A Regeneration Model for Mitigation Against Attacks on HTTP Servers for Mobile Wireless Networks

Original Scientific Paper

Abiodun Akinwale
A. Akinwale Obafemi Awolowo University, Department of Computer Science & Engineering
Ile-Ife, Osun State, Nigeria, logitronics@yahoo.com

Emmanuel Olajubu
E. A. Olajubu Obafemi Awolowo University, Department of Computer Science & Engineering
Ile-Ife, Osun State, Nigeria, emmolajubu@oauife.edu.ng

Adesola Aderounmu
G. A. Aderounmu Obafemi Awolowo University, Department of Computer Science & Engineering
Ile-Ife, Osun State, Nigeria, gaderun@oauife.edu.ng

Abstract – *The widespread expansion of the internet has fueled a global surge in the utilization of various online transactions, with a significant portion of such services operating on mobile web platforms. Simultaneously, the deployment of innovative Mobile Ad Hoc Network (MANET) technologies and mobile applications has grown as solutions for diverse tasks. Unfortunately, this progress has attracted the attention of hackers, who continually devise new strategies to exploit the vulnerabilities inherent in mobile networks. This study aims to address the escalating challenges posed by cyber threats in the era of widespread internet expansion, particularly focusing on securing mobile web platforms against sophisticated attacks such as Structured Query Language Injection (SQLi) for web-based database solutions and Denial of Service (DoS/DDoS) for various applications. In response to the identified vulnerabilities, this paper proposes an HTTP regeneration (HReg) model that not only detects various cyber-attacks but also ensures the uninterrupted provision of critical services during such incidents. The model introduces an innovative regeneration algorithm capable of scanning both the connection channel and web application to detect attacks, creating survivable connections within the underlying TCP engine to replace compromised ones during an ongoing attack. In simulated environments using OMNeT++, where the server is subjected to attacks, the experimental results demonstrate the efficacy of the model. The response and performance metrics, including throughput (73%), delivery ratio (68.8%), delay (3s), and network load, showcase the model's ability to detect and neutralize attacks. A comparison with state-of-the-art approaches highlights the superior performance of the regeneration model, attributed to its additional survivability layer. While the regeneration model proves robust in simulated environments, its real-world application may encounter limitations. Future research should explore these limitations to enhance the practical applicability of the proposed model. The proposed HReg model's resilient performance under attack conditions ensures the survivability of web-based applications. This innovative approach offers practical implications for securing mobile web platforms, providing continuous delivery of critical services even in the face of persistent and evolving cyber threats. This research addresses a significant gap in existing efforts by not only focusing on attack detection but also emphasizing the development of a survivable TCP connection for HTTP servers during attacks. The introduced regeneration model stands out for its unique ability to maintain service continuity, showcasing originality in the approach to cybersecurity in the context of web-based applications.*

Keywords: *Cyberattacks, DoS, HTTP, MANET, Network Regeneration, OMNeT++, Protocol, Security, SQLi, TCP, TCP/IP*

Received: December 27, 2023; Received in revised form: March 22, 2024; Accepted: March 25, 2024

1. INTRODUCTION

The exponential growth of information and communication technology (ICT) dependence in sectors like business, education, and governance has led to a significant increase in global online transactions.

A significant portion of these transactions now take place over mobile networks, with Mobile Ad Hoc Networks (MANETs) emerging as a crucial representative of such distributed systems or networks. This makes MANETs attractive targets for cybercriminals due to perceived vulnerabilities [1].

Mobile Ad Hoc Networks (MANETs) are a type of mobile wireless network where mobile devices communicate with each other without the need for a fixed infrastructure or centralized administration. In Mobile Ad Hoc Networks (MANETs), the network topology undergoes constant fluctuations due to nodes' movements, additions, and departures. This dynamic characteristic renders MANETs well-suited for scenarios lacking preexisting infrastructure, such as disaster areas, military operations, or temporary events. Additionally, MANETs operate without centralized control, empowering each node to manage its data routing and participate in the network's self-organization, enhancing adaptability and resilience.

Furthermore, mobile devices within MANETs often face resource constraints, including limited battery power, processing capabilities, and bandwidth. Hence, efficient resource utilization and energy-aware protocols become imperative in MANET design. Moreover, communication between nodes typically occurs through multi-hop routing in the absence of fixed infrastructure, wherein data is relayed through intermediate nodes to reach its intended destination. This multi-hop communication strategy enables effective data transmission despite the dynamic topology and resource limitations, underscoring the importance of protocols designed to efficiently manage changes in network topology, node mobility, link failures, and new node arrivals to ensure smooth MANET operation.

According to Kaspersky's 2023 attack statistics bulletin [2], the daily rate of attacks exceeded 400,000, marking a 3% rise compared to 2022 figures. Correspondingly, [3] highlighted economic losses of up to $100 billion annually due to heightened hacking activities in the United States, while [4] emphasized the importance of users embracing improved cybersecurity strategies, as per their survey on the web threats landscape.

Most of the reported activities happen at the application layer of the TCP/IP protocol stack since this layer supplies essential services for internet users [5]. Hypertext transfer protocol (HTTP) for example, runs on a client/server basis and defines the rules for applications running on diverse systems on how to pass messages to each other. It equally specifies the types, syntax, and semantics needed for processing and responding to requested messages. A node that needs some services is the client and the supply node is the server. Port-to-port connection for different services or processes is provided by the underlying TCP protocol that supplies persistent connectivity until all data segments are delivered to the client after which the TCP connection working in the background ends. Many protocols drive various internet services in this layer such as file transmission protocol (FTP), simple mail transmission protocol (SMTP), post office protocol (POP), domain name service (DNS) - a support service for other applications, Telnet, trivial file transmission protocol (TFTP) and HTTP [6]. HTTP is the most popular, most important and the most widely attacked protocol in the application layer and the subject of this study. It is almost synonymous with the World Wide Web.

The two common attacks on HTTP are DoS, with its variant called DDoS [7], and SQL injection attack [8]. The DoS and DDoS attacks on HTTP are very difficult to differentiate from valid traffic because they use standard URL requests that bear subtle similarities to legitimate requests. This makes them one of the most advanced non-vulnerability security challenges facing servers and applications today. Traditional rate-based detection is ineffective in detecting HTTP flood attacks since traffic volume in HTTP floods is often under detection thresholds or low sensitivity.

On the other hand, the Structure Query Language injection (SQLi) is an attack where a malicious query is posted to a database to manipulate the backend system to access unauthorized data. This attack manifests in various ways and this makes it very difficult to track and prevent. This has allowed hackers to gain access to sensitive unauthorized data in large databases. The common and simplest SQLi attacks are mostly based on the UNION operator. The UNION operator combines two query statements to fetch data from a database. Microsoft SQL servers are particularly vulnerable to error-based SQLi attacks. In this scenario, the attackers trick applications into flagging error messages that contain the data of interest to them. In a blind SQLi attack, the intruders post different types of SQL queries that make the database respond either TRUE or FALSE. The attackers gain information from the database response and use it to modify the SQL query before re-launching the attack. There are other ways of injecting malicious code into a database, such as using HTTP headers. Headers with random SQL queries will inject malicious queries into the database. The second degree of SQLi attacks are more complex and difficult to detect in that the malicious query can be dormant in the system for a long time which may even be considered harmless but when triggered can be very malicious [9].

