Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.040305
Article

An Enhanced Intelligent Intrusion Detection System to Secure E-Commerce Communication Systems

Adil Hussain[1], Kashif Naseer Qureshi[2,*], Khalid Javeed[3] and Musaed Alhussein[4]

[1]Department of Transportation Engineering, Chang'an University, Xi'an, China
[2]Department of Electronic & Computer Engineering, University of Limerick, Limerick, V94 T9PX, Ireland
[3]Department of Computer Engineering, College of Computing and Informatics, University of Sharjah, Sharjah, United Arab Emirates
[4]Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh, 11543, Kingdom of Saudi Arabia

*Corresponding Author: Kashif Naseer Qureshi. Email: kashifnaseer.qureshi@ul.ie
Received: 13 March 2023; Accepted: 23 May 2023; Published: 28 July 2023

Abstract: Information and communication technologies are spreading rapidly due to their fast proliferation in many fields. The number of Internet users has led to a spike in cyber-attack incidents. E-commerce applications, such as online banking, marketing, trading, and other online businesses, play an integral role in our lives. Network Intrusion Detection System (NIDS) is essential to protect the network from unauthorized access and against other cyber-attacks. The existing NIDS systems are based on the Backward Oracle Matching (BOM) algorithm, which minimizes the false alarm rate and causes of high packet drop ratio. This paper discussed the existing NIDS systems and different used pattern-matching techniques regarding their weaknesses and limitations. To address the existing system issues, this paper proposes an enhanced version of the BOM algorithm by using multiple pattern-matching methods for the NIDS system to improve the network performance. The proposed solution is tested in simulation with existing solutions using the Snort and NSL-KDD datasets. The experimental results indicated that the proposed solution performed better than the existing solutions and achieved a 5.17% detection rate and a 0.22% lower false alarm rate than the existing solution.

Keywords: E-commerce; NIDS; security; algorithm; network; applications; CIA; detection

1 Introduction

Internet users have increased due to attractive applications and services like e-commerce and online applications. The well-known e-commerce applications are online banking systems, online marketing, and smart industries. The developments of new and integrated technologies offered various new features, including remote monitoring and control systems and an online knowledge portal

[1,2]. Due to sensitive user data, identifying unknown attacks is a significant concern in e-commerce applications. Computer security measures such as Confidentiality Integrity and Availability (CIA) may breach because of malicious intrusions, security attacks, and database attacks. Traditional techniques are used mainly as standard security methods to secure computers and networks [3,4]. A Network Intrusion Detection System (NIDS) collects information that violates security policies and stores the data in a database [5]. These systems check the network input or output packages for intrusion signals. An effective NIDS system can identify and automatically respond by writing security logs or issuing warnings to most intrusion activities [6,7]. The three essential characteristics of a NIDS system are accuracy, extensibility, and adaptability. NIDS should discriminate between authorized and unauthorized users and use the first line of defense to avoid intrusions and deviations from the inside and outside of the network. Fig. 1 shows the e-commerce systems and NIDS classification overview.

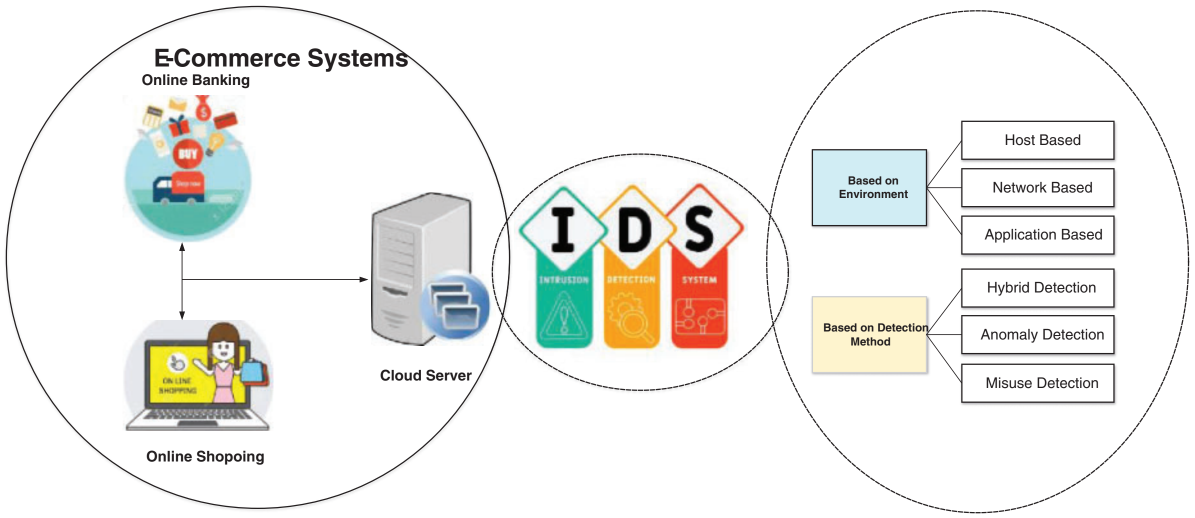

Figure 1: E-commerce systems and NIDS classification overview

NIDSs are implemented as signature-based, anomaly-based, and hybrid-based NIDS [8]. Signature-based NIDS searches the database for specific known attacks [9]. The audit logs are searched for occurrences interpreted as previous attack signs. Similarly, the NIDS generates an attack alarm to notify the network security and manage if the new action matches the existing database signatures [10,11]. The anomaly-based NIDS searches the typical behavior patterns saved in a database for statistical anomaly-based systems [12]. Both network activities are correctly monitored and evaluated in all of these networks. Any deviation from standard patterns is identified as an attack, and the IDS generates an alert to notify the security manager of the newly discovered attacks. The anomaly detection system does not require any prior knowledge of signatures.

