An Attack Simulation Tool for DDoS and XSS Attack using Graph Database

Tamasit Savanpopan, Peemawat Phosrinak, and Vasin Vorarit
School of Information, Computer and Communication Technology (ICT)
Sirindhorn International Institute of Technology, Thammasat University
Pathum Thani, Thailand

Dr. Somchart Fugkaew
School of Information, Computer and Communication Technology (ICT)
Sirindhorn International Institute of Technology, Thammasat University
Pathum Thani, Thailand

***Abstract*—It is essential to have strong tools to simulate and analyze various attack scenarios in the rapidly developing field of cybersecurity. The purpose of this project is to improve cybersecurity by creating a Python-based system that can detect and identify Cross-Site Scripting (XSS) and Distributed Denial of Service (DDoS) attacks. By simulating log histories, the python coding identifies IPs associated with these threats, contributing to the creation of improved defense mechanisms against future attacks.The project includes a Python-based framework for log data simulation and detection that creates fake log data that simulates both typical and anomalous activity. Neo4j, a graph database, generates the visual part. The simulated environment provides a clear comprehension of the network of interactions by visualizing detected activities and relationships. For cybersecurity professionals, this graphical representation improves the communication of information.**

***Index Terms*—Cybersecurity, DDoS Attacks, XSS Attacks, Python Coding, Log Data Simulation, Neo4j Visualization.**

I. Introduction

Nowaday, protecting computer systems from DDoS (Distributed Denial of Service) and XSS (Cross-Site Scripting) attacks requires effective cybersecurity prevention. The Distributed DoS (DDOS) attacks are extremely innovative and complex, making them almost inevitable to detect by the existing technology or detection system. Due to their complexity and difficulty, novel types of DDoS attacks are practically impossible for intrusion detection systems to detect or mitigate. [1] Sensitive information is compromised by XSS attacks, which control websites to distribute malicious scripts to users. Around 40 percent attacks on Websites globally occurred due to XSS (Cross Site Scripting). Cross Site Scripting targets both vulnerable and non-vulnerable website. To protect and prevent the website from XSS we must know the complexity and different ways of prevention. [2] Organizations strengthen their defenses against XSS threats by implementing web application firewalls, updating web applications on a regular basis, and using secure coding practices. Proactive cybersecurity measures strengthen the overall resilience of digital ecological systems by protecting not only the integrity of the internet but also user trust.

Understanding the complex principles and patterns of cyber threats, especially Distributed Denial of Service (DDoS) and Cross-Site Scripting (XSS) attacks, is crucial in the volatile field of cybersecurity.

This research project uses a combination of methods that includes Python code detection, log data simulation, and Neo4j visualization to further this understanding. The main objective of this study is to improve our understanding of the complexities involved in DDoS and XSS attacks.

Simulating log histories, the project identifies potentially malicious IPs, providing insight into the techniques used in these two common attack types. The knowledge gathered from this analysis forms the basis for creating defense systems that are more robust, essential for minimizing the effect of similar threats in the future.

The project involves using Python coding to develop a potent DDoS and XSS Attack Detection System. By effectively identifying suspicious activity in the log data, this system provides a preventive method to spot and address possible threats. Because coding is the main focus, the detection system can be flexible and developed to meet and help prevent new cybersecurity threats. The structure for Log Data Simulation and Data Analysis works together with the detection system. This component involves producing fake log data that includes both abnormal and normal activities, as well as creating a log format. By giving the detection system a realistic testing ground, this simulation helps to improve its effectiveness.

In addition, Neo4j, a powerful graph database, is integrated into the project for visualization purposes. A clear and simple understanding of the complex structure of interactions is provided by the visual representation of the relationships and activities that have been identified within the simulated environment. This visual component strengthens the project's communication impact and makes it easier for cybersecurity professionals to convert technical insights into workable strategies. The idea essentially explores DDoS and XSS attacks in depth, using log data simulation for practical testing, Neo4j for insightful visualization, and Python coding for dynamic

detection. By implementing a comprehensive approach, the research anticipates to advance defenses and encourage a proactive approach to evolving cyber threats.