Existing MANET Intrusion Detection Systems (IDSs) typically generate alerts or apply automated blocking in response to attacks. However, these approaches are insufficient against IP address spoofing, and designing efficient detectors with low computational overhead remains a concern. This work provides mitigation beyond blocking and introduces resilience for survivability with a simple low-power regeneration algorithm necessary to ensure service delivery continuity despite an attack incidence.

This paper introduces a protocol-based intrusion detection system with a regeneration algorithm to mitigate DoS and SQL injection attacks on HTTP servers in MANETs. The regeneration model, inspired by the biological concept of regeneration, replaces compromised TCP connections for HTTP services, ensuring resilience, recovery, adaptability, and survivability. This innovative model allows HTTP servers to continue delivering services without succumbing to attacks. Notably, this work represents the first instance of a regeneration model developed for the HTTP protocol in MANET networks.

The rest of the paper is arranged as follows: the next section presents the review of literature on detection systems for DoS, DDoS, and SQL injection attacks on the operations of HTTP. The system methodology is presented in section 3 while the results and conclusions are in sections 4 and 5, respectively.

2. RELATED WORKS

This section reviews current models deployed against DoS and its variants and SQL injection.

In [10], a two-sided algorithm (static and dynamic) was proposed to resolve SQL injection attacks. The static side performs the analysis of SQL query statements to discover any malicious code inserted into the query on the web applications. This is to ensure that user input is not compromised; this prevents any form of SQL injection attack. The second side refers to the dynamic nature of the algorithm which expunges any malicious code detected by the static algorithm. The advantage of the system is that it does not require any web application alteration. Nevertheless, any malicious code found must be manually expunged by the developer/administrator. The model has the same performance with BEBSSARI [11]. The advantage of the model over BEBSSARI is that while BEBSSARI is partially automated, the proposed model is fully automated.

Another machine learning algorithms proposal authored by [12] demonstrated the use of a sliding window which optimizes the detection accuracy by dynamically obtaining the segment for intrusion detection. The paper applied the Burstiness statistical measure, Sharon entropy, and Quantile Cumulative probability threshold algorithms for the calculation and optimization of the smallest window. The method used in this model is consistent with [13] and [14]. The study employed a sliding window algorithm based on morphological fractal dimension, the result presented showed that there is improved detection of DDoS attacks with an accuracy of 99.30%. Ref. [15] proposed a security model with two ways of operation against cyberbioattacks. Cyberbiotattacks use biologically engineered bacteria to implement distributed denial-of-service. The first part of the model produces molecules that can block the traffic sent by cyberbioattacks, this is called quorum quenching, while the second called the amplification approach, radiates molecules with the capacity to increase the proportion required to build more biofilm structure. The performance of the model was measured by creating a dynamic scenario that allows DDoS traffic to be generated and sent as a cyberbioattack. The result presented showed that the model reduced the influence of cyberbioattack. The quorum quenching scheme performs better than the amplification approach concerning attack detection, but the amplification approach familiarizes better in attack configurations with the biofilm formation. One of the major threats to 5G communication network reliability and quality of service is DoS. A model was developed by [16] to mitigate the impact of DoS on the 5G communication network so that proper quality of service can be experienced by the subscribers. The model deployed a differential flow management scheme to tackle the menace of DoS on the network. The traffic was classified as continuous or discrete flow. The discrete flow was used as a sub-optimal differential problem, the model aimed at converging the anomalous detection time and reformulating the allocation of resources as a continuous flow based on the leftover. To retain the response time and transmission rate of the user equipment, the flow was modeled continuously on service and transmission intervals. The results presented by the authors were consistent as intrusion detection time was minimized, the delivery ratio was maximized, and the response time of the system was retained. Information security on the web is becoming a herculean task as hackers have devised various attack schemes to access unauthorized information and disrupt opponent networks to gain some reputation.

Cyber-attack through SQLi is one of the most difficult attacks to trace and the detection often takes a long period. Ref. [17] designed an SQLi scanning model using the MySQL injector tool that can easily conduct penetration tests on a PHP-based website. The tool contains four developmental phases in which phase one is called Inception. This gathers the vulnerabilities of the website while the second phase called elaboration translates all the vulnerability requirements gathered in the first phase into software design models. The construction phase or phase three is where a prototype is developed based on the requirements and design while the transition phase is the full development of the system to testing. In the same stride, [18] worked on the vulnerability of wireless ad-hoc networks to DoS attacks and identified that attack drains the battery power of nodes, making the resources of the network unavailable resulting in degrading the network performance. The work reviews various techniques such as packet leash, spread spectrum, energy weight monitoring system, and lightweight secured mechanism among several others employed to counter DoS attacks. These schemes have not been able to effectively checkmate DoS attacks. Refs. [19] and [20] opined that DDoS attack is the main obstruction in the success rate of Software Define Networks (SDN), despite several proposals, mitigation of this type of attack remains very difficult. A detection proposal was modeled and implemented. The system consists of a programmable network monitoring kit and a control structure that is very flexible for fast attack detection. This architecture of the attack detection system is based on a statistical function that can address the flooding problem. The simulation results presented show that the model was effective in addressing the DDoS attack.

Ref. [21] conceptualized the SQLi attack as a Markovian decision process with a reinforcement learning problem. The reinforcement learning agent was developed to perform SQL injection. The training of the system was enacted in such a way as to have a generic guideline that affects SQL injection attacks and not specific code injection to the web. In the analysis, the results outlined the agent's learning process and the complexity of the algorithm. The developed agent succeeded in deep penetration testing. The overall result of the study demonstrated the possibility and weakness of using reinforcement learning in the security environment. A machine learning algorithm was proposed by [22] which is to monitor the external aggressor and the VM against DDoS attack. The model comprises of data clustering technique through a feature selection algorithm by principal component analysis. The Agglomerative Clustering and K-means algorithm was used in the model. The results showed that the model had 0.9130 on the adjusted Rand Index metric. The model was able to effectively handle the external and internal traffic against DDoS on a cloud computing platform.

Many research outputs have attempted to proffer solutions to the susceptibility of HTTP protocol stack to SQLi and DDoS attacks. Ref. [23] in their work responded to the growing reliance on cyber infrastructures and the escalating frequency of cyber-attacks with the continuous demand for the development of advanced protection mechanisms. The study focuses on creating and implementing a Deep Neural Network model for the detection of intrusions in computer networks. Techniques such as SMOTE and Random Sampling were applied to address data imbalance in the CICIDS 2017 dataset. The entire experiment was conducted on a single Jupyter notebook in the Google Colaboratory environment, importing and implementing relevant software libraries like Seaborn, Pandas, Matplotlib, Keras, and TensorFlow as needed. The results revealed excellent performance, with a 99.68% accuracy score and a loss of 0.0102 in predicting attacks with the CICIDS 2017 dataset.

