a significant step in our commitment to fostering a vibrant research culture, initially providing a crucial platform for our undergraduate students to showcase their intellectual curiosity, dedication to scholarly pursuit, and potential to contribute to the broader academic discourse in the fields of computing and mathematics. However, we envision PCMJ evolving into a beacon for researchers both nationally and internationally. We aspire to cultivate a space where groundbreaking research and innovative ideas converge, fostering collaboration and intellectual exchange among established scholars and emerging talents alike.

The manuscripts featured in this first volume, predominantly authored by our undergraduate students, are a testament to the hard work and dedication of these budding researchers, as well as the guidance and support provided by their faculty mentors. They cover a diverse range of topics, reflecting the breadth and depth of research interests within our college, and set the stage for the high-quality scholarship we aim to attract in future volumes.

As editors, we are honored to have played a role in bringing this journal to fruition. We extend our sincere gratitude to all the authors, reviewers, and members of the editorial board for their invaluable contributions. We also acknowledge the unwavering support of the college administration in making this initiative possible.

We hope that PCMJ will inspire future generations of students and researchers to embrace research and innovation, to push the boundaries of knowledge, and to make their mark on the world of computing and mathematics.

Editors
Progress in Computing and Mathematics Journal (PCMJ)
College of Computing, Informatics, and Mathematics
UiTM Cawangan Melaka

TABLE OF CONTENTS

PCMJ

Progress in Computer and Mathematics Journal (PCMJ)
volume 1 [October, 2024]
e-ISSN: 3030-6728
Website: fskmjebat.uitm.edu.my/pcmj

ENHANCING COMMUNITY SQL INJECTION RULE IN INTRUSION DETECTION SYSTEM USING SNORT WITH EMAIL NOTIFICATIONS

Nur Athirah Binti Noor Mohamad
Collage of Computing Informatics and Mathematics
UiTM Melaka, Campus Jasin, Melaka
athrhmohd@gmail.com
**Corresponding author*

Noor Ashitah Binti Abu Othman
Collage of Computing, Informatics and Mathematics
UiTM Melaka, Campus Jasin, Melaka
noor2106@uitm.edu.my

Mohd Hafifi Bin Mohd Supir
Collage of Computing, Informatics and Mathematics
UiTM Melaka, Campus Jasin, Melaka
hafifisupir@uitm.edu.my

Article Info

Abstract

This project focuses on enhancing the precision and recall rates of community-based intrusion detection systems, specifically targeting SQL injection attacks within the context of Snort. The study involves the integration of modified rules employing PCRE (Perl Compatible Regular Expressions) and fast pattern matching to improve the accuracy and performance of the intrusion detection system. Experimental results demonstrate a notable reduction in false positives and a perfect recall rate, showcasing the efficacy of the enhanced rules. The virtualized testing environment, comprising a Snort-protected server, a simulated attacker using Kali Linux and Metasploitable 2, and a vulnerable system facilitates a thorough evaluation of Snort's response to cyber threats. While acknowledging limitations and the controlled nature of the testing, this research emphasizes the importance of leveraging advanced technologies to fortify intrusion detection systems against evolving cybersecurity challenges. The incorporation of PCRE and fast pattern matching stands as a significant contribution to improving rule matching accuracy and overall system efficiency in the dynamic landscape of cybersecurity.

Keywords: Snort, SQL injection, PCRE, fast pattern, precision rate, recall rate.

Received: February 2024
Accepted: August 2024
Available Online: October 2024

INTRODUCTION

In the current era of advanced technology, the escalating sophistication of cyber-attacks poses a growing challenge in effectively identifying breaches, risking the integrity of security administrations, data privacy, and organizational accessibility. The 2017 Symantec Internet Security Threat Report revealed a surge in zero-day assaults, emphasizing the heightened ambition of the new generation of malware targeting organizations directly. The consequences include compromised sensitive data, disrupted or destroyed frameworks, financial losses, reputational damage, and legal consequences. Recognizing the global impact of cyber threats, there is a pressing need for intrusion detection systems (IDS) to evolve continually to safeguard digital assets. SQL injection attacks, a common exploitation method, pose a significant threat to data confidentiality and integrity, necessitating the constant updating of IDS like Snort. The presented project focused on enhancing Snort's capabilities in a virtualized setting by incorporating features like fast_pattern and PCRE, specifically fortifying the SQL injection rule against evolving techniques. The integration of a push email notification system aimed to enhance incident response capabilities, enabling real-time responsiveness and remote monitoring to meet the demands of proactive threat mitigation in the face of ever-evolving cyber threats.