It should be noted that false positives are common in these techniques because all abnormal patterns are not correctly attached [13]. A hybrid NIDS combines signature and anomaly detection systems. To improve identification rates and reduce false alarm rates, hybrid NIDS uses a signature-based mechanism to detect a match and search for anomalies if the typical baseline profile deviates from that. IDS are commonly divided into network-based and host-based systems [12,14]. Network-based IDS (NIDS) [15] monitor network traffic and evaluate all operations rather than packets directed at a specific host to detect and identify intrusions. More information about Denial of Service (DoS) attacks can be obtained, and intrusion detection is performed based on host analysis of attack signature log data [16,17]. Host NIDS examines host-based control streams for attack identification, such as audit trails, device logs, and program logs. Traditional host-based IDS conducts intrusion analysis primarily with an independent server and standalone monitoring tools. System inquiry single host is loaded with static computing and storage capacity [17].

Malicious cyber-attacks pose significant security issues that require a new, flexible, and reliable NIDS [18]. NIDS automatically detects and recognizes network and host systems intrusions, attacks, and violations. To combat intrusions, NIDS must be highly accurate and adaptable [19]. One of the issues faced in network security is the design of security measures to protect systems and network resources from unauthorized access. Pattern-matching algorithms are standard for NIDS systems and compare matching algorithms for each rule in an incoming network packet. Unfortunately, the system's detection efficiency has been significantly reduced. In addition, the intrusion detection criteria are constantly being updated, making it challenging to detect advanced network attacks precisely. As a result, in-network IDS systems, highly efficient pattern-matching algorithms are helpful. Because of the detection rate and processing speed, signature-based IDS systems have suffered from overhead due to high network data traffic.

Furthermore, existing signature-based methods have a slower detection speed, which can be improved with new techniques. The primary goal of this paper is to improve detection rates by adopting a pattern-matching algorithm. To enhance the performance of network IDS, the Backward Oracle Matching (BOM) algorithm is used to minimize the packet drop ratio and false alarms. To overcome the existing IDS systems limitations, the main objectives of this paper are as follows:

- To discuss the existing IDS systems, their types, and different pattern-matching techniques to find the research gap.
- To propose an enhanced version of the BOM algorithm for the IDS system by using multiple pattern-matching techniques.
- To evaluate the proposed system by using the Snort tool and NSL-KDD dataset.

The rest of the paper is organized as follows: Section 2 discusses related work, including summaries of existing IDS-based systems using different pattern-matching algorithms and checking their limitations. Section 3 presents the design and development details and utilized methodology. Section 4 discusses the details of the experiment to evaluate the proposed system with existing solutions. Finally, the paper concludes in Section 5 with a future direction.

2 Related Work

Different pattern-matching algorithms used for network IDS are reviewed in this section. An IDS records and analyzes all incoming network packets to detect vulnerabilities or intrusions. The system contains four modules: packet capturing, decryption, pre-processing, and string/pattern matching. The authors [20] described the algorithms and techniques most commonly used for the hardware implementation to fit strings. This solution concern is string matching of the computer-intensive portion, which is already being sped up by several hardware architectures/designs. As a result, it is critical to investigate current hardware architectures for string-matching algorithms.

Authors in [21], used a head-body finite automated algorithm for pattern-matching for evaluating packet payloads. This method is used in a multi-core GPP parallelism and a single-instruction multi-data operation using the Aho-Corasick (AC) algorithm. The proposed approach achieved 58% efficiency according to Snort and ClamAV patterns. Authors in [22], suggested a novel algorithm using effective Wu-member, Commentz-Walter, and its versions to match multi patterns and ruled out the

Wu-member prefix table. Furthermore, the authors employed only one pointer table instead of a three-shift table. Finally, a list of dynamic indicators in the hash table is used. The proposed solution achieved better results as compared with the existing algorithms and other pattern-compatibility tools with these adjustments in terms of time and space.

Authors in [23] presented the IDS technique based on signatures and matching strings. This study revealed how to optimize Snort's as a popular string-matching system using various string-matching algorithms. The authors used the Boyer-Moore string matching technique and numerous alternate algorithms. They discussed that more than one method is needed for the Snort ruleset. The experiment results indicated a five times higher average speed for specific package matching. Authors in [24] summarized key pattern algorithms and their variants in the BOM method, which is considerably faster than other algorithms. In this study, the authors recommended changes to the BOM algorithm to match the patterns of a multi-level efficient IDS architecture. The suggested Snort tool is used to evaluate the proposed algorithm's performance. In another paper, the same authors in [25], proposed a layer-based architecture where all tiers can provide the needed functionality. It can be coupled with the Snort tool with Code Refactoring to demonstrate the efficiency of the suggested design. A setup is also proposed to assess the performance of the updated Snort tool. Authors in [26], presented a new alert management system for Snort using Stateful pattern matching, which classifies warnings and provides network security. The system accurately identifies the attacks and different kinds of alerts. The suggested algorithm separates severe, low significance, and irrelevant signs with good performance. Based on the experimental findings on the DARPA KDD Cup 99 dataset, the system is categorized by notifications and reasons, significantly reducing false alarms.