In [24], the researchers aim to enhance automated intrusion detection by developing a highly accurate classifier with minimal false alarms. Their motivation is to address the challenges of high dimensionality in intrusion detection and improve classifier performance for more accurate and efficient intrusion detection. Experiments were conducted using the NSL-KDD dataset, initially employing the entire feature set, where the J48 tree achieves the highest reported accuracy of 79.1%. The study explores Random Projection and PCA to enhance classifier performance, with Random Projection proving more time-efficient and achieving a notable accuracy improvement to 82.0%. The research concludes that random projection is effective in improving intrusion detection accuracy while reducing training time, contributing valuable insights to the cybersecurity field. The study in [25] focused on SQL injection (SQLi) attacks on websites, using tools like Whois, SSL Scan, Nmap, OWASP Zap, and SQL Map. Researchers identified and mitigated vulnerabilities on the web server, discovering 14 issues with OWASP Zap, categorized by severity. SQL Map successfully accessed the database and username. The research aims to recommend firewall installation for mitigating SQLi risks, providing a structured methodology for addressing vulnerabilities in web servers and enhancing data security. According to [26], DDoS attacks pose a significant threat to the web due to the rise in web-based transactions and Internet application services. Countering these attacks has become more challenging with attackers utilizing vast resources and techniques. Unlike traditional network-layer attacks, application-layer DDoS attacks, particularly slowloris attacks, can be more effective by inundating victim resources with legitimate HTTP requests. The paper proposes a slowloris attack detection method based on an adaptive timeout-based approach, comprising a suspect determination module and an attacker verification module. The experimental results demonstrate the algorithm's efficacy in detecting attackers with low false alarms and high accuracy before consuming all resources.

A frequently observed constraint noted across papers [10-26] is the lack of mechanisms ensuring uninterrupted service delivery during attacks despite the prevalent utilization of machine learning algorithms in the reviewed methods for highly accurate attack detection. This research enhances the functionality of the HTTP protocol through the introduction of a novel model termed HReg (HTTP Regeneration). HReg not only identifies but also withstands SQL injection and DoS attacks targeting HTTP servers. It achieves this through the incorporation of network-based attack scanners, protocol surveillance detection for protocol-specific application-based attacks, and a regeneration-based technique aimed at ensuring system survivability.

2.1. HTTP OPERATIONS AND TRANSACTIONS

HTTP defines the protocol for retrieving various types of messages, including text, video, audio, graphics, and other files, from World Wide Web servers through client browsers. Clients initiate web page requests, and servers respond with the requested information. When HTTP clients seek web pages from server hosts, they utilize resource identifiers, with the most common being the Uniform Resource Locator (URL), also known as the web address. The URL follows a standard format: protocol://host:port/path. Often, specifying protocol://host is sufficient to connect to a resource, allowing users to navigate to desired paths. For instance, http://www.yahoo.com features the host or domain name www.yahoo.com, and HTTP utilizes TCP port 80 for its connections. While HTTPS is a more secure protocol and industry preference is shifting towards it, the proposed model's operations are equally applicable to it, as it is not entirely immune to all attacks.

HTTP transactions follow a straightforward process:

- The client initiates a request using methods such as GET, POST, OPTION, HEAD, PUT, TRACE, etc. For instance, GET is used for static content requests, while POST is employed for dynamic content like databases. The request method is accompanied by a header specifying the desired response format from the server.
- The server responds with a reply typically commencing with a status code (e.g., 100, 101, 200, 400, 401, 501, etc.), followed by the header and the message in case of successful retrieval. The client browser then displays the response contents using Hypertext Markup Language (HTML).

2.2. ATTACKS ON HTTP PROTOCOL

HTTP attacks are usually carried out through code injection and result in snooping/sniffing, modification, masquerading, and denial of service attacks. The attack is most effective when it forces the server or application to distribute the maximum resources possible in response to each single request. Thus, the perpetrator will generally aim to inundate the server or application with multiple process-intensive requests. For this reason, HTTP flood attacks using POST requests tend to be the most resource-effective from the attacker's perspective; as POST requests may include parameters that trigger complex server-side processing. On the other hand, HTTP GET-based attacks are simpler to create, and can more effectively scale in a botnet scenario.

3. METHODOLOGY

In the context of this work, regeneration refers to the process through which networks under attack replace compromised TCP connections underlying HTTP with replicas from an available pool of sockets. The replacement socket is an exact copy of the compromised or lost one, and this substitution occurs nearly instantaneously.

Drawing inspiration from bio-defense mechanisms, cell regeneration in organisms serves as a captivating biological process allowing them to regenerate body parts as a defense against predators. For instance, octopuses exhibit an impressive ability to regenerate lost body parts, including arms and their central nervous system. Similarly, in the proposed model, the replication of new arms (connections) promptly replaces lost ones from an internal pool of cells. While regeneration is advantageous for defense, it comes at a cost to the prey (network resources). HTTP connections, akin to cells, require replenishment or promotion of regeneration to restore damaged connections during an attack. This capability ensures that networks continue delivering on their stated missions despite damaged links or nodes, maintaining service quality without stochastic fluctuations. This aspect is crucial in network design to ensure high-quality service (QoS) or secured socket connections in the HTTP protocol, mitigating the impact of intruders on link quality or connections. The ability to sustain good service delivery despite attacks determines the survivability of the network, which is the primary focus of this work.

It is worth noting that while this study focuses on HTTP servers, the transport engine is the TCP protocol, ensuring the connection between the server and the client. In socket programming, the process of setting up a TCP socket on the server side involves creating a socket, binding it to an address, listening for connections, accepting a connection, sending and receiving data, and closing the connection.

3.1. REGENERATION MODEL FORMULATION

In this work, the regeneration mechanism for the HTTP protocol is initially modeled to achieve optimal link or connection replacement from the network pool of TCP sockets providing connection service. The modeling of TCP socket regeneration in a network entails the consideration of factors including the generation of new sockets, the closure of existing sockets, and the potential influence of network events like the velocity of mobile nodes.

Let S(t) be the size of active TCP sockets at time t. The regeneration of TCP sockets can be modeled as follows:

$$\frac{dS}{dt} = \alpha - \beta S - \gamma E \qquad (1)$$

Here:

- α represents the rate at which new TCP sockets are created or regenerated.
- β represents the rate at which existing TCP sockets close or become inactive.
- γ represents the impact of velocity (denoted by E) that might affect the TCP socket number.

This model assumes a constant regeneration rate α and a linear dependence on the current number or population of sockets. The term $-\beta S$ represents the closing or inactivation of existing sockets, and $-\gamma E$ accounts for external events affecting the regeneration process.

Next, consideration is given to the impact of attacks like SQL injection (SQLi) on HTTP server connections. The regeneration of underlying TCP sockets in the presence of SQLi attacks necessitates an examination of their impact on socket size.