LITERATURE REVIEW

An Intrusion Detection System (IDS) functioned as a digital security guard for computer networks. Its primary role was to consistently watch and analyze the flow of data within a network, identifying any unusual or suspicious activities that could indicate a cyber-attack. It operated like a vigilant electronic sentry, monitoring for signs of unauthorized access, malicious software, or other cyber threats (CheckPoint, n.2023.). When the IDS detected something out of the ordinary, it raised an alert, enabling administrators to investigate and take action to protect the network from potential harm. An Intrusion Detection System (IDS) is a tool capable of distinguishing malicious network activity from normal and legitimate traffic on an individual computer or across a network. It detects attacks by analyzing network traffic and can respond by generating alarms. On the other hand, an Intrusion Prevention System (IPS) not only identifies attacks but also takes action by creating alarms and actively preventing the

Detection techniques in an Intrusion Detection System (IDS) are like special skills that help it identify and catch cyber threats. Just like a detective uses different methods to solve a case, an IDS uses various techniques to spot unusual or harmful activities in a computer network. These techniques include looking for specific patterns or behaviors that might indicate an attack, checking if someone is trying to break into the system, or even watching out for signs of malicious software (Villanueva, 2022). In a host-based system, the IDS examines the activity of each individual host in the system while, network-based intrusion detection systems are dedicated software systems that sit on a network wire and analyze the individual packets flowing through a network (Importance of Intrusion Detection System (IDS), 2010). Misuse detection, also known as signature-based detection, relies on predefined attack signatures to match observed activities against known patterns, effectively identifying recognized threats (Depren et al., 2005). While misuse detection excels in recognizing known threats, anomaly detection offers proactive defense against emerging risks.

Snort is an open-source Intrusion Detection System (IDS) software widely used for network traffic analysis and security monitoring (Kumar & Prakash Sangwan, 2012). Key features of Snort include its ability to perform real-time traffic analysis, detection of various types of network attacks, and flexibility in customization through rule creation. Snort, being an open-source intrusion detection system (IDS) relies on a collaborative approach for rule development, with community-contributed rules playing a crucial role in enhancing its detection capabilities. Community-contributed SQL injection rules in Snort demonstrate notable strengths in their ability to capture prevalent and commonly observed patterns associated with SQL injection attacks. The collaborative nature of rule creation allows for a collective understanding of typical SQL injection techniques, ensuring a broad coverage of known attack vectors. Despite their strengths, community SQL injection rules face challenges when dealing with sophisticated and nuanced attack techniques. Advanced SQL injection attempts often involve obfuscation, evasion techniques, or variations that deviate from typical patterns, making them harder to detect with generic rules.

PCRE, or Perl Compatible Regular Expressions, is a powerful pattern matching library used to implement regular expression pattern matching. In the context of Snort, PCRE is employed to define complex and flexible patterns for matching strings within network traffic (Otw, 2022). In addition, fast pattern matching is a mechanism used to optimize the performance of Snort's rule-matching engine. Traditional pattern matching involves scanning through the entire payload of a packet to identify specific content. This "fast pattern" serves as an initial filter, allowing Snort to quickly discard packets that do not contain the specified content (Kirk, 2010). The use of fast pattern matching significantly improves the efficiency of Snort's rule-matching process, enabling faster and more scalable intrusion detection without compromising accuracy.

SQL injection represents a security vulnerability in web applications, occurring when attackers insert malicious SQL code into user-input fields or parameters. The danger arises when these inputs are not adequately validated or sanitized by the application. In the absence of proper handling, an attacker can manipulate SQL queries, leading to the execution of unintended database operations. This manipulation can have severe consequences, including unauthorized access, data manipulation, or even deletion of records, jeopardizing the confidentiality and integrity of the database (PortSwigger, 2019).

SMTP is an application layer protocol where when sending an email, the client connects to the SMTP server via TCP and sends the message through it (GeeksForGeeks, 2019). The SMTP server is always in the listening mode. The SMTP process opens a connection through port 25 as soon as it starts listening for a TCP connection from any client. Upon establishing a successful TCP connection, the client process immediately sends the message. Thus, integration of Elastic email server for notifications involves a comprehensive exploration of incorporating Elastic Email servers into the broader context of intrusion detection system (IDS) notifications to elevate incident response capabilities.

