Presentation to **Dr.Somchart Fugkeaw**

CSS453 | Cyber Crimes & Digital Forensics

Attack Simulation Tool

DDoS & XSS

Group Members

01 Name : Peemawat Phosrinak
ID : 6222770164

02 Name : Vasin Vorarit
ID : 6222772038

03 Name : Tamasit Savanpopan
ID : 6222781815

Presentation to **Dr.Somchart Fugkeaw**

CSS453 | Cyber Crimes & Digital Forensics

Overview

Tools

DDoS Attack

XSS Attack

Future Work

Overview

This study of the project has enhanced the understanding of the principles and patterns of DDoS and XSS attacks. By monitoring log history, it identifies IPs that are likely involved in these two types of attacks. This leads to the development of improved defense systems to prevent such attacks in the future.

Objective

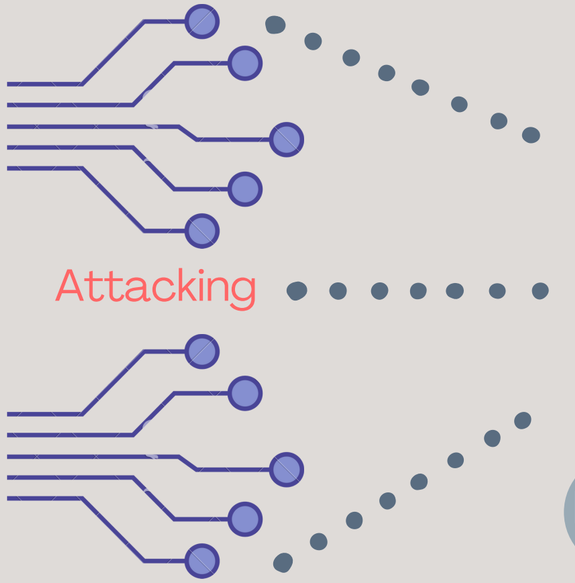
01 Develop DDoS and XSS Attack Detection System

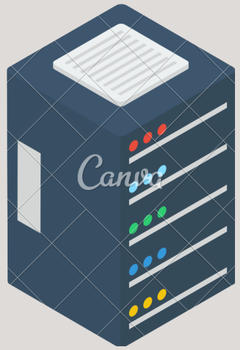
02 Log Data Simulation & Data Analysis

03 Use of Neo4j for Visualization

Attacker

Attacking



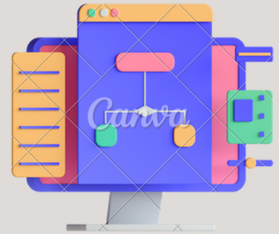


DNS Server

Return

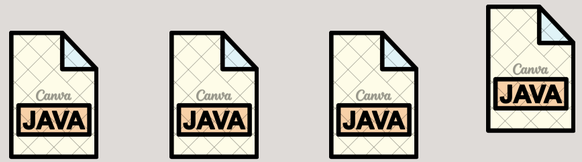
Log Events

Analyzed by



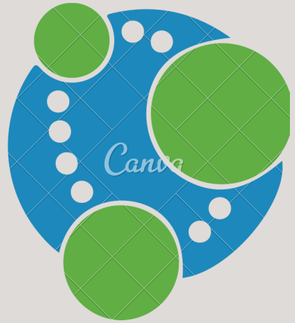
Algorithm Detection

Detect



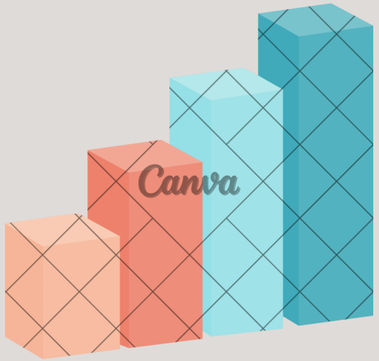
Detected Cross-site Script

Simulated

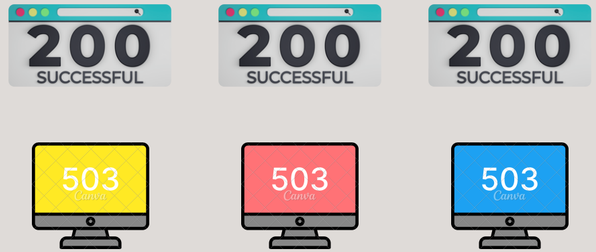


Neo4J

Result in

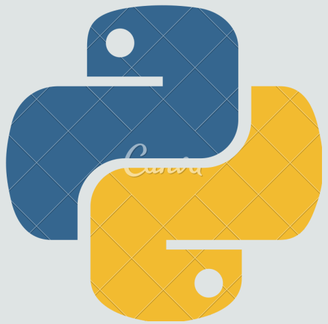


Graph Database Visualization



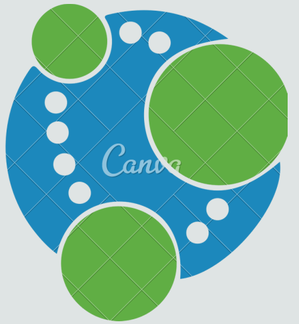
Detected DDoS

PYTHON



- The language applied to DDoS along with XSS attack log classification.