The ensuing differential equation, which integrates the effect of SQLi attacks, is presented as follows:

If $S(t)$ is the number of active TCP sockets at time t, and $A(t)$ is the size of SQL injection attacks. The regeneration of TCP sockets in the presence of SQLi attacks is as follows:

$$\frac{dS}{dt} = \alpha - \beta S - \gamma ES - \delta A \qquad (2)$$

$$\frac{dA}{dt} = \phi A - \theta SA \qquad (3)$$

Where:

- α represents the rate at which new TCP sockets are created or regenerated.
- β represents the rate at which existing TCP sockets close or become inactive.
- γ represents the impact of external events (denoted by E) on the existing TCP sockets.
- δ represents the impact of SQL injection attacks (A) on the TCP socket population.
- ϕ represents the rate of SQL injection attacks.
- θ represents the interaction strength between the TCP sockets and the SQL injection attacks.
- The term $-\gamma ES$ in the equation for dS/dt models the effect of external events such as node movement on the existing sockets, and δA models the impact of SQL injection attacks on the socket population. The equation for dA/dt describes the dynamics of the SQL injection attacks, with ϕ being the rate of attacks and θS representing the interaction strength between the SQL injection attacks and the existing TCP sockets.

The regeneration of TCP sockets in the presence of DDoS attacks is as follows:

$$\frac{dS}{dt} = \alpha - \beta S - \gamma ES - \delta A^2 \quad (4)$$

$$\frac{dA}{dt} = \phi A - \theta SA \quad (5)$$

Where:

- α represents the rate at which new TCP sockets are created or regenerated.
- β represents the rate at which existing TCP sockets close or become inactive.
- γ represents the impact of external events (denoted by E) on the existing TCP sockets.
- δ represents the impact of SQL injection attacks (A) on the TCP socket population. The square term A^2 indicates a potential nonlinear impact distributed attack.
- ϕ represents the rate of DDoS attacks.
- θ represents the interaction strength between the TCP sockets and the DDoS attacks.

3.1.1. ATTACK PROBABILITIES

Given the stochastic nature of attack scenarios, probabilities of attack neutralization are now incorporated into the differential equations mentioned earlier.

Let $P_n(t)$ be the probability of neutralizing the SQL injection attack, and $P_o(t)$ be the probability of the attack overcoming the defense. The regeneration of TCP sockets in the presence of SQLi attacks with probabilities of attack neutralization can be modeled as follows:

$$\frac{dS}{dt} = \alpha - \beta S - \gamma ES - \delta A2Po \quad (5)$$

$$\frac{dA}{dt} = \phi A - \theta SAPn \quad (6)$$

$$\frac{dP_n}{dt} = \rho Pn(1 - Po) \quad (7)$$

$$\frac{dP_o}{dx} = \sigma Pn - \omega Po \quad (8)$$

Here:

- α represents the rate at which new TCP sockets are created or regenerated.
- β represents the rate at which existing TCP sockets close or become inactive.
- γ represents the impact of mobility (denoted by E) on the existing TCP sockets.
- δ represents the impact of SQL injection attacks (A) on the TCP socket population, with P_o indicating the probability of the attack overcoming the defense.
- ϕ represents the rate of SQL injection attacks.
- θ represents the interaction strength between the TCP sockets and the SQL injection attacks, with P_n indicating the probability of attack neutralization.
- ρ is the rate at which the probability of neutralizing the attack increases.
- σ is the rate at which the probability of the attack overcoming the defense increases.
- ω is the rate at which the probability of the attack overcoming the defense decreases.

The terms $-\delta A2P_o$ in the equation for dS/dt and θSAP_n in the equation for dA/dt introduce the probabilities P_o and P_n, respectively, influencing the dynamics of TCP socket regeneration in the presence of SQLi attacks. The term $-\gamma ES$ in the equation for dS/dt models the effect of external events such as node movement on the existing sockets, and $\delta A2$ models the impact of DDoS attacks on the socket population. The equation for dA/dt describes the dynamics of the DDoS attacks, with ϕ being the rate of attacks and θS representing the interaction strength between the DDoS attacks and the existing TCP sockets.

2.1.1. COST OF REGENERATION DEFENSE DEPLOYMENT

Deploying regeneration defense incurs a cost associated with actively neutralizing attacks, and it influences the decision-making process in the network dynamics. Incorporating the cost of attack neutralization into the equations involves introducing a cost term in the equation for TCP socket dynamics dS/dt.

The updated set of equations including cost becomes:

$$\frac{dS}{dt} = \alpha - \beta S - \gamma ES - \delta A2P_o - \chi P_n \quad (9)$$

$$\frac{dT}{dt} = \phi A - \theta SAP_n \quad (10)$$

$$\frac{dP_n}{dt} = \rho Pn(1 - Po) - \omega Pn \quad (11)$$

$$\frac{dP_o}{dx} = \sigma Pn - \omega Po \qquad (12)$$

In Equations (9) – (11):

χ represents the cost of attack neutralization. The term $-\chi P_n$ is subtracted from the TCP socket regeneration rate, reflecting the cost incurred in neutralizing attacks. This is essentially the product of the cost parameter (χ) and the probability of attack neutralization (P_n). Mathematically, the cost of attack neutralization (C_n) can be expressed as:

$$Cn = \chi P_n \qquad (13)$$

This equation reflects that the cost of attack neutralization is proportional to both the cost parameter (χ) and the probability of attack neutralization (P_n).

3.1.3. MODEL PARAMETRIZATION

Model parameters can be obtained by experiment, systems in production, or empirically. To ensure that attacks are always neutralized, parameters must be set to promote a high probability of attack neutralization and prevent the probability of the attack overcoming the defense (P_o) from becoming dominant. The following considerations are important to achieve this:

1. Set ρ to a high value: A high value for ρ in the equation $dP_n/dt = \rho P_n(1-P_o)$ will make the probability of attack neutralization (P_n) increases rapidly.
2. Set σ and ω to low values: Low values for σ and ω in the equation $dP_o/dt = \sigma P_n - \omega P_o$ will make the probability of the attack overcoming the defense (P_o) change slowly.
3. Ensure ϕ, the rate of SQL injection attacks is not too high: If the rate of attacks (ϕ) is very high, it might be challenging to neutralize all of them. A reasonable ϕ value, combined with a high ρ, will help maintain a high probability of neutralization.
4. Adjust the other parameters (α, β, γ, δ, θ): These parameters govern the dynamics of TCP socket regeneration and attack interactions. Adjust them based on your understanding of the system and the desired behavior.

3.2. INTRUSION DETECTION FOR HTTP PROTOCOL

HReg detection methodology entails the deployment of a scanner at each susceptible point within the HTTP protocol. The detection engine consists of two main modules: the protocol scanning module and the dynamic database updater module. Through continuous analysis of all traffic, the scanner identifies new attacks and maintains communication with the database module to verify signatures of known attacks. Upon detection of a novel attack, the database is promptly updated. The IDS scanners vigilantly monitor socket addresses (port + IP addresses) to effectively detect protocol-specific threats.