Authors in [27], discussed the problem of multiple pattern matching and proposed an efficient technique for increasing the processing performance of IDS or other comparable programs that use for pattern matching, such as antivirus systems. They demonstrate V-PATCH, a cache-efficient filtering method combined with modern vectorization techniques that enable data parallelism within each processor unit. The authors also presented an analytical model for this approach, predicting performance and evaluating new real-time designs. Authors in [28] used an improved Wu-Manber algorithm, a rapid pattern-matching method commonly used in IDS. It generates hash tables by employing algorithms based on the shortest patterns. The disparity between patterns frequently reduces the algorithm's efficiency. This paper also incorporated Bloom filters to reduce the reliance on the shortest patterns. The experimental results indicated the proposed solution reduces matching time by 10% in the worst case and 78% in the best case. It also reduces memory consumption by 0.3%.

Authors in [29], presented a new pattern-matching method used as a future enhancement platform to develop fast pattern-matching algorithms. The most commonly used algorithms are Boyer-Moore, Aho-Corasick, Nave String Search, Rabin Karp String Search, and Knuth-Morris-Pratt. The performance is evaluated where the processing time is faster than the available proven pattern-matching. Authors in [30], proposed a network IDS based on a combinatorial algorithm using various databases. The proposed solution is identifying new threats and accelerating network traffic by the combinatorial algorithm during traffic analysis. Based on previous research, a reasonable threshold of 12 is selected to reduce the false-positive rate. The proposed solution is tested against other online schemes, and the results are a low false-positive rate of 3% and higher accuracy of 96.5%.

Authors in [31], implemented hardware using a well-known Aho-Corasik pattern-matching technique. The findings showed that the Aho-Corasick method efficiently utilizes resources (memory and logical cells). Furthermore, using this method, the same patterns can be accommodated in a larger FPGA. More progress in the efficient hardware implementation of pattern-matching algorithms is needed. Authors in [32], presented and analyzed several pattern-matching algorithms and methodologies for detecting patterns in network intrusion detection. After implementing the algorithms, it is discovered that the KMP and Trie data structures are realistically competent. KMP requires less time than the naïve technique by producing a prefix array, whereas the Rabin-Karp algorithm and Trie data structure are used for a smaller number of patterns.

On the other hand, Trie takes longer than all other methods for fewer patterns. An effective pattern-matching technique significantly takes less time, as required by network intrusion detection, with rapid and scalable protection. Therefore, selecting a proper algorithm is critical for the performance improvement of network IDS.

Authors in [33], discussed the problem of multiple pattern matching and proposed an IoT feature extraction and IDS algorithm by using deep migration learning model. The authors presented a modeling scheme of migration learning and data extraction. The proposed solution enhanced the optimization whereas the transfer mine model established the connection of source and target task. The proposed solution is tested on public complex data to evaluate the data set robustness and validation. However, authors neglected the important factors like accuracy of IDS system to ensure the classification. The authors in [34] explained that the Wu-Manber algorithm is a rapid pattern-matching method often employed in IDS. It builds hash tables using hash algorithms based on the shortest patterns. However, the disparity between patterns frequently reduces the algorithm's efficiency. In this study, authors presented an enhanced Wu-Manber algorithm by adding Bloom filters and minimizing reliance on the shortest patterns. According to the testing findings, this technique decreases matching time by 10% in the worst scenario and 78% in the best situation compared to the original Wu-Manber algorithm. It also reduces memory by 0.3%. The authors in [35] introduced a new pattern-matching approach that might serve as a future enhancement platform. Researchers in this discipline developed a quick pattern-matching algorithm. The most common algorithms are Boyer-Moore, Aho-Corasick, Nave String Search, Rabin Karp String Search, and Knuth-Morris-Pratt. Based on this Research, authors designed algorithms that process text data using various algorithms and strategies. The performance is evaluated and compared the processing time with the quickest proven pattern-matching algorithms.

The authors in [36], proposed a network IDS system that integrates combination of algorithms such as a feature selection algorithm, a clustering algorithm, and filter and wrapper methods. The wrapper strategy is know as the Cuttlefish Algorithm (CFA), and the filter approach is known as the feature grouping based on Linear Correlation Coefficient (FGLCC) algorithm. The classifier in the suggested method is a decision tree. The suggested method was put to the test on sizable datasets taken from the KDD Cup 99 dataset in order to evaluate its performance. Authors in [37] explored the possibility of a novel hrdware implementation of the traditional Aho-Corasik pattern matching technique. The findings revealed that the Aho-Corasick algorithm uses systems resources efficiently like memory and logical cells even though it results in lower performance. This method allows for the compression of the same pattern set onto a smaller FPGA chip. It is anticipated that hardware implementations of pattern matching algorithms will continue to advance.

The discussed existing solutions have limitations in terms of low data throughput, low detection rate, high false alarm rate, tests on old datasets, use memory, and hardware limitations. Table 1 shows a comparison of existing methods and their limitations.

Table 1: Comparison of existing solutions

S#	Existing solutions	Used methods	Limitations
1	Head-body finite automated algorithm [27]	Pattern matching	• High space • Low throughput
2	An effective novel algorithm [28]	Wu Manber, Commentz Walter Algorithm	• Less efficient because of separate trios for all the patterns
3	Numerous patterns matching based on NIDS [29]	Pattern Matching	• High cache size • String matching component
4	Backward Oracle Matching algorithm [30]	String matching	• Low detection rate • High false alarm rate
5	A new alert management system for Snort [32]	Stateful pattern matching	• Performance tested on an old dataset
6	Data Feature Extraction Algorithm for IDS [33]	Multiple pattern matching	• Cache based and uses cache
7	Bloom filters in the Wu Manber algorithm [34]	Wu Manber Algorithm	• High memory usage
8	A new pattern-grouping approach [35]	Pattern matching	• Needs improvements for better efficiency
9	Combinational algorithms based IDS [36]	Feature Selection algorithm	• False alarm rates need to be reduced
10	New hardware for Aho-Corasik pattern matching [37]	Aho-Corasik Algorithm	• Improvements are required for better hardware implementation

