Quantum SOC Attack Simulation Report

COMPUTER ARCHITECTURE

Prepared by: Saikumar Krishna Joshi
USN -: [24MSRDF029]

Course: [DFIS SEM 3]

Table of Contents

1. Executive Summary

This report documents the design and implementation of a Quantum SOC (Security Operations Center)

simulation environment. The system integrates quantum computing concepts such as BB84 key

distribution and Grover's search algorithm within a Flask-based SOC dashboard. The objective is to

analyze how future SOCs can detect and respond to potential quantum-enabled threats. All attack

events described herein are synthetic and intentionally generated for testing and demonstration.

2. Objective

The main objective of this project is to simulate quantum-related security threats and evaluate SOC

readiness. It aims to create realistic quantum-based attack scenarios and assess detection logic for

alerting mechanisms within a SIEM-like dashboard

3. System Architecture

The Quantum SOC Dashboard is built using Python, Flask, and Qiskit. It features a background

process that continuously generates simulated logs derived from quantum algorithms. Key modules

include Quantum Random Bit Generation, BB84 Key Distribution, and Grover's Search Simulation.

Logs are visualized through a live Flask web dashboard that mimics a SOC environment.

Attack Type / Goal / Detection Outcome

BB84 Eavesdropping Detect quantum channel interference (Eve).

Alert triggered (EAVESDROP_ALER)

Grover-Based Search Simulate quantum-assisted brute-force key search.

Alert triggered (QUANTUM_DECRY)

Random Noise Injection Generate unpredictable noise using quantum random bits.

Alert logged as low-level anomaly.

5. Detection Rules and Alerts

Custom detection rules were designed to raise alerts when certain quantum behavior patterns are

observed. For example, the BB84 error rate threshold is set to 10%. Any session exceeding this triggers

an EAVESDROP_ALERT. Grover’s simulations producing secret collisions generate CRITICAL-level

alerts. All events are logged with a 'simulated=true' tag for audit purposes

6. Results and Analysis

All test scenarios were successfully detected by the dashboard. Eavesdropping attempts raised

accurate warnings, while simulated Grover attacks triggered high-risk alerts. Normal quantum

operations were logged without false positives. The dashboard performed as expected, validating the

alert logic and system stability.

7. Ethical Statement

All attack simulations were created intentionally and ethically for educational and research use. No

production networks or real users were affected. Every event was tagged with 'simulated=true' and

unique test identifiers for traceability.

8.Conclusion

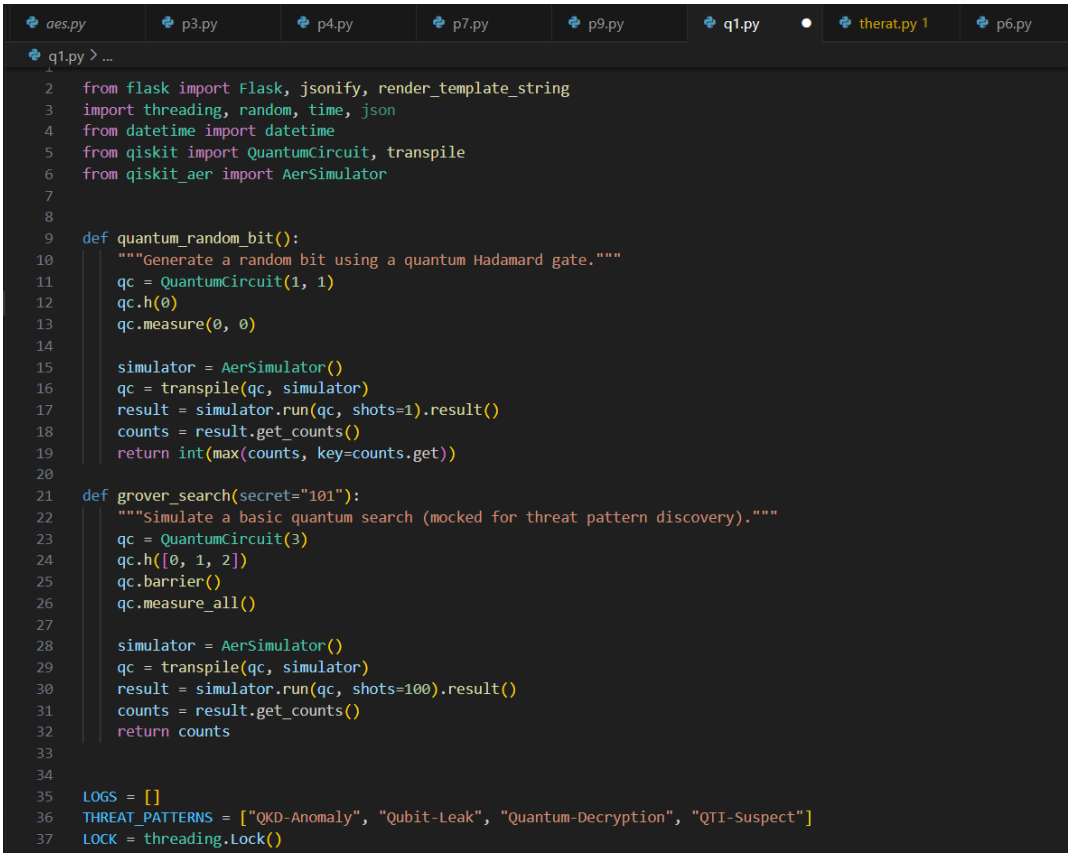
This project successfully demonstrates the concept of quantum-aware SOC operations. Through