Each socket is associated with a port number, facilitating the recognition of the intended application by the TCP layer. Intrusions are identified when multiple GET/POST requests originate from the same IP address, indicating suspicious activity. Furthermore, alerts are triggered when fake URL commands or syntax anomalies are detected in other requests. Upon identifying an attack, the IDS system issues an alert, initiating the activation of countermeasures for regeneration.

3.3. REGENERATION MODEL ALGORITHM

Algorithm 1 delineates the series of steps involved in overseeing HTTP requests, executing intrusion detection procedures, and deploying necessary actions upon intrusion detection. The procedure encompasses interactions between the HTTP client and server, detection of potential intrusions through scanning, and the restoration of connections following intrusion detection alerts. Upon activation of an intrusion alert, the socket linked with the intruder is blocked and placed on a blacklist. Subsequently, a new 'create socket' instruction is executed, as illustrated in Fig. 1.

The algorithm is designed based on the behavior of an HTTP connection to a server during an attack, such as a Denial of Service (DoS). Each available socket connection represents a potential route to withstand the attack. The client utilizes the discovered connection and explores other connections to the socket at the server end.

Algorithm 1: Regeneration Algorithm Pseudocode

```
Algorithm: RegenerationDefense
1. Start
2. ConnectToServer() // HTTP Client issues CONNECT
   Request for TCP connection at Port 80 to HTTP Server
3. NotifyServerConnection() // Server accepts request
   for TCP connection at port 80 and notifies client
4. SendRequestToServer() // Client sends GET/POST
   Request to server
5. ScanAndCheckURLSyntax() // IDS Scans Socket Addresses and checks URL syntax
6. IF MultipleRequestsFromSameIP() OR
   FakeURLorSQLiDetected() THEN
     6.1. IntrusionDetectionAlert()
     6.2. BlockIntruderSocketAddress()
     6.3. RegenerateNewSocketAndNotifyClient()
     6.4. GO TO 2
7. ELSE
     7.1. ReceiveHTTPRequestAndProcessObjects()
     // Web Server receives HTTP Request (URL)
     Message and processes requested objects
     7.2 SendHTMLAndObjectsToClient() // Client
     receives HTML file and objects and displays the file
     7.3 IF MoreRequestsFromClient() THEN
           7.3.1 GO TO 4
     7.4 ELSE
           7.4.1 CloseConnection()
8. Stop
```

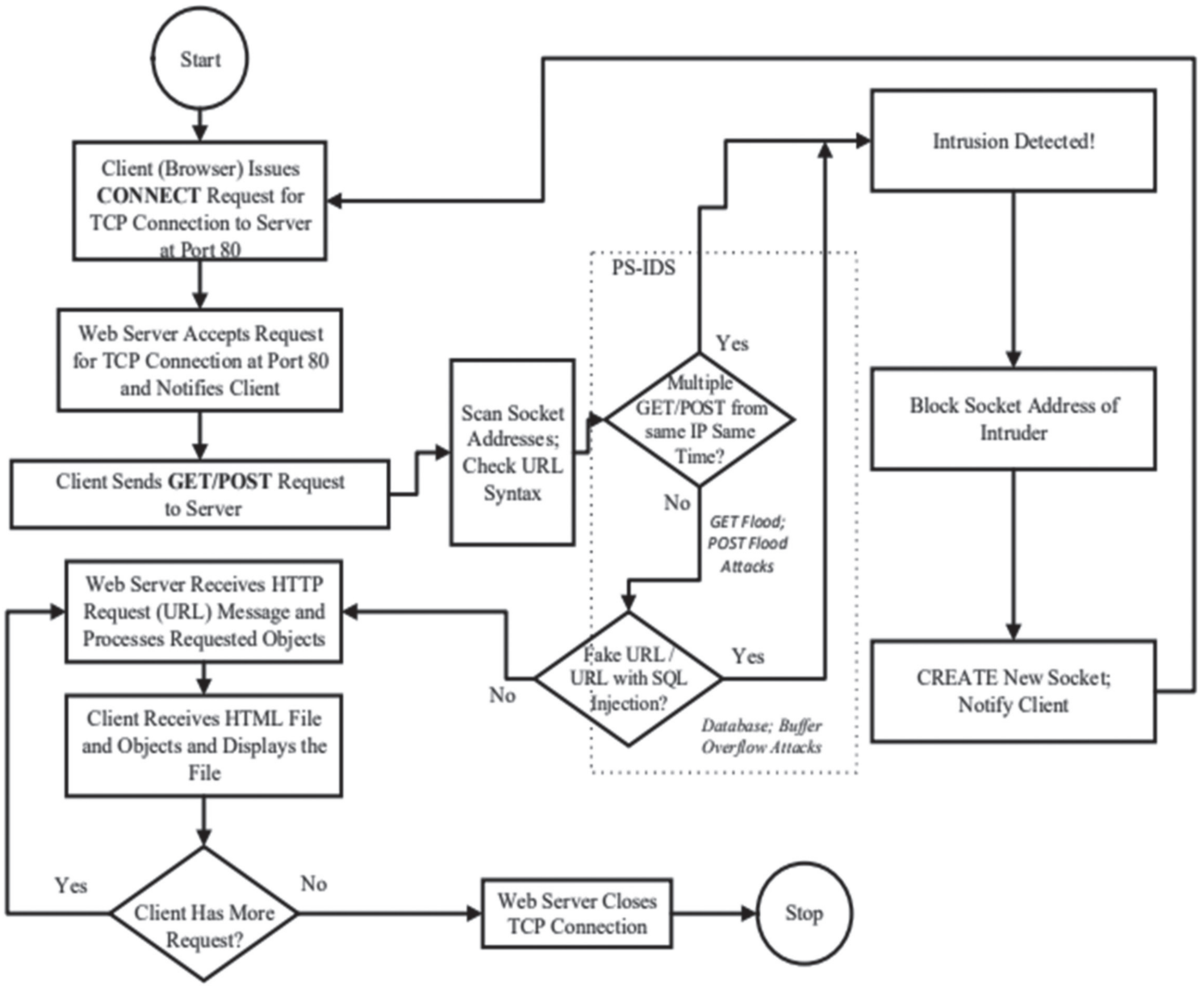
Start
Client (Browser) Issues CONNECT Request for TCP Connection to Server at Port 80
Web Server Accepts Request for TCP Connection at Port 80 and Notifies Client
Client Sends GET/POST Request to Server
Scan Socket Addresses; Check URL Syntax
PS-IDS
Multiple GET/POST from same IP Same Time?
Yes
No
GET Flood; POST Flood Attacks
Intrusion Detected!
Block Socket Address of Intruder
CREATE New Socket; Notify Client
Fake URL / URL with SQL Injection?
Yes
No
Database; Buffer Overflow Attacks
Web Server Receives HTTP Request (URL) Message and Processes Requested Objects
Client Receives HTML File and Objects and Displays the File
Client Has More Request?
Yes
No
Web Server Closes TCP Connection
Stop

Fig. 1. Regeneration Defense Flowchart

OMNeT++ served as the primary tool for evaluating the performance of the proposed model. Widely recognized as one of the most effective network simulators, OMNeT++ enjoys broad support within the scientific community. Written in C++, it is a prevalent choice for simulating networks. The INET Framework module suite was employed in OMNeT++ because of its comprehensive assortment of models for mobile networks, TCP/IP protocol stack, mobility, and its flexibility in allowing users to customize output vector statistics according to their needs. The experiment's parameters are detailed in Table 1, while Fig. 2 illustrates the simulation scenario for the model.