3 Proposed System Design and Development

An IDS is a tool that every organization uses for intrusion detection and prevention. However, organizations still need some help with existing IDS systems and establishing new methods to overcome the limitations, especially for high data traffic, low detection rate, and manual observation. During the network IDS setup, these constraints determine the stimulus. Networks are vulnerable and jeopardize confidentiality, availability, and integrity. Such attacks can cost a lot in terms of economics, status, and slow destruction for customers. There is a need to improve detection systems efficiency by focusing on critical attacks such as port scanning, hijacking sessions, and DoS [38]. Pattern-matching algorithms are used mainly for IDS and reveal a vibrant benefit. There are many pattern-based algorithms, but it is critical to analyze these algorithms and improve systems performance. Network IDS executes matching algorithms to fit each rule of the incoming network packet. As a result, the accuracy rate of the device is reduced substantially. Furthermore, the rules for intrusion detection increase significantly and make accurate detection difficult because of advanced network attacks. Therefore, the detection systems depend on efficient pattern-matching algorithm design.

An IDS based on an extended BOM algorithm achieved better patterns matching and improved the system performance. The layer-based IDS architecture contains three algorithms. The key reason for using the extended BOM is to increase the speed and efficiency of the detection rates to handle high data traffic. The layer-based architecture with improved algorithms can handle the existing IDS challenges and attacks. In most cases, the extended BOM algorithm is preferable because it is flexible and quick. It uses a fast loop implementation that includes two consecutive transitions within a single table. Unlike the original algorithm, it is also effective with short patterns. The design layers are sequential to prevent problems and ensure network security. For example, a DoS attack can be used to manipulate the system confidentiality and affect the availability of the network services; packet capturing and port scanning attacks can compromise the integrity. The three features can be developed with three-tier architecture, which detects and prevent attacks on each level. All three layers use an extended BOM algorithm to detect attacks more effectively. Each fast-loop iteration involves two transitions that affect the algorithm's overall performance. Extended BOM is implemented in the tool to increase the speed of the existing architecture.

3.1 Algorithms for Better Pattern Matching

Boyer-Moore-based algorithms are used to match pattern suffixes. From right to left, scanning the current text window matches specific prefixes or factors to lengthen shifts. This is possible through the use of factor automata and factor oracles. The pattern 'p' is the only factor length greater than or equal to 'm', which is the factor oracle recognizes. It can be used to search for patterns in texts despite identifying words that are not factors of the pattern. In addition, the oracle's computation is linear in time and space over the pattern. The bad character heuristic is typically iterated in a checkless cycle to quickly find an instance of the pattern's rightmost character algorithms based on the flawed character heuristic function optimally only with large alphabets and modest patterns. In addition, test results indicate that the performance of automata to match prefixes or components improves noticeably as the pattern length increases. This behavior occurs because an occurrence of the rightmost character of the window, i.e., t[j], can be found in large patterns. As the pattern lengthens and the alphabet size decreases, the probability that the rightmost occurrence is close to the rightmost location increases. In this instance, a rapid loop iteration produces a brief movement. Contrarily, when using an oracle for matching, it is usual that more transitions cannot be made beyond a specific number of characters. This method generally aims for more than one character per iteration but produces a length shift 'm'. The fast loop is designed to read two consecutive characters for each iteration, increasing the likelihood that it would identify an undefined oracle transition. The extended BOM algorithm is created and found to be quick and adaptable.

3.2 Algorithm 1 for Availability Layer

All the patterns must first be sent to build a state machine for the provided pattern set. A factor oracle is then created using the supply of reversed patterns. Using changed patterns reduces the difficulty of matching. To determine the minimum, a search for brief patterns is conducted. Throughout the pattern-matching procedure, a vital position is maintained to advertise the right side of the halting position. The pattern matching method is initiated using factor oracle. Because the factor oracle utilizes a reversed pattern, matching the input to the pattern proceeds from right to left. The search procedure concludes when there is a mismatch or all input characters have been scanned. In cases where the process is transformed into a state machine, imperfections cannot lead to advancements in the oracle factor. The incorrect character is relocated to the right side and the state machine's starting position. Up until the longest prefix match occurs, begin matching the state machine. As stated in Algorithm 1, which does not require input scanning as a single character. There can be several characters in a scan window at once. With the entire matching process, the critical place pointing stops and window length avoid superfluous input characters.

Algorithm 1: Multiple Keyword Set Factor Oracle Construction

```
1.  Input:
2.  $x$: keywords array
3.  $m$: array of $p$ integers of keyword length
4.  factor oracle = TrieConstruct(x, m, p)
5.  $i$ = factor oracle root
6.  S[$i$] = ø
7.  **for** $q$ = states from factor oracle, **do**
8.  $l$ = parent($q$)
9.  k = S[$l$]
10.     **while** $k$ = ø and no transition from $k$ to $q$ **do**
                Create a transition and label it
                $k$ = S[$k$]
11.     **end**
12.     **if** $k$ is ø, **then**
                Initialize $S$ to factor Oracle root $i$
13.     **else**
                Index state where $k$ leads to transition
14.     **end**
15. **end**
```

3.3 Algorithm 2 for Confidentiality Layer

In Algorithm 2, the BOM algorithm extended version is used. A fast loop in this algorithm makes it more efficient than the original backward algorithm. In addition, extended BOM enhances the system's efficiency, as discussed in the results section.