METHODOLOGY

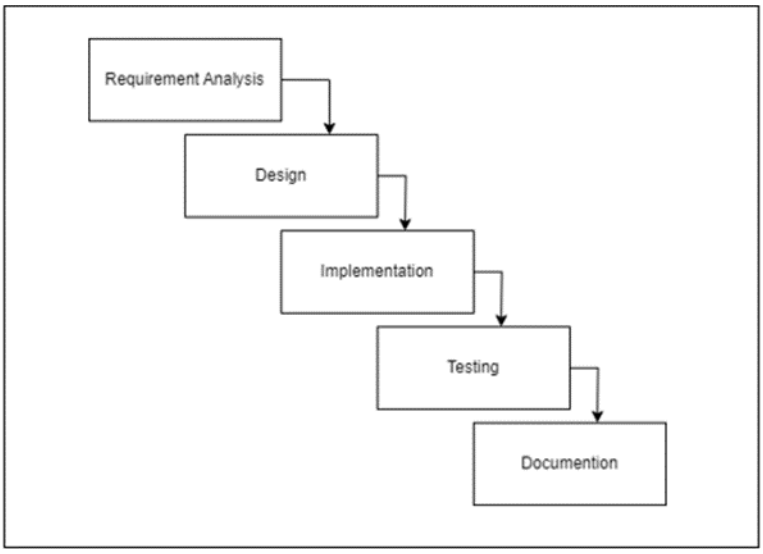


Figure 1: Waterfall Model

Figure 1 showed summary of details regarding the phases of the system based on the waterfall model. In the Waterfall model of the Software Development Life Cycle (SDLC), the process unfolded sequentially through distinct phases. The initial phase, requirement analysis, involved a feasibility study to identify problems, research objectives. The subsequent design phase outlined the integration plan for Snort with enhanced SQL injection rules and the implementation of push email notifications, documented through flowcharts and diagrams. Implementation translated design into an operative IDS system, followed by the testing phase, rigorously evaluating its effectiveness, reliability, and security. Documentation concluded the process, providing a detailed record of requirements, design, implementation, and testing methodologies. This linear approach ensured each phase's completion before proceeding, emphasizing a systematic progression through the SDLC.

A feasibility study was conducted to comprehensively evaluate the practicality and viability of the proposed project. In the context of the project "Enhancing Community SQL Injection Rule in Intrusion Detection System Using Snort with Email Notifications" involved analyzing the existing processes, identifying problems, and determining if the envisioned solution aligned with the project's objectives. The study considered factors such as technological requirements, cost implications, and potential benefits, providing a foundation for informed decision-making before proceeding with the development.

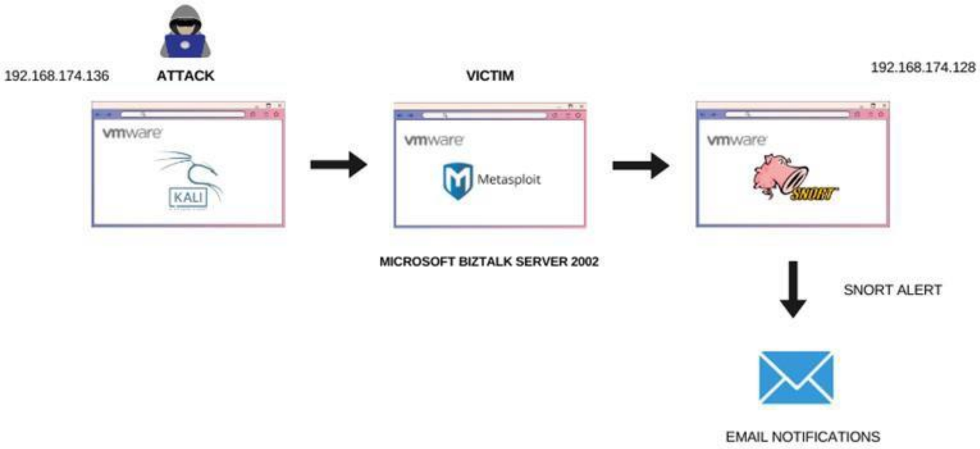


Figure 2: Logical Design

In this simulated environment, various VMware instances are strategically employed to replicate a controlled testing scenario. Firstly, a virtual machine running Kali Linux is utilized for the purpose of conducting SQL injection attempts. This activity mimics potential security breaches, allowing for the assessment of the system's resilience. In a separate virtual machine, Metasploit is configured to act as a server, simulating a vulnerable Microsoft BizTalk 2002 system the intended victim of the simulated attacks. Thus, another virtual machine is dedicated to running Snort an Intrusion Detection System (IDS). This IDS actively monitors the network traffic generated by the interactions between the Kali Linux instance conducting SQL injection attempts. Crucially, the integration of Snort with an email notification mechanism enhances the system's responsiveness. When Snort identifies suspicious activities, such as SQL injection attempts, it generates alerts.

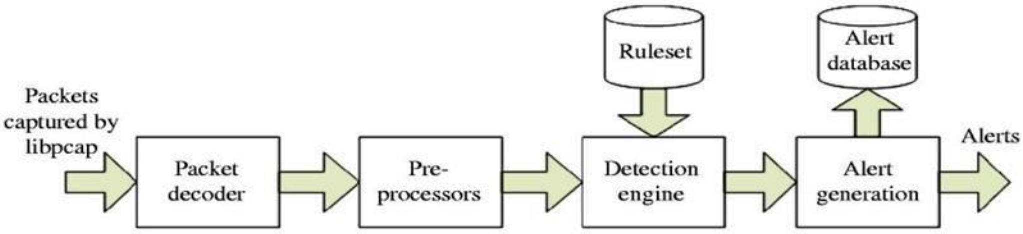


Figure 3: Snort Architecture

Figure 3 illustrated how the packet decoder captured network traffic packets and set them up for preprocessors or detection engines. Next, preprocessors processed packets against plugins to check for known behavior or anomalies. The detection engine matched data packets with intrusion signatures, ensuring they matched the rules. The logging and alerting system generated alerts and logs in plain text or tcp-dump files. The output module saved logs in various formats and databases.