In the context of the OMNeT++ discrete event simulator, the purpose of HTTP is to manage traffic between the browser (referred to as the client) and the server. Three key components are employed: the browser (simulated by HTTPBrowser), the server (simulated by HTTPServer), and the controller (managed by HTTP-Controller).

To initialize the HttpTools components in a simulation, a single controller object needs to be instantiated at the scenario level. It is crucial to note that any number of nodes with browser or server components can be generated, subject to memory constraints, processing power, and practical considerations. Nodes can be linked through OMNeT++ communication links and the INET TCP/IP networking stack implementation. Alternatively, direct message passing can be employed to eliminate network effects. Simulated HTTP messages are employed by the browser and server components.

For the browser and server components, two types of hosts are available. *StandardHost* from the INET framework, suitable for a complete network simulation using the TCP/IP stack. *DirectHost*, a component of *HttpTools*, is designed for hosts employing direct message passing.

The server and browser modules leverage the TCP/IP modeling of the INET framework for transport and integrate with its StandardHost module. Additionally, these modules support OMNeT++ direct message passing when the network infrastructure effects are not considered. To provide a lightweight alternative to StandardHost, a simple container named DirectHost has been created for this purpose.

In the event of an attack, the regeneration defense is employed to reset the TCP connection, and the connection is altered.

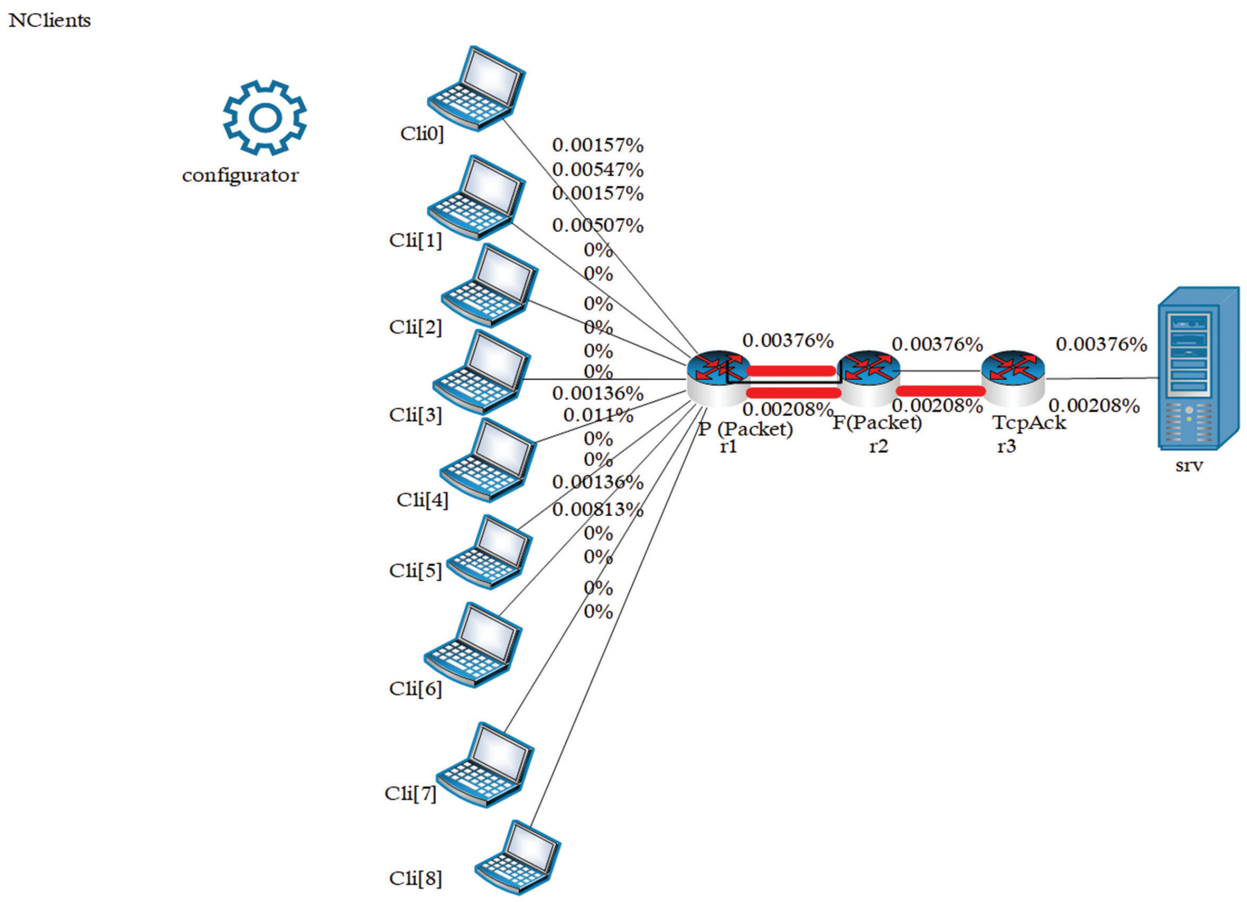

Fig. 2. OMNeT++ Simulation Setup

3.5. OMNET++ PERFORMANCE METRICS FOR EVALUATION

Performance metrics serve as a comprehensive characterization of an entire network or a specific node under examination. These metrics played a crucial role in evaluating the simulated regeneration model. The study employed four key performance metrics: network load, packet delivery ratio, network throughput, and end-to-end delay.

The concept of packet load is crucial in assessing the traffic volume generated during both the establishment and maintenance phases of the HTTP protocol, incorporating overhead for regeneration implementation. Notably, the proposed model integrates mobility rate as a significant factor, as increased mobility may result in intermittent connection disruptions during regeneration defense operations. This consideration underscores the importance of accounting for mobility's impact on connection stability and overall system performance.

Network throughput, a key metric reflecting the efficiency of data transmission, is essential for evaluating protocol performance. Despite being a desirable goal for any protocol, this study acknowledges the trade-offs involved, particularly when regeneration defense operations cause temporary connection resets. Additionally, heightened mobility exacerbates the situation by inducing frequent topology changes, affecting packet transmission across different sockets and potentially impeding throughput optimization. These insights highlight the necessity of balancing network throughput objectives with the disruptive effects of regeneration defense and mobility on connection stability and data transmission efficiency.