II. Proposed System

A. Python-based DDoS and XSS Attack Detection System

This essential component includes utilizing Python programming to create an operating detection system. Incoming log data can be continuously monitored by the system, which will use complicated algorithms to spot patterns that point to DDoS and XSS attack activity. The simplicity of use and adaptability of Python coding with current cybersecurity infrastructure ensures deployment in a variety of environments with minimal difficulties.

B. Log Data Simulation and Data Analysis

The implementation of a Log Data Simulation and Data Analysis module will verify the effectiveness of the detection system. This requires generating fake log data with both normal and abnormal activities, as well as developing a detailed log format. The simulation offers a secure environment for evaluating and improving the accuracy of the detection system, guaranteeing reliable operation in a variety of attack scenarios. Through the use of data analysis tools, the simulated log data will provide insightful information that will enable the detection algorithms to be continuously improved.

C. Utilization of Neo4j for Visualization

Neo4j is a graph database that used in the project to visualize relationships and activities that are found in the simulated environment. The graph-based representation provided by Neo4j helps cybersecurity experts understand attack patterns and how they impact the system by providing a user-friendly visualization of the complex network of interactions. Visualization is an effective tool for communication that allows effective collaboration and knowledgeable decision-making by providing a clear picture of the risks that have been identified.

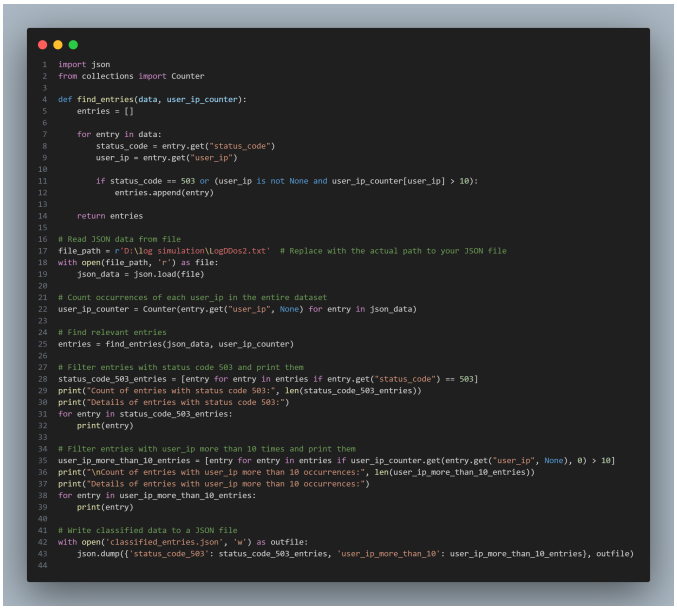
III. Implementation

- Install Visual Studio Code (VScode) or other notebook Python programming and set all the important helper tools plugin that are able to create an efficient code for detecting to simulation log.
- Install the neo4j community edition with version 4.4.28 or higher and get OpenJDK 8 or Oracle Java 8.
- Setting up Python programming with import the necessary library and setting up the connectivity for the localhost with neo4j.

A. Distributed Denial-of-Service Attack

For the Denial-of-Service Attack (DDoS attack), In this project we use 2 conditions to process the log classification to find the log event that is suspicious to be a DDoS attack. The first condition, status code 503, status code 503 means that the service of the server is unavailable due to the server crash, after researching information for the DDoS attack pattern, we have found that status code 503 is one of the patterns that can happen in DDoS attack pattern. So, we utilize the python code can create a logic function for checking the log event that has the status code equal to 503, and can be assumed that may maybe DDoS will happen. The second condition, the number of requests per IP address is more than 10 times in a short period of time, the threshold value depends on the web server structure. So, we utilize the Python code can create a logic function for checking the request IP that is within the condition scope and we assume that this IP address will or will be the cause of a DDoS attack.