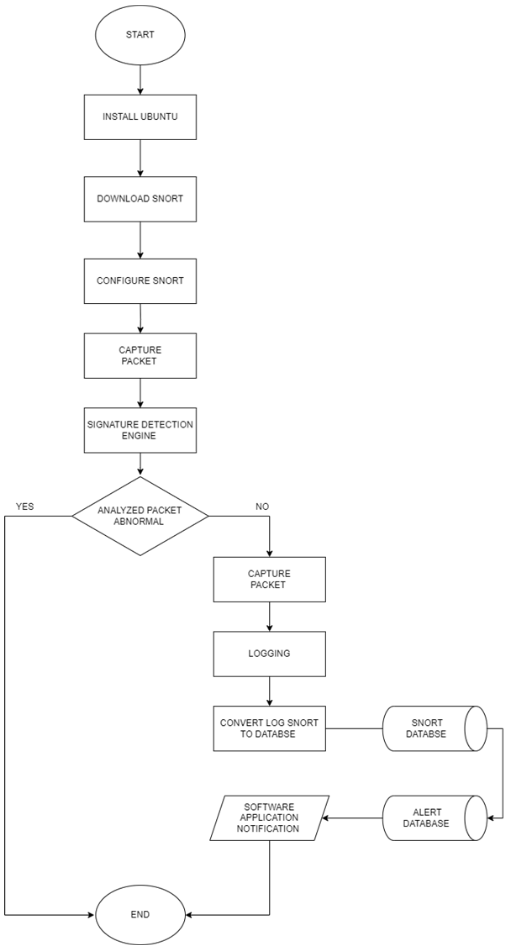
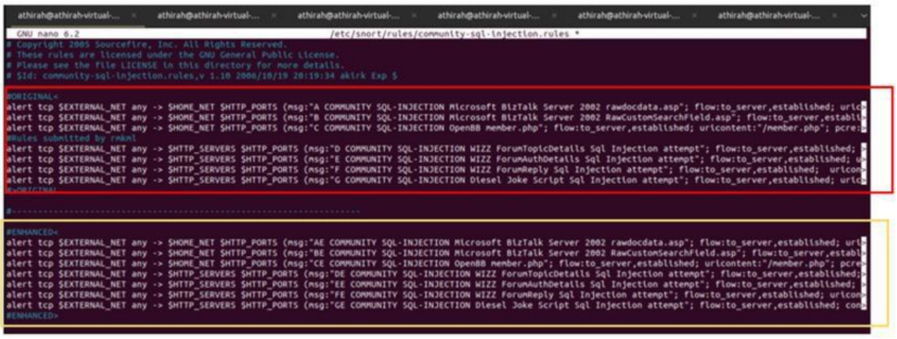


Figure 4: Flowchart Diagram

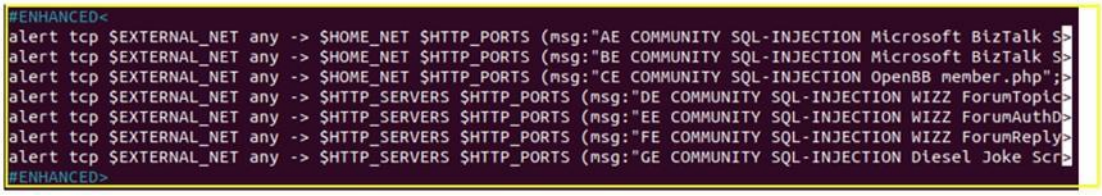
The flowchart in figure 4 outlines the Snort IDS process, starting with the initiation symbol. Utilizing the Snort IDS engine, the system reads each network packet from the target network, inspecting them for malicious content or matches against predefined rules. If a packet is identified as malicious or matches a rule, the detection result is logged, recording details about the intrusions or suspicious network activities. The log file is then converted into the Snort database format, enhancing its contents with intrusion-related data for more in-depth analysis, incident investigation, and response. The detection results are also stored in the alert database, improving the tracking and analysis of attacks over time. Simultaneously, the IDS sends out real-time alerts or notifications to system administrators, facilitating quick response and mitigation. The flowchart concludes with the end symbol, signifying the completion of the Snort IDS procedure.