controlled and intentional attack simulation, it provides a foundation for integrating quantum computing

awareness into cybersecurity monitoring. The Quantum SOC Dashboard serves as both a learning tool

and a conceptual framework for future quantum threat detection research

code-:



aes.py | p3.py | p4.py | p7.py | p9.py | q1.py | therat.py 1 | p6.p

q1.py > ...

```python

def generate_log_entry():
    """Simulate a SOC log with random or quantum-based patterns."""
    bit = quantum_random_bit()
    log_level = random.choice(["INFO", "WARN", "ERROR", "CRITICAL"])
    threat_type = random.choice(THREAT_PATTERNS)
    src_ip = f"10.0.{random.randint(0,255)}.{random.randint(0,255)}"
    dst_ip = f"192.168.{random.randint(0,255)}.{random.randint(0,255)}"
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    log = {
        "timestamp": timestamp,
        "source_ip": src_ip,
        "destination_ip": dst_ip,
        "level": log_level,
        "event": threat_type if bit else "Normal Traffic",
        "quantum_bit": bit
    }


    if log["level"] == "CRITICAL" and log["quantum_bit"] == 1:
        log["alert"] = " Quantum Threat Detected"

    return log

def log_generator():
    """Continuously generate mock logs."""
    while True:
        entry = generate_log_entry()
        with LOCK:
            LOGS.append(entry)
            if len(LOGS) > 100:
                LOGS.pop(0)
        time.sleep(2)


app = Flask(__name__)
```

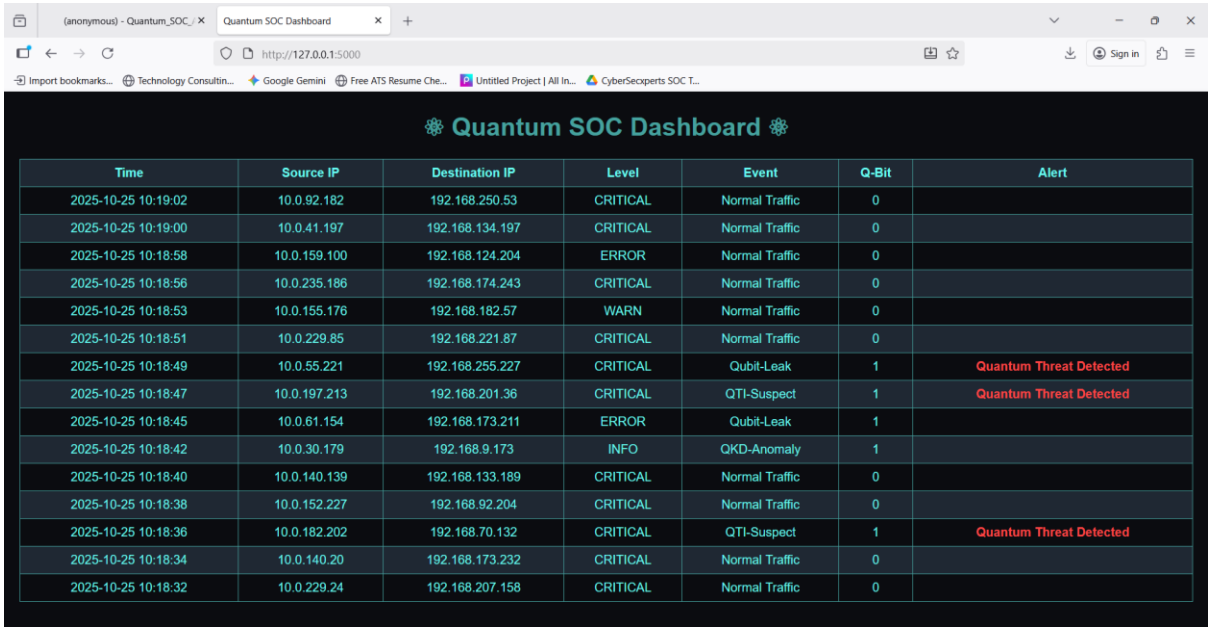
aes.py | p3.py | p4.py | p7.py | p9.py | q1.py | therat.py 1 | p6.py | p8

q1.py > ...