The packet delivery ratio, indicative of data loss and protocol efficiency, plays a critical role in assessing a protocol's performance. However, it's vital to recognize that a high mobility rate can diminish protocol throughput, underscoring the intricate relationship between mobility, data delivery, and protocol efficiency. Furthermore, end-to-end delay, encompassing various factors influencing packet transmission time, such as buffer queues and transmission delays, is observed to increase with higher mobility rates. These findings emphasize the need for comprehensive evaluations that consider mobility's impact on data delivery, protocol efficiency, and end-to-end transmission delays to ensure robust system performance.

4. EXPERIMENTAL RESULTS AND DISCUSSION

This section provides a detailed discussion of the simulation results based on the input parameters outlined in Table 1. The proposed regeneration defense model's performance was assessed using the four metrics when confronted with denial-of-service attacks.

Table 1. Parameter Settings for Layer 4 HTTP Protocol Simulation

Parameter	Value
Simulation	3600
Number of Nodes	100
Simulation Area (m)	500 X 500
Sending Interval) s)	1
Protocol	HTTP
Node Velocity (m/s)	10
Data Rate (MB/s)	10
Transmit Power	2.0mW
No. of Channels	10
Carrier Frequency	2.4/5Ghz
Traffic Model	TCP
MAC Protocol	IEEE 802.11g
Packet Size – CBR Bytes	100

As illustrated in Fig. 3a, during the attack, packet data delivery plummeted to 31% of the normal value, preventing nodes from transmitting data and causing a denial of service to the server for intended packet transmissions. Upon deploying the regeneration defense, channel access was swiftly restored to 69% of its initial value within the 10ms experimental timeframe, effectively neutralizing the attack. Once neutralized, resources were promptly redirected to data transmission, achieving a 69% data transmission rate. The remaining 31% of resources were utilized to assess the extent of damage caused by the attack and transmit attack information to the network administrator for policy enforcement.

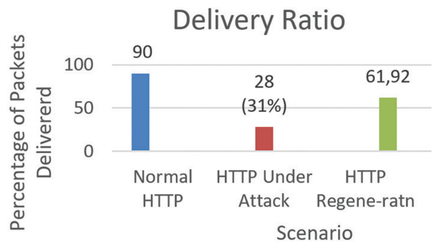

Fig. 3a. Delivery Ratio Results

Fig. 3b depicts the system throughput, revealing a reduction to 62 packets/s under attack conditions, signifying a 19-packet/s decrease during the investigation and neutralization period. As the system resumed normal data transmission, throughput increased to 73% of its value within the 10ms experimental time.

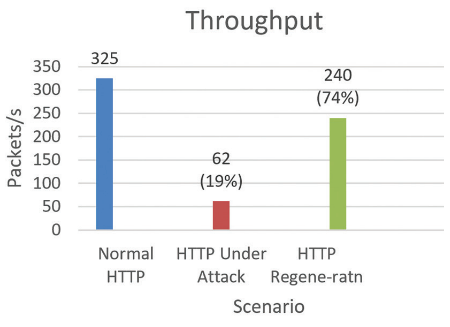

Fig. 3b. Throughput Results

The delay experienced during the attack is presented in Fig. 3c, indicating a delay of 3s. With the deployment of regeneration, the delay was reduced to 0.6s, despite the need for connection resets and resource reallocations. Notably, this delay is nearly equivalent to the delay observed in the absence of an attack (0.5s).

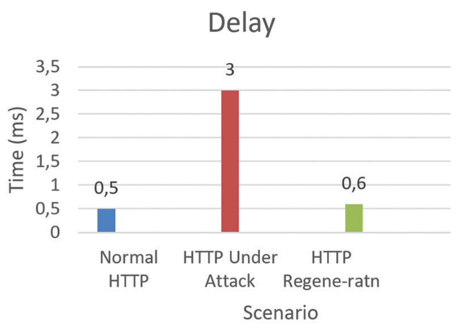

Fig. 3c. Delay Results

In Fig. 3d, the load is considerably lower (2%) during an attack compared to normal or defense operations. This reduction is attributed to the denial-of-service attack, which suspended all network control and maintenance packets during DDoS, rendering normal network load packets non-functional. The model efficiently restored the native load to 100% of its value, along with an additional overhead (136%) from the regeneration implementation, almost immediately.

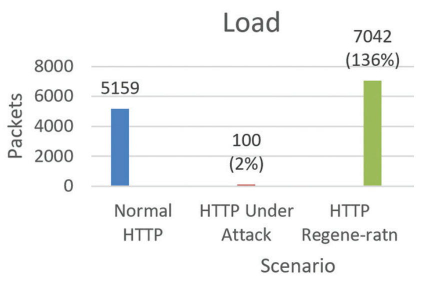

Fig. 3d. Network Load Results

4.1. COMPARISON RESULTS

This section presents a comprehensive comparison of the proposed HReg model with the state-of-the-art approach [26]. The evaluation encompasses the design, experimental methodology, metrics employed, results obtained, and outlook. Table 2 provides a succinct summary of this comparative analysis. For a detailed performance assessment, our proposed model evaluates throughput, delivery ratio, delay, and load individually. This approach allows us to discern the simulation's efficiency in terms of network performance, message delivery, delay characteristics, and load handling. Notably, this work focuses on survivability as its primary goal.

In contrast, the existing approach employs binary classification to measure results concerning recall, precision, specificity, and sensitivity. Their methodology is tailored towards detection only, aligning with the primary objective of the previous work. Furthermore, while the datasets for the existing model are generated from their university network, the proposed model utilizes embedded datasets from OMNET++. This choice enhances the versatility and applicability of the HReg model in diverse scenarios.

Table 2. Comparative Analysis of Regeneration Model and State-of-the-art

Category	Existing Model	Regeneration Model
Experimental Platform	C and Java Programs	OMNET++ Simulation Environment
Method	Binary Classification	Discrete Events Simulation
Dataset	Tezpur University Web Server Dataset	OMNET++ INET Dataset
Metrics	Recall, Precision, Specificity, and Sensitivity	Throughput, Delivery Ratio, Delay, and Load
Attack Target(s)	Slowloris DDoS Attack	DoS, DDoS, SQLi Attacks
Model Objective(s)	Detection of Attacks Only	Detection and Survivability from Attacks
Deployment Platform(s)	No Specific Domain	Low-resource, Full-resource Mobile Wireless Domains

One significant distinction lies in the scope of attack detection. The existing model can only detect Slowloris DDoS attacks, whereas our model is proficient in identifying all variants of DoS and SQLi attacks, ensuring a broader range of security coverage and improved survivability. Moreover, our model is designed to operate seamlessly on both low-resource and full-resource devices within wireless networks. In contrast, the existing model lacks a specified target domain or platform, making our approach more adaptable to various environments and resource constraints.

This study developed a distinctive approach to safeguarding HTTP servers by merging survivability with intrusion detection. This method stands in contrast to the current state-of-the-art practices, which typically focus on either intrusion detection or intrusion prevention.