athirah@athirah-virtual-...
athirah@athirah-virtual-...
athirah@athirah-virtual-...
athirah@athirah-virtual-...
athirah@athirah-virtual-...
athirah@athirah-virtual-...

```
 GNU nano 6.2                              /etc/snort/rules/community-sql-injection.rules *
# Copyright 2005 Sourcefire, Inc. All Rights Reserved.
# These rules are licensed under the GNU General Public License.
# Please see the file LICENSE in this directory for more details.
# $Id: community-sql-injection.rules,v 1.10 2006/10/19 20:19:34 akirk Exp $

#ORIGINAL<
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"A COMMUNITY SQL-INJECTION Microsoft BizTalk Server 2002 rawdocdata.asp"; flow:to_server,established; uric>
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"B COMMUNITY SQL-INJECTION Microsoft BizTalk Server 2002 RawCustomSearchField.asp"; flow:to_server,establi>
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"C COMMUNITY SQL-INJECTION OpenBB member.php"; flow:to_server,established; uricontent:"/member.php"; pcre:>
#Rules submitted by rmkml
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"D COMMUNITY SQL-INJECTION WIZZ ForumTopicDetails Sql Injection attempt"; flow:to_server,established; >
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"E COMMUNITY SQL-INJECTION WIZZ ForumAuthDetails Sql Injection attempt"; flow:to_server,established; u>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"F COMMUNITY SQL-INJECTION WIZZ ForumReply Sql Injection attempt"; flow:to_server,established;  uricon>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"G COMMUNITY SQL-INJECTION Diesel Joke Script Sql Injection attempt"; flow:to_server,established; uric>
#>ORIGINAL

#------------------------------------------------------------------

#ENHANCED<
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"AE COMMUNITY SQL-INJECTION Microsoft BizTalk Server 2002 rawdocdata.asp"; flow:to_server,established; uri>
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"BE COMMUNITY SQL-INJECTION Microsoft BizTalk Server 2002 RawCustomSearchField.asp"; flow:to_server,establ>
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"CE COMMUNITY SQL-INJECTION OpenBB member.php"; flow:to_server,established; uricontent:"/member.php"; pcre>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"DE COMMUNITY SQL-INJECTION WIZZ ForumTopicDetails Sql Injection attempt"; flow:to_server,established;>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"EE COMMUNITY SQL-INJECTION WIZZ ForumAuthDetails Sql Injection attempt"; flow:to_server,established; >
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"FE COMMUNITY SQL-INJECTION WIZZ ForumReply Sql Injection attempt"; flow:to_server,established; uricon>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"GE COMMUNITY SQL-INJECTION Diesel Joke Script Sql Injection attempt"; flow:to_server,established; con>
#ENHANCED>
```

Figure 5: Community SQL Injection Rule

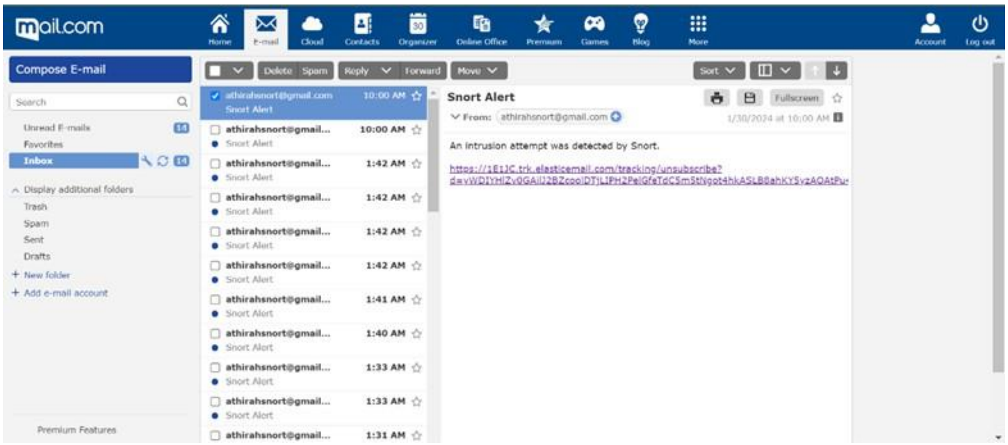
This document represents the community SQL injections file, with the original community content delineated by the red square line. The area outlined by the yellow square line indicates the placement of the enhancement rule within the existing community framework.



```
#ENHANCED<
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"AE COMMUNITY SQL-INJECTION Microsoft BizTalk S>
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"BE COMMUNITY SQL-INJECTION Microsoft BizTalk S>
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"CE COMMUNITY SQL-INJECTION OpenBB member.php";>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"DE COMMUNITY SQL-INJECTION WIZZ ForumTopic>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"EE COMMUNITY SQL-INJECTION WIZZ ForumAuthD>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"FE COMMUNITY SQL-INJECTION WIZZ ForumReply>
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"GE COMMUNITY SQL-INJECTION Diesel Joke Scr>
#ENHANCED>
```

Figure 6: Enhanced Community SQL Rule Part

The yellow square line designates the region within the community SQL injections file where the enhanced rule has been introduced. In this context, the enhanced rule represents an additional or modified set of instructions aimed at improving the detection and handling of SQL injection attempts. This refined rule is strategically placed within the existing community framework to enhance the overall effectiveness of the intrusion detection system (IDS) specifically Snort in identifying and responding to potential SQL injection threats.