NEO4J



- A tool for demonstrating data that constructs nodes in order to display the relationships between nodes as a graph.

How can We detect DDoS Attack?

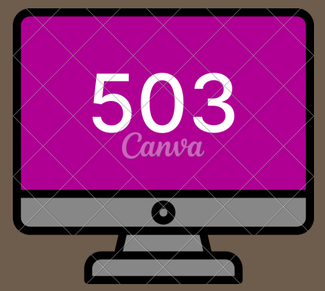
DDoS Attack Patterns

01 Threshold Analysis

- It will be suspected of a DDoS attack if the server receives 10 or more requests via the same IP in short period of time.

02 Status Code 503

- A status code 503—which appears to be a DDos attack—occurs when the server is experiencing more traffic than it can handle.



EXAMPLE OF LOGS

```
{
    "timestamp": "2023-11-12T12:00:21Z",
    "request_id": "req_021",
    "user_ip": "192.168.1.21",
    "request_method": "POST",
    "url": "https://example.com/login",
    "status_code": 200,
    "response_time_ms": 210
},
{
    "timestamp": "2023-11-12T12:00:22Z",
    "request_id": "req_022",
    "user_ip": "192.168.1.22",
    "request_method": "GET",
    "url": "https://example.com",
    "status_code": 200,
    "response_time_ms": 220
},
{
    "timestamp": "2023-11-12T12:00:23Z",
    "request_id": "req_023",
    "user_ip": "192.168.1.23",
    "request_method": "POST",
    "url": "https://example.com/login",
    "status_code": 200,
    "response_time_ms": 230
},
{
    "timestamp": "2023-11-12T12:00:24Z",
    "request_id": "req_024",
    "user_ip": "192.168.1.24",
    "request_method": "GET",
    "url": "https://example.com",
    "status_code": 200,
    "response_time_ms": 240
},
```

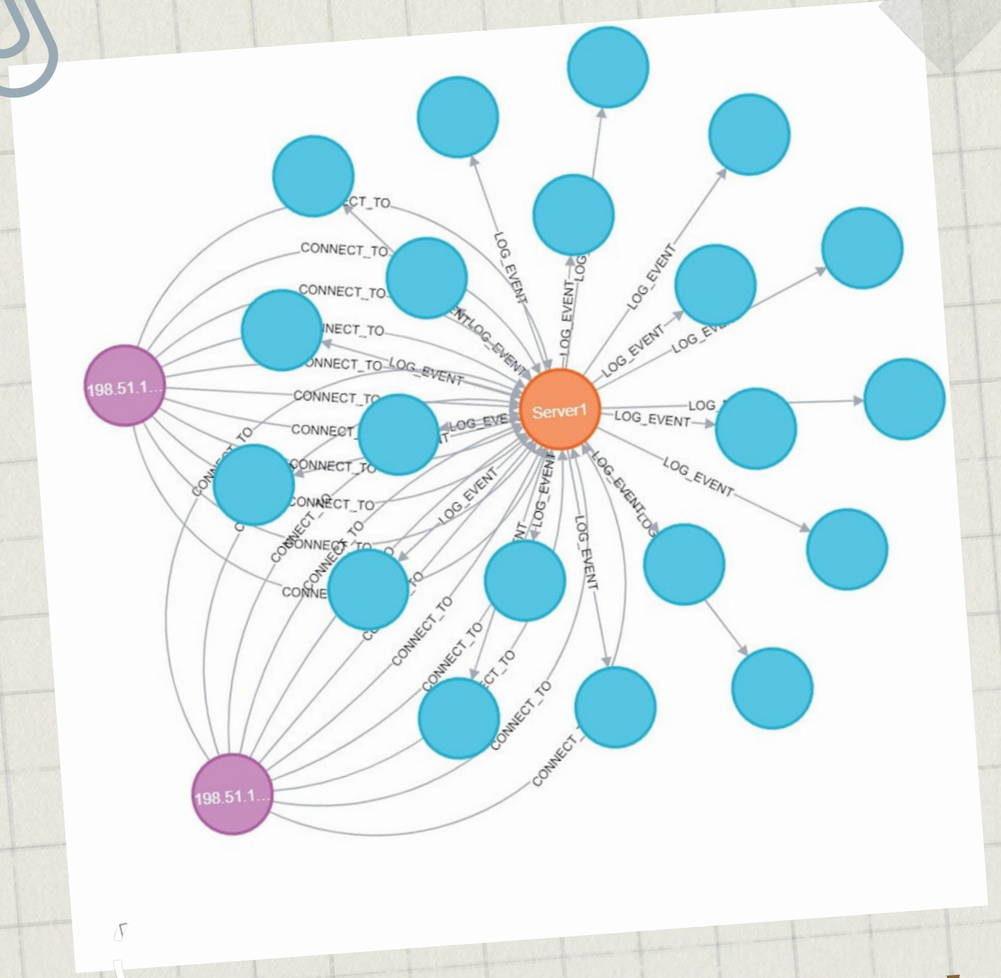
TRESHOLD DETECTED

```
{
    "timestamp": "2023-11-12T11:30:01Z",
    "request_id": "req_000",
    "user_ip": "198.51.100.99",
    "request_method": "GET",
    "url": "https://example.com",
    "status_code": 200,
    "response_time_ms": 0
},
{
    "timestamp": "2023-11-12T11:30:02Z",
    "request_id": "req_000",
    "user_ip": "198.51.100.99",
    "request_method": "GET",
    "url": "https://example.com",
    "status_code": 200,
    "response_time_ms": 0
},
{
    "timestamp": "2023-11-12T11:30:03Z",
    "request_id": "req_000",
    "user_ip": "198.51.100.99",
    "request_method": "GET",
    "url": "https://example.com",
    "status_code": 200,
    "response_time_ms": 0
},
```