Algorithm 2: Pattern Search using Factor Oracle

```
1.  Input:
2.  $t$ = text input array  //  t be a text of length n
3.  $n$ = text length
4.  p: patterns     //  p is a pattern of length m
5.  m: keyword length
6.      **factororacle** = FactorOracle(p)
7.      **for** a ∈ Σ **do**
8.          q = **factororacle** (m, a)
9.          **for** b ∈ Σ **do**
10.             **if** q = NULL **then**
11.                 *table* (a, b) = NULL    //a and b are any characters
12.             **else**
13.                 *table* (a, b) = **factororacle** (q, b)
14.             **end**
15.         **end**
```

(Continued)

Algorithm 2 (continued)

```
16. Append t [n ... n + m − 1] AND p          //append text with pattern
17. j = m–1
18. end
19. while j < n do
20.      q = table (t[j], t [j − 1])
21.      while q = NULL do
22.           j = j + m–1                      // text for a shift s, i.e., j = s + m−1
23.           q = table (t[j], t [j−1])         //t[j] right most character
24.      i = j–2
25.     end
26.     while q ≠ NULL do
27.          q = factororacle (q, t[i])
28.          i = i–1
29.     end
30.     if i < j−m + 1 then
31.     Output(j)                //j is the position/state
32.      i = i + 1
33.      end
34.      j = j + i + m
35. end
```

3.4 Algorithm 3 for Integrity Layer

Algorithm 3 works similarly to algorithm 2 but returns the matched state when the pattern match occurs. For pattern matching, a loop is used as a condition to find the patterns in the current state. If a pattern match occurs at that state, that state is returned. Otherwise, the loop keeps running until it finds a pattern-matching condition or the keyword ends. The purpose of algorithm 3 is to find the patterns at each state.

Algorithm 3: Pattern Search using states

```
1.  Input:
2.  t: input array of p keywords
3.  n: text length
4.  p: pattern
5.  m: keyword length
6.  factororacle = FactorOracle(p)
7.  j = m–1
8.  while j < n do
9.       q = table (t[j], t [j−1])
10.       while q = NULL do
11.            j = j + m – 1                  // text for a shift s, i.e., j = s + m−1
12.            q = table(t[j], t [j−1])        // t[j] right most character
13.       i = j – 2
14.       end
```

(Continued)

Algorithm 3 (continued)

```
15.         while q ≠ NULL do
16.             q = factororacle(q, t[i])
17.             i = i – 1
18.         end
19.         if i < j–m + 1 then
20.             q = table (t[j], t[j\,–\,1]);
21.             for all pat matches at this state q
22.                 if a pattern match is found, then
23.                     return match state q
24.                 end
25.             end
26.         end
27.     j = j – 1;
28. end
```

The following section will present a detailed evaluation of these algorithms.

4 Experiments and Results

This section presents the experimental details to assess the proposed system's effectiveness using Snort and monitoring different attacks. Simulation is used to evaluate the impact of several attacks, including SYN Flooding, Port Scanning, and Session Hijacking attacks. We launched the live traffic in the simulation and used the NSL-KDD dataset. This dataset is a new version that provides different intrusion detection methods. For all the experiments, Intel i5 2.40 GHz, 16 GB RAM system is used by installing the virtual machine with Kali Linux and Ubuntu operating systems. We launched different web server attacks in the simulation and monitored the private networks. Snort NIDS is installed and configured on Ubuntu to monitor the private network. The hping3 and hamster tools are used in Kali Linux to generate traffic/packets. Wireshark is also used to sniff and observe the packets on the network. After algorithms integration in Snort, the code is refactored, and the updated Snort is used to detect intrusion and Zabbix to monitor Snort's performance. Zabbix is a network monitoring open-source program. The proposed NIDS is tested and validated in several real-world use cases, from low-traffic environments to high data rates. The three traffic scenarios ate used in the experiments. The rapid speed is achieved by generating 1,000 packets of 128 bytes and introducing them into the network at one-second intervals. Huge traffic is generated by using 5,000 network packets, every 128 bytes in size, and pushing them into the network at 15-s intervals; the high speed with enormous traffic generated by combining the above two cases, i.e., generating 5,000 network packets, every 128 bytes in size, and pushing them into the network at one-second intervals.

The attacks are launched from the attacker's machine. Initially, the hping3 tool is used to launch a syn flooding attack. Then, randomizing the source, hping3 bombards the victim server with 128-byte ICMP packets from a randomized source. The attacker then initiated a port scanning attack with the hping3 tool by specifying a range of reserved port numbers for victim servers. In the end, a syn flood and session hijacking assault in which the victim server is flooded with as much traffic as possible using hping3, and the hamster tool is used to execute a session hijacking attack. These attack scenarios are completed, each activity is recorded, and multiple alerts are generated. The system's average detection rate is determined to compare the results with the previous system for the NSL-KDD dataset and real traffic. Fig. 2 shows the comparison of the packet analysis rate, whereas Fig. 3 compares the packet loss rate.