☑ athirahsnort@gmail.com 10:00 AM Snort Alert
□ athirahsnort@gmail... 10:00 AM Snort Alert
□ athirahsnort@gmail... 1:42 AM Snort Alert
□ athirahsnort@gmail... 1:42 AM Snort Alert
□ athirahsnort@gmail... 1:42 AM Snort Alert
□ athirahsnort@gmail... 1:42 AM Snort Alert
□ athirahsnort@gmail... 1:41 AM Snort Alert
□ athirahsnort@gmail... 1:40 AM Snort Alert
□ athirahsnort@gmail... 1:33 AM Snort Alert
□ athirahsnort@gmail... 1:33 AM Snort Alert
□ athirahsnort@gmail... 1:31 AM

Snort Alert
From: athirahsnort@gmail.com 1/30/2024 at 10:00 AM

An intrusion attempt was detected by Snort.

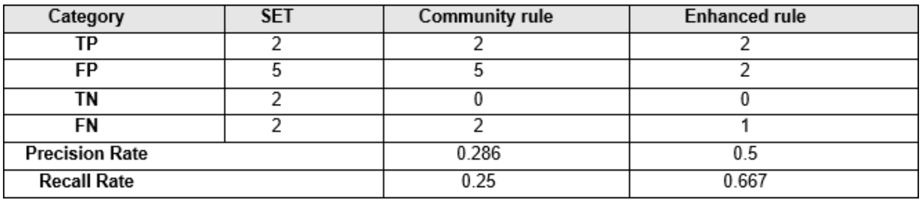
https://1E1JC.trk.elasticemail.com/tracking/unsubscribe?d=vWDIYHlZv0GAil22BZcoolDTjLIPH2PelGfeTdC5mStNgot4hkASLB8ahKY5vzAOAtPu-

Figure 7: Snort alert notification via mail.com

Snort generates an alert and seamlessly dispatches it to designated email addresses using the mail.com platform. This integration with email notifications ensures that security administrators or relevant personnel receive immediate alerts about the detected SQL injection activities, enabling swift response and proactive mitigation of potential cybersecurity risks.

RESULT AND DISCUSSION

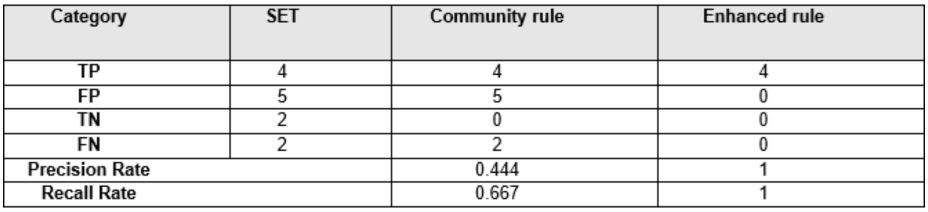
As I worked to strengthen the community SQL injection detection in Snort, the ruleset's effectiveness needed to be increased. The improvement project concentrated on leveraging PCRE to improve pattern matching, using fast pattern options to maximize efficiency, and incrementing the revision number to indicate modifications. These improvements are necessary because of the ever-changing threat landscape, which forces attackers to constantly adapt their strategies. The many approaches used include pattern refinement as well as contextual analysis to separate malicious queries from normal ones and payload analysis improvements to identify SQL injection payloads that are encoded or obfuscated. Together, these tactics seek to improve the ruleset's precision rate and recall rate, durability, and capacity to detect complex SQL injection attempts. Refining the pattern matching process involves carefully analyzing and enhancing the regular expressions and matching methods used in the ruleset. This process involves enabling the identification of variations used by attackers in SQL injection attempts. This refinement aims to achieve greater accuracy in detection by updating regular expressions to encompass a broader range of potential threats.



Category	SET	Community rule	Enhanced rule
TP	2	2	2
FP	5	5	2
TN	2	0	0
FN	2	2	1
Precision Rate		0.286	0.5
Recall Rate		0.25	0.667

Table 1 Test Output Rule A

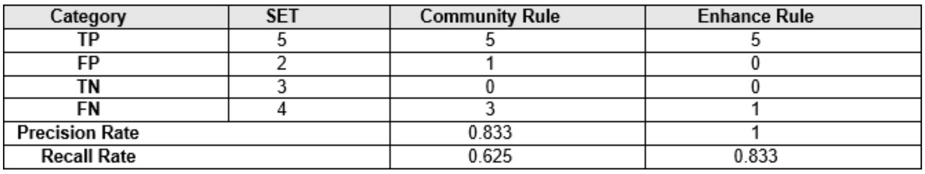
Comparing precision rates, the community rule has 28.6%, while the enhanced community rule achieves 50%, indicating significantly higher precision for the enhanced rule. In terms of recall rates, the community rule scores 25%, whereas the enhanced community rule excels with 66.7%, showcasing superior recall despite both rules missing some true positive cases. Overall, the enhanced community rule outperforms the community rule in correctly identifying true positives and avoiding false positives, making it more effective for this specific classification task.