STATUS CODE 503 DETECTED

```
{
    "timestamp": "2023-11-12T12:00:25Z",
    "request_id": "req_025",
    "user_ip": "192.168.1.25",
    "request_method": "POST",
    "url": "https://example.com/login",
    "status_code": 503,
    "response_time_ms": null,
    "error_message": "Service Unavailable - DDoS Attack"
},
{
    "timestamp": "2023-11-12T12:00:26Z",
    "request_id": "req_026",
    "user_ip": "192.168.1.26",
    "request_method": "GET",
    "url": "https://example.com",
    "status_code": 503,
    "response_time_ms": null,
    "error_message": "Service Unavailable - DDoS Attack"
},
```

```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)

```

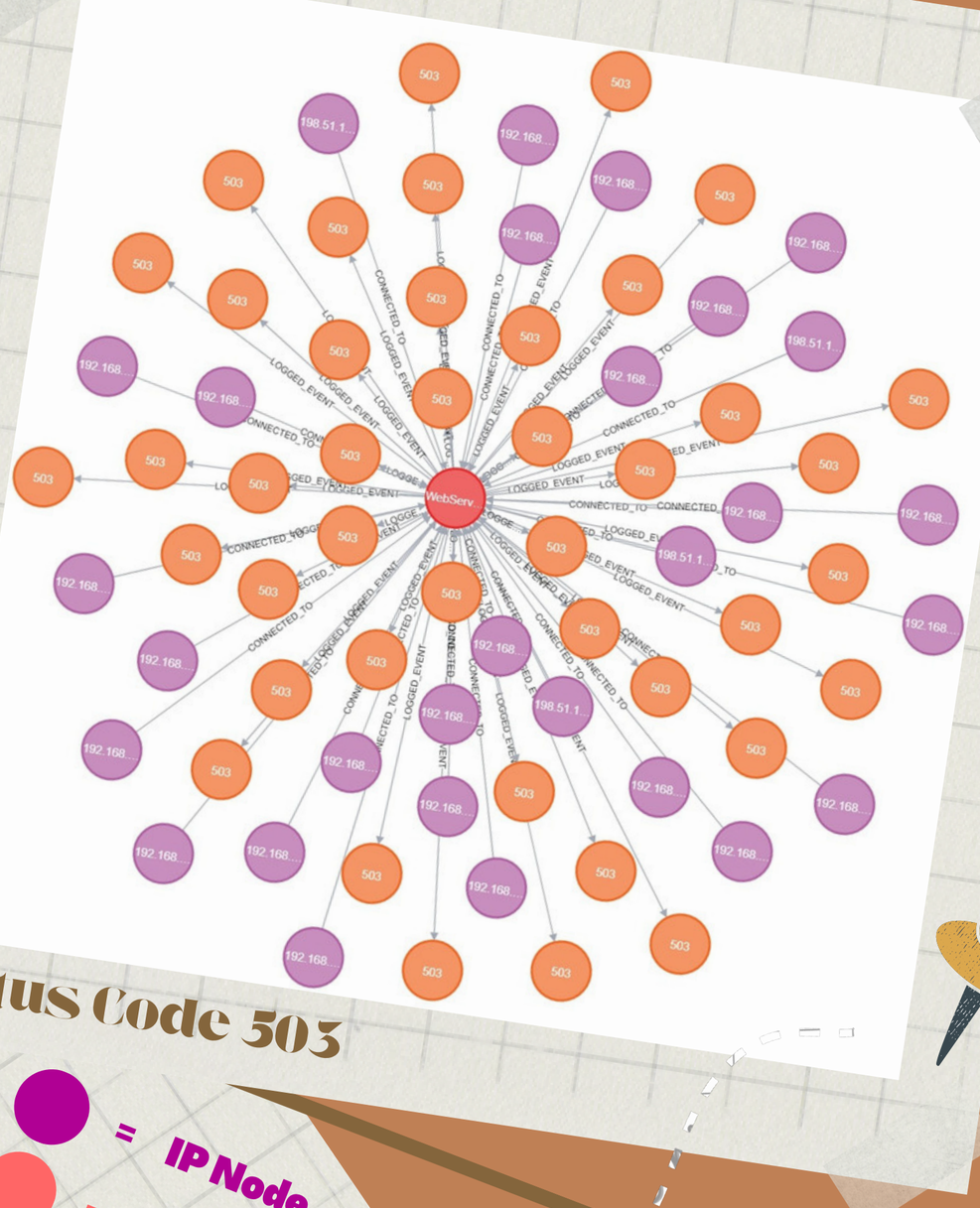


Threshold Analysis

= IP Node

= Server Node

= Status Node



Status Code 503

= IP Node

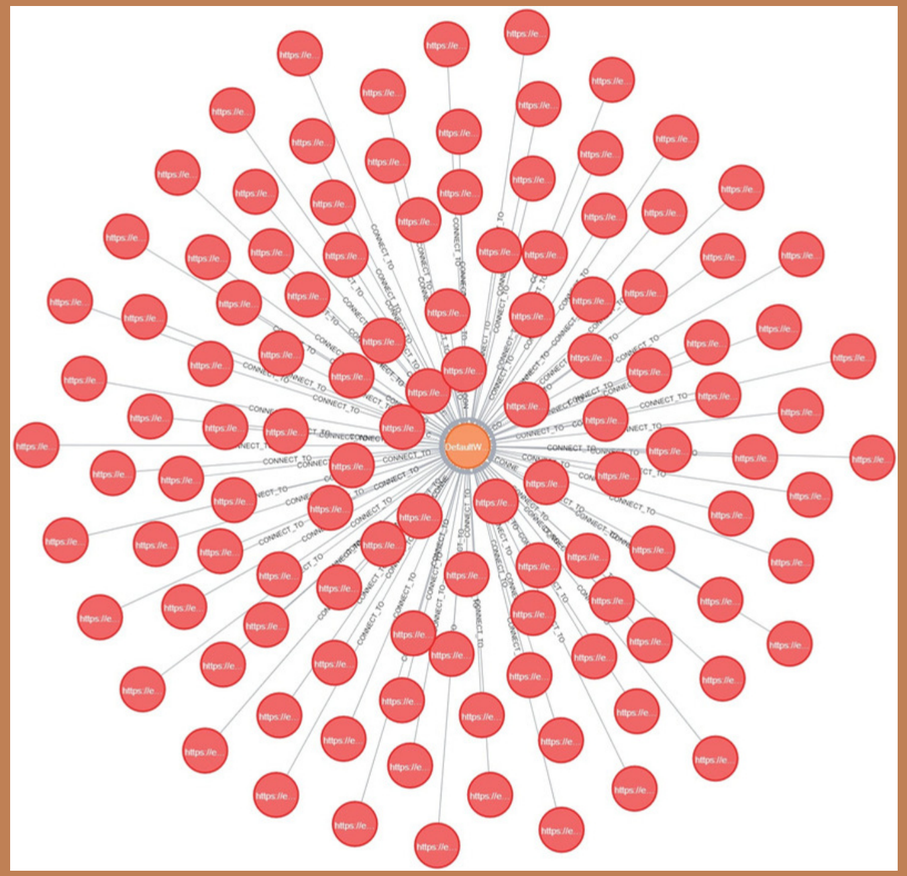
= Server Node

= Status Node

How can We detect XSS Attack?

```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)
```