From the code, first we code to read the log file in a text file but in the text file we generate a log in JSON format. We focused on two variables which are status code and user IP. Then, create a function that detect two conditions that we stated before in above section. The python code is shown in Fig. 1



```python
import json
from collections import Counter

def find_entries(data, user_ip_counter):
    entries = []

    for entry in data:
        status_code = entry.get("status_code")
        user_ip = entry.get("user_ip")

        if status_code == 503 or (user_ip is not None and user_ip_counter[user_ip] > 10):
            entries.append(entry)

    return entries

# Read JSON data from file
file_path = r'D:\log simulation\LogDDos2.txt'  # Replace with the actual path to your JSON file
with open(file_path, 'r') as file:
    json_data = json.load(file)

# Count occurrences of each user_ip in the entire dataset
user_ip_counter = Counter(entry.get("user_ip", None) for entry in json_data)

# Find relevant entries
entries = find_entries(json_data, user_ip_counter)

# Filter entries with status code 503 and print them
status_code_503_entries = [entry for entry in entries if entry.get("status_code") == 503]
print("Count of entries with status code 503:", len(status_code_503_entries))
print("Details of entries with status code 503:")
for entry in status_code_503_entries:
    print(entry)

# Filter entries with user_ip more than 10 times and print them
user_ip_more_than_10_entries = [entry for entry in entries if user_ip_counter.get(entry.get("user_ip", None), 0) > 10]
print("\nCount of entries with user_ip more than 10 occurrences:", len(user_ip_more_than_10_entries))
print("Details of entries with user_ip more than 10 occurrences:")
for entry in user_ip_more_than_10_entries:
    print(entry)

# Write classified data to a JSON file
with open('classified_entries.json', 'w') as outfile:
    json.dump({'status_code_503': status_code_503_entries, 'user_ip_more_than_10': user_ip_more_than_10_entries}, outfile)
```

Fig. 1. DDoS Detection Code

From Fig. 2 is the picture that shows the attack pattern of DDoS using status code 503. The purple node represents the user IP address connected to the orange node which is the server node. All IP nodes that are shown on the graph have the status code 503. To sum up, this picture showing means that a DDoS attack is happening or in the process of attacking.

From Fig. 3 is the picture that shows the attack pattern of DDoS using IP address request more than 10 times in a short period of time. The purple node represent the user IP that trying to connect to the server many times in a short period of time. The output shows the blue node, which may have a status code of 200, possibly indicating the onset of a DDoS attack. However, if there is a high volume of requests within a specific time, it could suggest a suspicious IP address. In summary, it is assumed that it might happen, but it's not confirmed to occur 100 percent.