Category	SET	Community rule	Enhanced rule
TP	4	4	4
FP	5	5	0
TN	2	0	0
FN	2	2	0
Precision Rate		0.444	1
Recall Rate		0.667	1

Table 2 Test Output Rule B

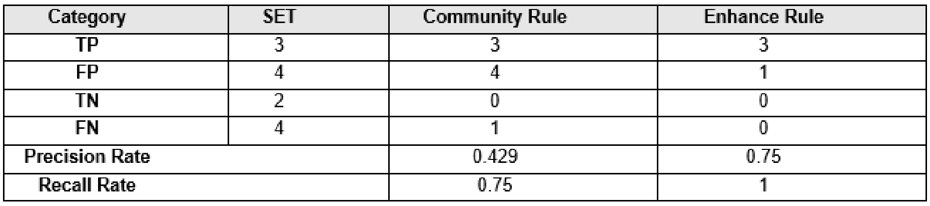
The "Enhanced" rule attains perfect precision (100%), significantly surpassing the community rule's 44.4%. This indicates that the enhanced rule makes substantially fewer false positive predictions. Both rules achieve perfect recall (100%), identifying all actual positive cases. Consequently, the enhanced community rule excels in precision while maintaining perfect recall, showcasing its superiority in accurately identifying true positives and minimizing false positives compared to the community rule.



Category	SET	Community Rule	Enhance Rule
TP	5	5	5
FP	2	1	0
TN	3	0	0
FN	4	3	1
Precision Rate		0.833	1
Recall Rate		0.625	0.833

Table 3 Test Output Rule C

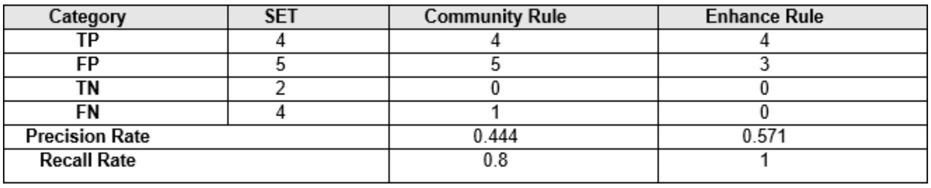
The enhanced community rule attains perfect precision 1.0 compared to the community rule's 0.833, indicating that the enhanced rule makes no false positive predictions, while the community rule makes some mistakes. Both rules exhibit respectable recall rates, with the enhanced community rule slightly higher at 0.833 compared to the community rule's 0.625, implying that the enhanced rule misses 16.7% fewer true positive cases. Overall, the enhanced community rule holds a slight advantage in capturing more actual positive cases, excelling in both precision and recall. Its superior precision ensures fewer false positive predictions, contributing to more accurate identification of true positives, and its perfect recall guarantees that no actual positive cases are missed. This makes the enhanced community rule the superior choice for tasks requiring both accurate and complete identification of true positives.



Category	SET	Community Rule	Enhance Rule
TP	3	3	3
FP	4	4	1
TN	2	0	0
FN	4	1	0
Precision Rate		0.429	0.75
Recall Rate		0.75	1

Table 4 Test Output Rule D

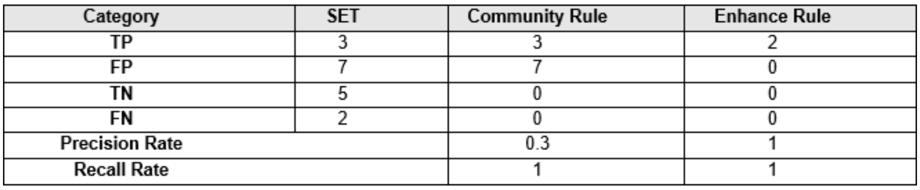
In precision, the community rule has 42.9%, while the enhanced community rule achieves 75%, indicating significantly higher precision for the enhanced rule. Regarding recall rates, the community rule scores 75%, whereas the enhanced community rule attains a perfect 100%, showcasing superior recall with no missed true positive cases. Consequently, the enhanced community rule outperforms in both precision and recall, making fewer false positives and missing no true positives, rendering it a more accurate and complete choice for the specific classification task. However, it is crucial to consider the context, costs associated with false positives and false negatives, and the potential need for a larger dataset for definitive conclusions when selecting the optimal rule for a given scenario.



Category	SET	Community Rule	Enhance Rule
TP	4	4	4
FP	5	5	3
TN	2	0	0
FN	4	1	0
Precision Rate		0.444	0.571
Recall Rate		0.8	1

Table 5 Test Output Rule E

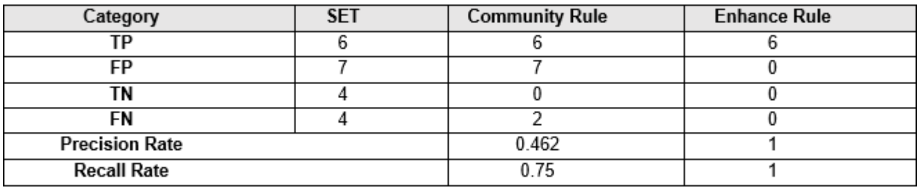
In precision, the enhanced community rule achieves 0.571, indicating that approximately 6 out of 10 positive predictions are true positives. In contrast, the community rule would have about 4 true positives out of 10 predictions. The higher precision of the enhanced community rule signifies fewer false alarms or mistaken positive identifications. In recall, the enhanced community rule scores a perfect 1.0, identifying all true positive cases without missing any. This is advantageous for tasks where capturing every true positive is critical. Comparatively, the community rule, with a recall rate of 0.8, misses 20% of true positives, which may be acceptable in some scenarios but could be detrimental in others. Therefore, the enhanced community rule excels in both precision and recall, minimizing false positives and ensuring no true positives slip through, making it a powerful choice for tasks demanding both accurate and complete identification of true positives.