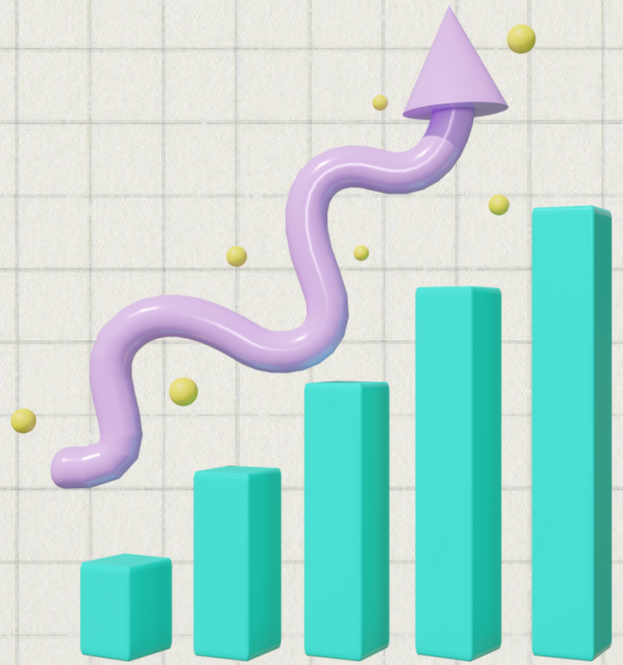
NEO4J SIMULATION



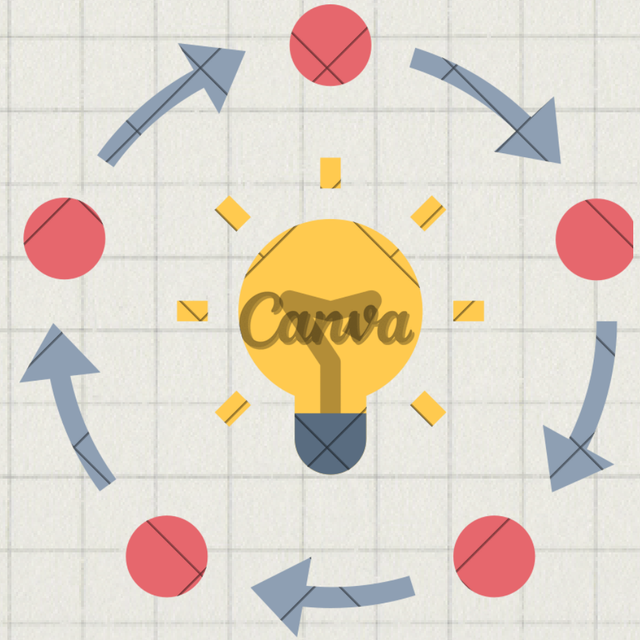
01 = Node with Script

02 = Server Node

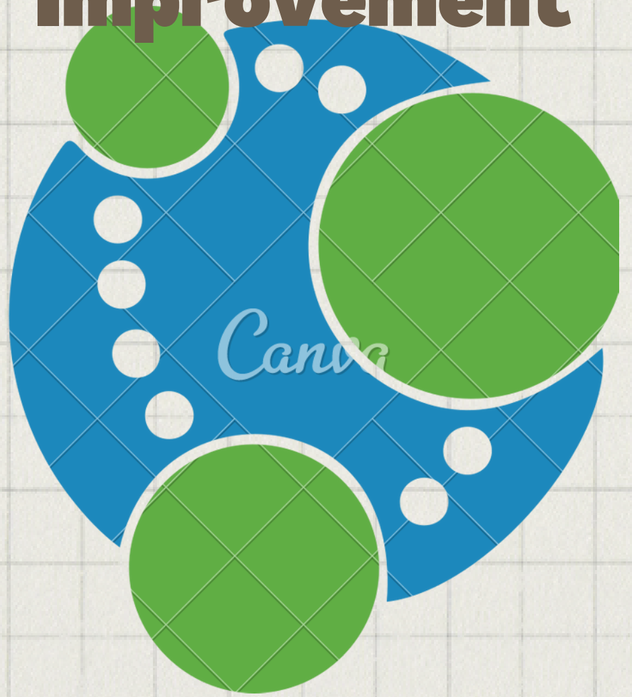
Logic of DDos & XSS Improvemnt



Log Realistic



Neo4J Simulation Improvement



Thank you for your attention