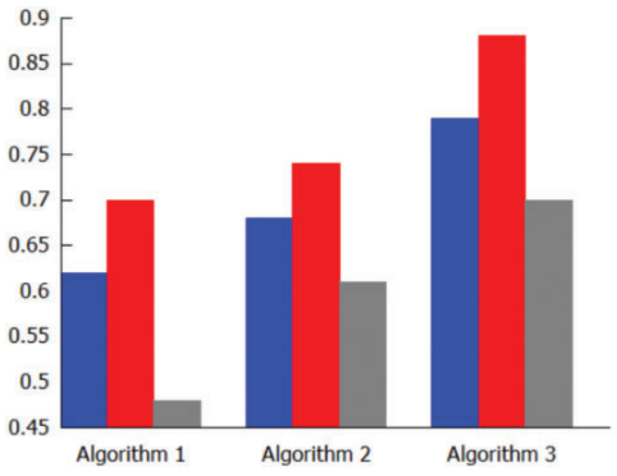

Figure 2: Comparison of packet analysis rate

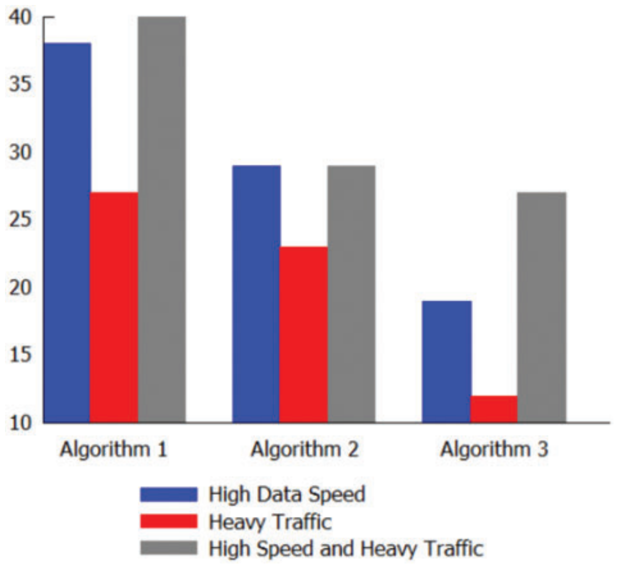

Figure 3: Comparison of packet drop rate

The accuracy is evaluated after assessing the performance of the algorithms based on the number of packets analyzed and the number of packets dropped under conditions of high data rate and heavy traffic. The Detection Rate (DR) and False Alarm Rate (FAR) are evaluated to compare the accuracy of the designed algorithms. From Fig. 3, we may have concluded that the DR of the proposed methods is promising, with algorithm 3 being superior to the AC-Std algorithm. With a lower FAR and a higher average pattern matching percentage, algorithm 3 handled massive traffic efficiently. Moreover, it selects Algorithm 1 and Algorithm 2, and it is evident from the statistical analysis that Algorithm 2 also performs well. Fig. 4 depicts a comparison between DR and FAR.

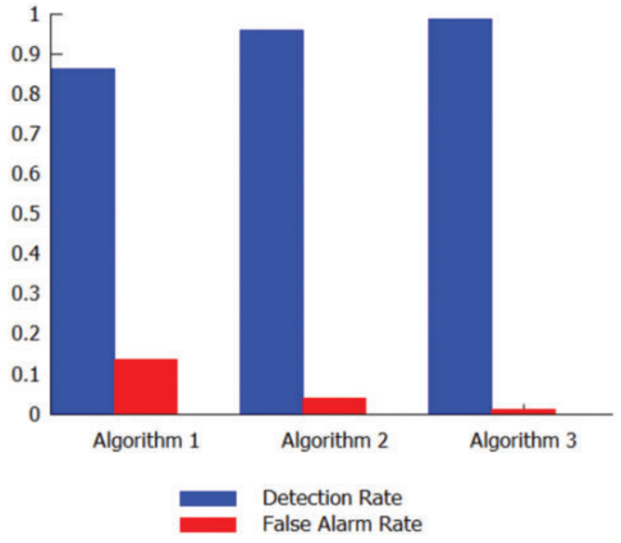

Figure 4: Comparison of DR and FAR

It is determined that the algorithms performed well based on the number of packets examined and the number of packets dropped under live conditions of high data rate and heavy traffic. The algorithm's accuracy is also determined by analyzing the DR and FAR. Then, it is compared to the existing solution [31]. The DR and FAR are calculated to assess the accuracy of the designed algorithms. A network IDS with a high detection rate and a low number of false alarms is considered a superior system. Fig. 4 demonstrates that the system's detection rate is encouraging compared to the current system for live traffic. The proposed system has a higher detection rate than the current one. In addition, the proposed system has fewer false alarms, which increases its efficiency. The relationship between DR and FAR for real traffic is depicted in Fig. 5.

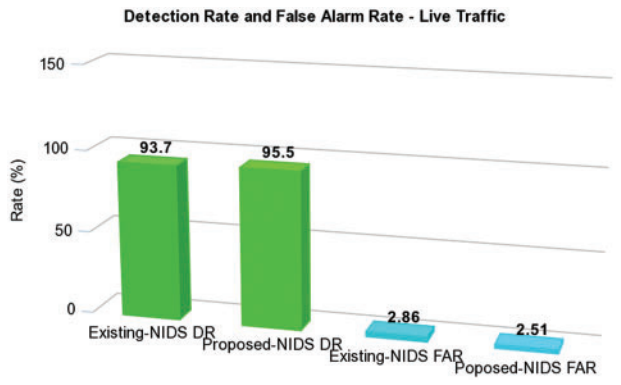

Figure 5: Comparison of DR and FAR for live traffic

The proposed algorithms are compared to the performance of existing methods using the NSL-KDD dataset and an experiment with benign and the most recent widely used attacks to evaluate the proposed network IDS. The Test+ file includes a comprehensive collection of assault records. DoS, U2R, R2L, Probe, and regular traffic scenarios are included in this data collection. This Test+ file contains a total of 22544 attack records. This data set is extracted to evaluate 7458 DoS records, 200

U2R records, 2754 R2L records, 2421 Probe records, and 9711 Normal traffic records. Based on the detection and false alarm rates for the NSL-KDD dataset, the proposed network IDS has a higher detection rate and fewer false alarms than the current NIDS. The NIDS has performed better with a higher detection rate than its predecessor. In addition, the rate of false alerts is reduced, which increases the system's efficiency. Fig. 6 compares the NSL-KDD Dataset's DR and FAR.