5. CONCLUSION

A thorough examination of existing Intrusion Detection Systems (IDS) reveals a notable emphasis on the detection of attacks, often overlooking the critical aspect of survivability during such incidents. The proposed HReg regeneration model for HTTP intrusion detection and defense mechanism breaks away from this trend by addressing both the connection channel and the web application's protection. This approach aims to detect attacks and establish survivable connections within the underlying TCP layer, replacing compromised connections during imminent or full-scale attacks. Specifically designed for Mobile Ad-Hoc Networks (MANETs) and other HTTP-based mobile wireless devices, the model is adaptable to diverse distributed networks in real-world scenarios. It incorporates a dynamic threat database for MANET networks to respond to evolving and known threats effectively. Continuous monitoring of its performance over time in various network environments will provide insights into the dynamic nature of network traffic and potential emerging threats.

Future research endeavors should aim to integrate simulation methodologies with machine learning techniques. This involves training the machine learning model on simulated datasets to optimize its performance in real-world applications.

Acknowledgments:

The authors express gratitude to TETFUND for the support received through the 2020 research grant intervention. Special thanks to ACE OAK-PARK, Obafemi Awolowo University, for providing additional funding, laboratory facilities, and necessary space for this research.

6. REFERENCES

[1] K. Kumar, “Denial of service attacks – an updated perspective”, Systems Science & Control Engineering, Vol. 4, No. 1, 2016, pp. 285-294.

[2] Kaspersky, “Rising Threats: Cybercriminals Unleash 411,000 Malicious Files Daily in 2023”, https://www.kaspersky.com/about/press-releases/2023_rising-threats-cybercriminals-unleash-411000-malicious-files-daily-in-2023 (accessed: 2024)

[3] Z. Cekerevac, Z. Dvorak, L. Prigoda, P. Cekerevac, “Hacking, Protection and the Consequences of Hacking”, Komunikacie, Vol. 20, No. 2, 2018, pp. 83-87.

[4] B. A. Obotivere, A. O. Nwaezeigwe, “Cyber Security Threats on the Internet and Possible Solutions“, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 9, No. 9, 2020, pp. 92-97.

[5] B. A. Forouzan, “Data Communications and Networking”, The McGraw-Hill Companies Inc., 2007.

[6] V. J. Glowniak, “An Introduction to the Internet, Part 3, Internet Services”, Journal of Nuclear Medicine Technology, Vol. 23, No. 4, 1995, pp. 231-248.

[7] R. F. Silva, R. Barbosa, J. Bernardino, “Intrusion Detection Systems for Mitigating SQL Injection Attacks: Review and State-of-Practice”, International Journal of Information Security and Privacy, Vol. 14, No. 2, 2020, pp. 20-40.

[8] Patil, A. Laturkar, S. V. Athawale, R. Takale, P. Tathawade "A multilevel system to mitigate DDOS, brute force and SQL injection attack for cloud security", Proceedings of the International Conference on Information, Communication, Instrumentation and Control, Indore, India, 17-19 August 2017, pp. 1-7.

[9] A. Dizdar, "SQL Injection Attack: Real Life Attacks and Code Example", https://brightsec.com/blog/sql-injection-attack/ (accessed: 2023)

[10] I. Lee, S. Jeong, S. Yeo, J. Moon, "A novel method for SQL injection attack detection based on removing SQL query attribute values", Mathematical and Computer Modelling, Vol. 55, 2012. pp. 58-68.

[11] Y. Huang, F. Yu, C. Hang, C. H. Tsai, D. T. Lee, S. Y. Kuo, "Securing web application code by static analysis and runtime protection", Proceedings of the 13th International World Wide Web Conference, May 2004, pp. 40-52.

[12] G. Baldini, I. Amerini, "Online Distributed Denial of Service (DDoS) intrusion detection based on the adaptive sliding window and morphological fractal dimension", Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 210, No. C, 2022.

[13] G. Baldini, "On the application of entropy measures with sliding window for intrusion detection in automotive in-vehicle networks", Entropy, Vol. 22 No. 9, 2022, p. 1044.

[14] S. D. Çakmakç, T. Kemmerich, T. Ahmed, N. Baykal, "Online DDoS attack detection using Mahalanobis distance and kernel-based learning algorithm", Journal of Network and Computing Applications, Vol. 168, 2020, p. 102756.

[15] S. L. Bernal, D. P. Martins, A. H. Celdrán, "Towards the mitigation of distributed denial-of-service cyberbioattacks in bacteria-based biosensing systems", Digital Signal Processing, Vol. 118, 2021, p. 103241.

[16] P. Sakthibalan, K. Devarajan, "DFMS: Differential flow management scheme for denial-of-service impact mitigation in 5G communications", Journal of King Saud University – Computer and Information Sciences, Vol. 34, 2022, pp. 5366–5374.

[17] A. B. M Ali, A. Y. I Shakhatreh, M. S. Abdullah, J. Alostad, "SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks", Procedia Computer Science, Vol. 3, 2011, pp. 453-458.

[18] A. Patil, R. Gaikwad, "Comparative analysis of the Prevention Techniques of Denial-of-Service Attacks in Wireless Sensor Networks", Procedia Computer Science, Vol. 48, 2015, pp. 387-393.

[19] F. Bensalah, N. E. Kamoun, M. E. Houssaini, "Online detection of Denial-of-Service Attacks in Software Defined Networking using the Hotelling Chart", Procedia Computer Science, Vol. 160, 2019, pp. 785-790.

[20] Y. Tian, V. Tran, M. Kuerban, "DOS Attack Mitigation Strategies on SDN Controller", Proceedings of the IEEE 9th Annual Computing and Communication Workshop and Conference, Las Vegas, NV, USA, 7-9 January 2019, pp. 701-707.

[21] L. Erdődi, Å. Å. Sommervoll, F. M. Zennaro, "Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents", Journal of Information Security and Applications, Vol. 61, 2021, p. 102903.

[22] F. J. Abdullayeva, "Distributed denial of service attack detection in E-government cloud via data clustering", Array, Vol. 15, 2022, p. 100229.

[23] E. Osa, P. E. Orukpe, U. Iruansi, "Design and implementation of a deep neural network approach for intrusion detection systems", e-Prime - Advances in Electrical Engineering, Electronics and Energy, Vol. 7, 2024, p. 100434.

[24] F. Nabi, X. Zhou, "Enhancing intrusion detection systems through dimensionality reduction: A comparative study of machine learning techniques for cyber security", Cyber Security and Applications, Vol. 2, 2024, p. 100033.

[25] A. Fadlila, I. Riadib, M. A. Mu'min. "Mitigation from SQL Injection Attacks on Web Server using Open Web Application Security Project Framework", Internation Journal of Engineering, Vol. 37, No. 4, 2024, pp. 635-645.

[26] T. Kangkan, B. Debojit, "Slowloris Attack Detection Using Adaptive Timeout-Based Approach", ISeCure, Vol. 16, No. 1, 2024, p. 79.