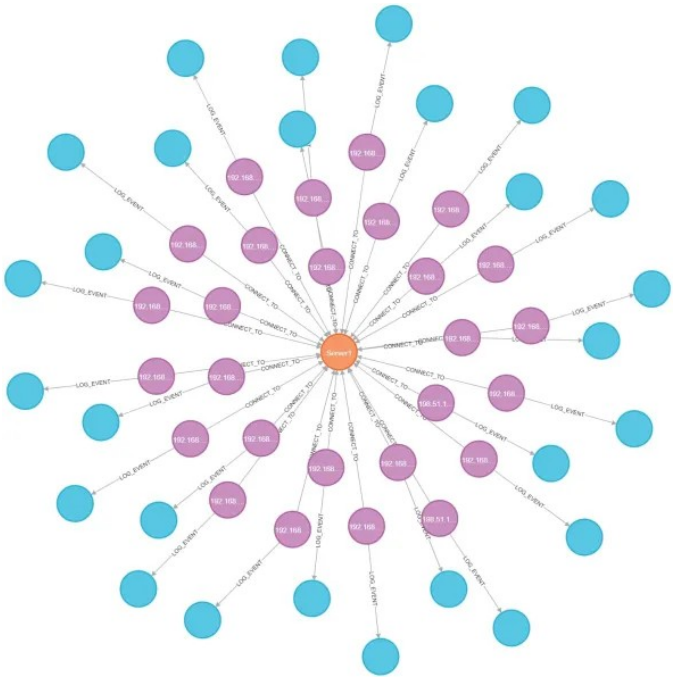


Fig. 2. Condition 1: Status Code 503

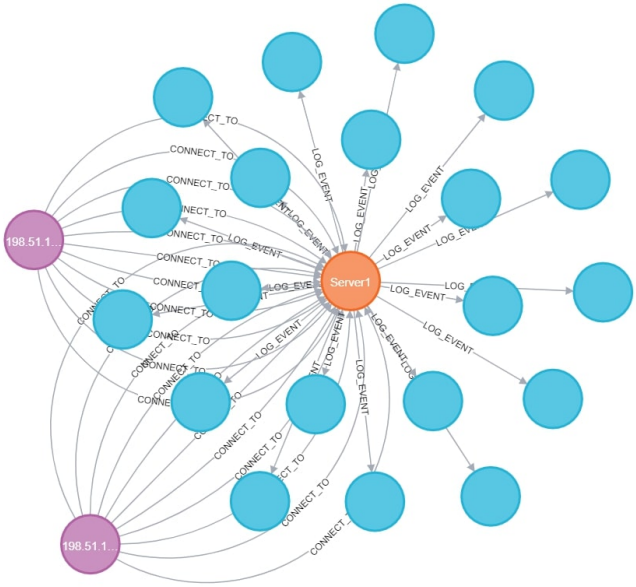


Fig. 3. Condition 2: Request more than 10 times

B. Cross-site Scripting Attack

For the Cross-site Scripting (XSS attack), In this project we use 1 condition to detect the pattern of log which can be IP address that send the request to web server and in the request it could have script, alert, JavaScript, tag etc. and in these script, the Trojan or another kind of attack can be involves. And all the tag that be in the log is possibly suspicious to be a harmful log. We code the python to detect these kinds of script. And the python code for detection is shown as Fig. 5

From Fig. 4 is a Neo4j graph database visualization that shows an attack pattern of XSS cross-site script by the red node is a node that could be an IP address that has a tag JavaScript or malicious script inside and sent to the web server (orange node). Some of the red nodes are not the script that involves the IP address but involve from another source or directly to the web server for attacking.

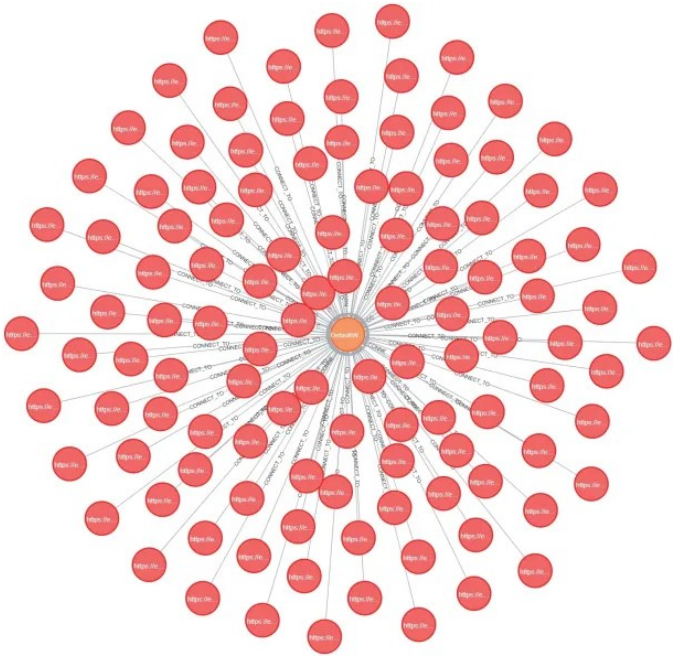


Fig. 4. Python Code for Detecting XSS

C. Log File

From Fig. 6 is show the example of log in DDoS. From Fig. 7 is show the output of DDoS condition 1, the status code is 503. From Fig. 8 is show the output of DDoS condition 2, the request is too much in a period of time. From Fig. 9 is show the example of log in XSS. From Fig. 10 is show the output of XSS condition, detecting the tag script or malicious script.