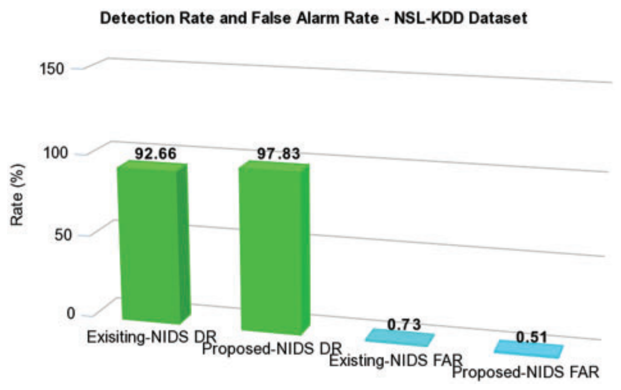

Figure 6: Comparison of DR and FAR for NSL-KDD dataset

Enhanced algorithms show potential in detecting and preventing threats like DoS and port scanning, with a low false positive rate. The system outperforms the current setup in every way. Reducing the number of packets lost during actual traffic is the primary obstacle.

Algorithms 1 and 2 can handle around 70%–75% of incoming traffic at full speed, while the rest of the tested algorithms handle over 85%. The high volume of incoming traffic caused an unacceptable rate of dropped packets, over 20%. Yet, algorithm 3 fared exceptionally well across all three tests in this crucial setting. This method competes with the built-in AC-Std algorithm to reduce packet loss and focuses on the most important test scenario with a high data rate and heavy traffic.

5 Conclusion

The expanding use of the internet and e-commerce applications makes objects more susceptible to various attacks. NIDS systems are employed to detect such assaults. Any IDS type, such as signature-based or anomaly-based, can be applied to the network and host system. This paper examined the attacks and existing NIDS systems for the prevention and how existing systems utilized methodologies and strategies. This work also presented the enhanced version of BOM techniques to provide an efficient layer-based NIDS architecture. These algorithms are upgraded using an extended BOM algorithm and incorporated into the proposed NIDS for improved pattern matching. The primary rationale for adding these algorithms is to manage high-volume and high-speed traffic. The performance evaluation is conducted using the Snort. The outcomes demonstrated that the enhanced algorithms based on extended BOM worked better than the original BOM. The traditional BOM algorithm has a 1.8% detection rate and 0.35% fewer false alarms for live traffic, whereas the extended BOM algorithm has a 5.17% higher detection rate and 0.22% less false alarms. The experimental results indicated that the functionality of the modified algorithms is adequate for DoS, Port Scanning, U2R, and R2R attacks in both real traffic and with the NSL-KDD dataset. Extended backward oracle matching is particularly effective for larger patterns and shorter strings. In the future, we will test the system with other datasets to evaluate the overall performance of the system.

Acknowledgement: This research is funded by Research Supporting Project Number (RSPD2023R553, King Saud University, Riyadh, Saudi Arabia

Funding Statement: This research is funded by Research Supporting Project Number (RSPD2023R553, King Saud University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Nagaraja and T. S. Kumar, "An extensive survey on intrusion detection-past, present, future," in *Presented at the Proc. of the Fourth Int. Conf. on Engineering & MIS 2018*, Istanbul, Turkey, 2018. https://doi.org/10.1145/3234698.3234743

[2] R. W. Anwar, M. Bakhtiari, A. Zainal and K. N. Qureshi, "Malicious node detection through trust aware routing in wireless wensor networks," *Journal of Theoretical Applied Information Technology*, vol. 74, no. 1, pp. 88–92, 2015.

[3] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Computer Networks*, vol. 136, pp. 37–50, 2018.

[4] M. J. Nazar, A. Alhudhaif, K. N. Qureshi, S. Iqbal and G. Jeon, "Signature and flow statistics based anomaly detection system in software-defined networking for 6G internet of things network," *International Journal of System Assurance Engineering and Management*, vol. 14, pp. 1–11, 2021.

[5] C. Xu, J. Shen, X. Du and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[6] F. Salo, M. Injadat, A. B. Nassif, A. Shami and A. Essex, "Data mining techniques in intrusion detection systems: A systematic literature review," *IEEE Access*, vol. 6, pp. 56046–56058, 2018.

[7] K. N. Qureshi, A. Alhudhaif, S. W. Haider, S. Majeed and G. Jeon, "Secure data communication for wireless mobile nodes in intelligent transportation systems," *Microprocessors and Microsystems*, vol. 90, pp. 104501, 2022. https://doi.org/10.1016/j.micpro.2022.104501

[8] A. Aldweesh, A. Derhab and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, pp. 105124, 2020.

[9] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Computer Communications*, vol. 49, pp. 1–17, 2014.

[10] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–35, 2019.

[11] K. N. Qureshi, G. Jeon and F. Piccialli, "Anomaly detection and trust authority in artificial intelligence and cloud computing," *Computer Networks*, vol. 184, pp. 107647, 2020.