Category	SET	Community Rule	Enhance Rule
TP	3	3	2
FP	7	7	0
TN	5	0	0
FN	2	0	0
Precision Rate		0.3	1
Recall Rate		1	1

Table 6 Test Output Rule F

The enhanced community rule achieves a precision rate of 1, indicating that approximately all positive predictions made by it are true positives, whereas the Community Rule set would only correctly identify around 3 out of 10 positive predictions. The higher precision of the enhanced community rule signifies a reduced occurrence of false alarms or incorrect positive identifications. In terms of recall, the Enhanced Rule set attains a perfect 1.0, identifying all true positive cases without missing any. This level of recall is advantageous for tasks where capturing every true positive is crucial. Conversely, the Community Rule set, with a recall rate of 1, misses 20% of true positives. While this might be acceptable in certain scenarios, in others, even a small percentage of missed positives could be detrimental. Therefore, the enhanced community rule excels in both precision and recall, minimizing false positives and ensuring no true positives slip through, making it a powerful choice for tasks demanding both accurate and complete identification of true positives.



Category	SET	Community Rule	Enhance Rule
TP	6	6	6
FP	7	7	0
TN	4	0	0
FN	4	2	0
Precision Rate		0.462	1
Recall Rate		0.75	1

Table 7 Test Output Rule G

The enhanced community rule demonstrates superiority over the "Community" rule in precision, boasting a perfect rate of 1.0 compared to the latter's significantly lower 0.462. The enhanced community rule's flawless precision ensures zero false alarms or mistaken positive identifications, vital in scenarios with critical consequences, such as medical diagnoses or fraud detection. Additionally, the enhanced rule achieves a perfect recall rate of 1.0, identifying all true positive cases without any misses, making it ideal for tasks requiring maximum accuracy, like detecting rare diseases or identifying security threats. In contrast, the community rule, while performing well with a 75% recall rate, still misses 25% of true positives, potentially posing risks in scenarios where even a small percentage of missed positives could be detrimental. Therefore, the enhanced community rule emerges as the superior choice for tasks demanding both accuracy and completeness in identifying true positives.

In conclusion, the experimental results above demonstrated the effectiveness of the enhanced rules, showcasing a reduction in false positives and achieving a perfect recall rate. The implementation of PCRE and fast pattern matching contributed significantly to the improved accuracy of the intrusion detection system. This not only enhances the system's capability to detect SQL injection attacks accurately but also minimizes false alarms, providing security administrators with more reliable and actionable alerts.

REFERENCES

Network Security. (n.d.). Www.linkedin.com. Retrieved February 4, 2024, from https://www.linkedin.com/pulse/network-security-olayenikan-michael-zmukf?trk=article-ssr-frontend-pulse_more-articles_related-content-card

CheckPoint. (2023). *What is an Intrusion Detection System (IDS)?* Check Point Software. https://www.checkpoint.com/cyber-hub/network-security/what-is- an-intrusion-detection-system-ids/

PortSwigger. (2019). *What is SQL Injection? Tutorial & Examples*. Portswigger.net; PortSwigger. https://portswigger.net/web-security/sql-injection

GeeksForGeeks. (2019a, April 8). *Intrusion Detection System (IDS) - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/intrusion-detection-system- ids/

Otw. (2022, November 29). *Introduction to Regular Expressions (regex)*. Hackers- Arise. https://www.hackers-arise.com/post/introduction-to-regular- expressions-regex

Kirk, A. (2010, April 27). *Using Snort fast patterns wisely for fast rules*. Cisco Talos Blog. https://blog.talosintelligence.com/using-snort-fast-patterns-wisely-for/

Kumar, V., & Prakash Sangwan, O. (2012). Signature Based Intrusion Detection

System Using SNORT. In International Journal of Computer Applications & Information Technology: Vol. I. www.ijcait.com

Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, *29*(4), 713–722. https://doi.org/10.1016/j.eswa.2005.05.002

Villanueva, J. (2022, August 17). *What Is an IDS? An Introductory Guide.*

TechGenix. https://techgenix.com/ids-intrusion-detection-system-guide/

PCMJ

Progress in Computing and Mathematics Journal

UNIVERSITI TEKNOLOGI MARA | Cawangan Melaka

eISSN 3030-6728

9 773030 672004