IV. Conclusion

In this project, we implement the fake log simulation with normal and abnormal activities or IP addresses and write a Python code to detect the abnormal activity from the log file lastly, we use the neo4j graph database to visualize the relations of the flow of each condition that we used to classify

```python
import json

def detect_xss_entries(json_data):
    xss_entries = set()
    count = 0

    def is_potential_xss(value):
        # Check for potential XSS patterns
        xss_patterns = ['<script>', 'javascript:', 'onerror=', 'alert(']
        for pattern in xss_patterns:
            if pattern in value:
                return True
        return False

    for entry in json_data:
        has_xss = False
        for key, value in entry.items():
            if isinstance(value, str) and is_potential_xss(value):
                has_xss = True
                break

        # Check URL parameters for potential XSS
        if "url" in entry and isinstance(entry["url"], str) and is_potential_xss(entry["url"]):
            has_xss = True

        if has_xss:
            xss_entries.add(json.dumps(entry))
            count += 1

    return count, [json.loads(entry) for entry in xss_entries]

file_path = r"c:\Users\User\Desktop\LogXSS.txt"
try:
    with open(file_path, "r") as file:
        log_data = json.load(file)
except json.JSONDecodeError as e:
    print(f"Error decoding JSON: {e}")

# Detect entries related to XSS attacks
count, xss_entries = detect_xss_entries(log_data)

# Output the result
print(f"Number of potential XSS entries: {count}")
print(f"Entries related to potential XSS attacks:")
for entry in xss_entries:
    print(entry)
```

Fig. 5. Python Code for Detecting XSS

Fig. 6. Example of Log for DDoS attack

Fig. 7. Output from DDoS Condition 1

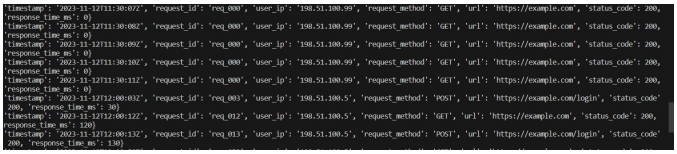


Fig. 8. Output from DDoS Condition 2

Fig. 9. Example of Log for XSS attack

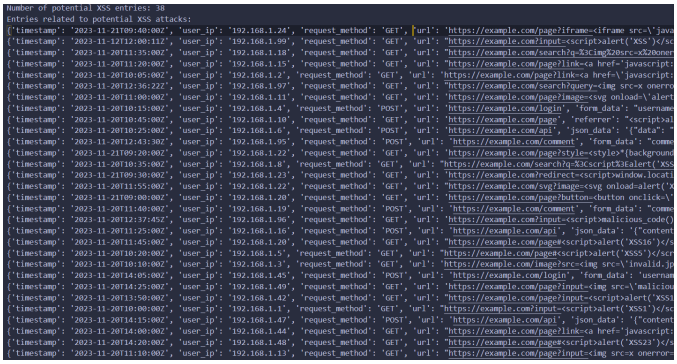


Fig. 10. Example of Log for XSS attack

the log whether it is normal or abnormal with a different kind of attack has a different condition set up. In DDoS, we have 2 conditions which are threshold analysis and monitoring the status code. In XSS, we have 1 condition, which is looking at various kinds of patterns of the script.

V. Future Work

Improving the logic of both attacks to be more efficient and focus more on timestamp or adapting the condition suitable to that environment. Improve the log file to make it more realistic and have a lot of periods so that we can compare the attack time. For the graph, we can visualize traffic data that relate to the time of the web server so, we can detect what period of time has a high traffic then we can assume that high traffic may cause the attack. Improve relation node in Neo4j such as making a query to update node to the browser and make it more efficient, easy-looking, and easy-understading.

Acknowledgment

The project could not be complete without the hard work and contribution of our members. We would like to gratitude to Dr. Somchart Fugkaew, our advisor, for his knowledge sharing, kindness, and support of our project. We would like to thank to School of Information, Computer and Communication Technology (ICT), Sirindhorn International Institute of Technology, Thammasat University for providing the necessary resources for our project.

References

[1] R. S. Devi, R. Bharathi, and P. K. Kumar, "Investigation on efficient machine learning algorithm for ddos attack detection," in *2023 International Conference on Computer, Electrical Communication Engineering (ICCECE)*, 2023, pp. 1–5.

[2] A. Mishra and S. Juneja, "Prevention of website from cross site scripting," in *2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*, 2023, pp. 471–473.