[12] E. Viegas, A. O. Santin, A. Franca, R. Jasinski, V. A. Pedroni *et al.,* "Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 163–177, 2016.

[13] L. Villalba, A. Orozco and J. Vidal, "Anomaly-based network intrusion detection system," *IEEE Latin America Transactions*, vol. 13, no. 3, pp. 850–855, 2015.

[14] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Applied Soft Computing*, vol. 92, pp. 106301, 2020.

[15] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat *et al.,* "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[16] N. Moustafa, J. Hu and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *Journal of Network Computer Applications*, vol. 128, pp. 33–55, 2019.

[17] M. Liu, Z. Xue, X. Xu, C. Zhong and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–36, 2018.

[18] M. A. Ambusaidi, X. He, P. Nanda and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[19] A. Nisioti, A. Mylonas, P. D. Yoo and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3369–3388, 2018.

[20] M. Rashid, M. Imran and A. R. Jafri, "Exploration of hardware architectures for string matching algorithms in network intrusion detection systems," in *Proc. of the 11th Int. Conf. on Advances in Information Technology*, USA, pp. 1–7, 2020.

[21] C. L. Lee and T. H. Yang, "A flexible pattern-matching algorithm for network intrusion detection systems using multi-core processors," *Algorithms*, vol. 10, no. 2, pp. 58, 2017.

[22] K. Azarudeen, G. V. Chakkaravarthy, P. Murugiah and S. Kharthikeyan, "A novel approach for pattern string matching in intrusion detection system," in *Journal of Physics: Conf. Series*, IOP Publishing, Coimbatore, India, vol. 1916, no. 1, pp. 012007, 2021.

[23] M. Fisk and G. Varghese, "Applying fast string matching to intrusion detection," *Los Alamos National Lab NM*, pp. 1–21, 2002.

[24] R. Gaddam and M. Nandhini, "An analysis of various snort based techniques to detect and prevent intrusions in networks proposal with code refactoring snort tool in Kali Linux environment," in *2017 IEEE Int. Conf. on Inventive Communication and Computational Technologies (ICICCT)*, Coimbatore, India, pp. 10–15, 2017.

[25] R. Gaddam and M. Nandhini, "Prospective backward oracle matching algorithm for network intrusion detection system," in *2017 2nd IEEE Int. Conf. on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, pp. 1143–1148, 2017.

[26] E. M. Chakir, Y. I. Khamlichi and M. Moughit, "Handling alerts for intrusion detection system using stateful pattern matching," in *2016 4th IEEE Int. Colloquium on Information Science and Technology (CiSt)*, Tangier, Morocco, pp. 139–144, 2016.

[27] C. Stylianopoulos, M. Almgren, O. Landsiedel and M. Papatriantafilou, "Multiple pattern matching for network security applications: Acceleration through vectorization," *Journal of Parallel Distributed Computing*, vol. 137, pp. 34–52, 2020.

[28] S. Lee and T. T. Phan, "Enhancement of Wu-Manber multi-pattern matching algorithm for intrusion detection system," in *Context-Aware Systems and Applications, and Nature of Computation and Communication*, Springer: ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018, pp. 69–78, 2017.

[29] I. Obeidat and M. AlZubi, "Developing a faster pattern matching algorithms for intrusion detection system," *International Journal of Computing*, vol. 18, no. 3, pp. 278, 2019.

[30] O. Folorunso, F. E. Ayo and Y. Babalola, "Ca-NIDS: A network intrusion detection system using combinatorial algorithm approach," *Journal of Information Privacy Security*, vol. 12, no. 4, pp. 181–196, 2016.

[31] M. Karimov, K. Tashev and S. Rustamova, "Application of the Aho-Corasick algorithm to create a network intrusion detection system," in *2020 Int. Conf. on Information Science and Communications Technologies (ICISCT)*, Tashkent, Uzbekistan, pp. 1–5, 2020.

[32] V. Dagar, V. Prakash and T. Bhatia, "Analysis of pattern matching algorithms in network intrusion detection systems," in *2016 2nd Int. Conf. on Advances in Computing, Communication, & Automation (ICACCA) (Fall)*, Bareilly, India, pp. 1–5, 2016.

[33] D. Li, L. Deng, M. Lee and H. Wang, "IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning," *International Journal of Information Management*, vol. 49, pp. 533–45, 2019.

[34] S. Lee and T. T. Phan, "Enhancement of Wu-Manber multi-pattern matching algorithm for intrusion detection system," in *Context-Aware Systems and Applications, and Nature of Computation and Communication: 6th Int. Conf., ICCASA 2017, and 3rd Int. Conf., ICTCC*, Tam Ky, Vietnam, pp. 69–78, 2017.

[35] S. Vakili, J. P. Langlois, B. Boughzala and Y. Savaria, "Memory-efficient string matching for intrusion detection systems using a high-precision pattern grouping algorithm," in *Proc. of the 2016 Symp. on Architectures for Networking and Communications Systems*, pp. 37–42, 2016.

[36] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee and H. Karimipour, "Cyber intrusion detection by combined feature selection algorithm," *Journal of Information Security and Applications*, vol. 44, pp. 80–88, 2019.

[37] D. Pao, W. Lin and B. Liu, "A memory-efficient pipelined implementation of the aho-corasick string-matching algorithm," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 7, no. 2, pp. 1–27, 2010.

[38] N. Butt, A. Shahid, K. N. Qureshi, S. Haider, A. O. Ibrahim *et al.,* "Intelligent deep learning for anomaly-based intrusion detection in IoT smart home networks," *Mathematics*, vol. 10, no. 23, pp. 4598, 2022.