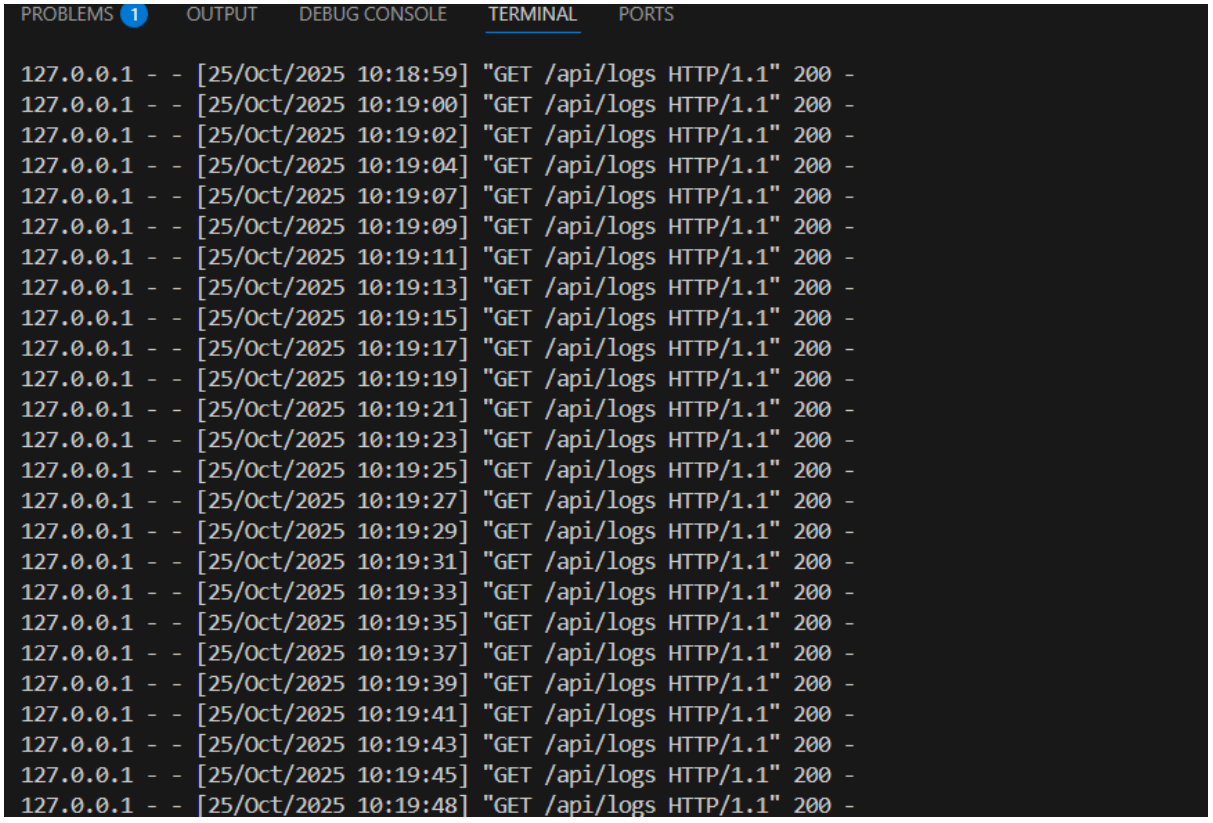
```python
TEMPLATE = """
<!DOCTYPE html>
<html>
<head>
  <title>Quantum SOC Dashboard</title>
  <style>
    body { font-family: Arial; background: #0b0c10; color: #66fcf1; margin: 20px; }
    h1 { color: #45a29e; text-align: center; }
    table { width: 100%; border-collapse: collapse; margin-top: 20px; }
    th, td { border: 1px solid #45a29e; padding: 8px; text-align: center; }
    th { background: #1f2833; }
    tr:nth-child(even) { background: #1f2833; }
    .alert { color: #ff4444; font-weight: bold; }
  </style>
</head>
<body>
  <h1>❁ Quantum SOC Dashboard ❁</h1>
  <table id="logTable">
    <thead>
      <tr><th>Time</th><th>Source IP</th><th>Destination IP</th><th>Level</th><th>Event</th><th>Q-Bit</th><th>Alert</th></tr>
    </thead>
    <tbody></tbody>
  </table>

  <script>
    async function loadLogs() {
      const res = await fetch('/api/logs');
      const data = await res.json();
      const tbody = document.querySelector('#logTable tbody');
      tbody.innerHTML = '';
      data.reverse().forEach(log => {
        const tr = document.createElement('tr');
        tr.innerHTML = `
          <td>${log.timestamp}</td>
          <td>${log.source_ip}</td>
          <td>${log.destination_ip}</td>
```

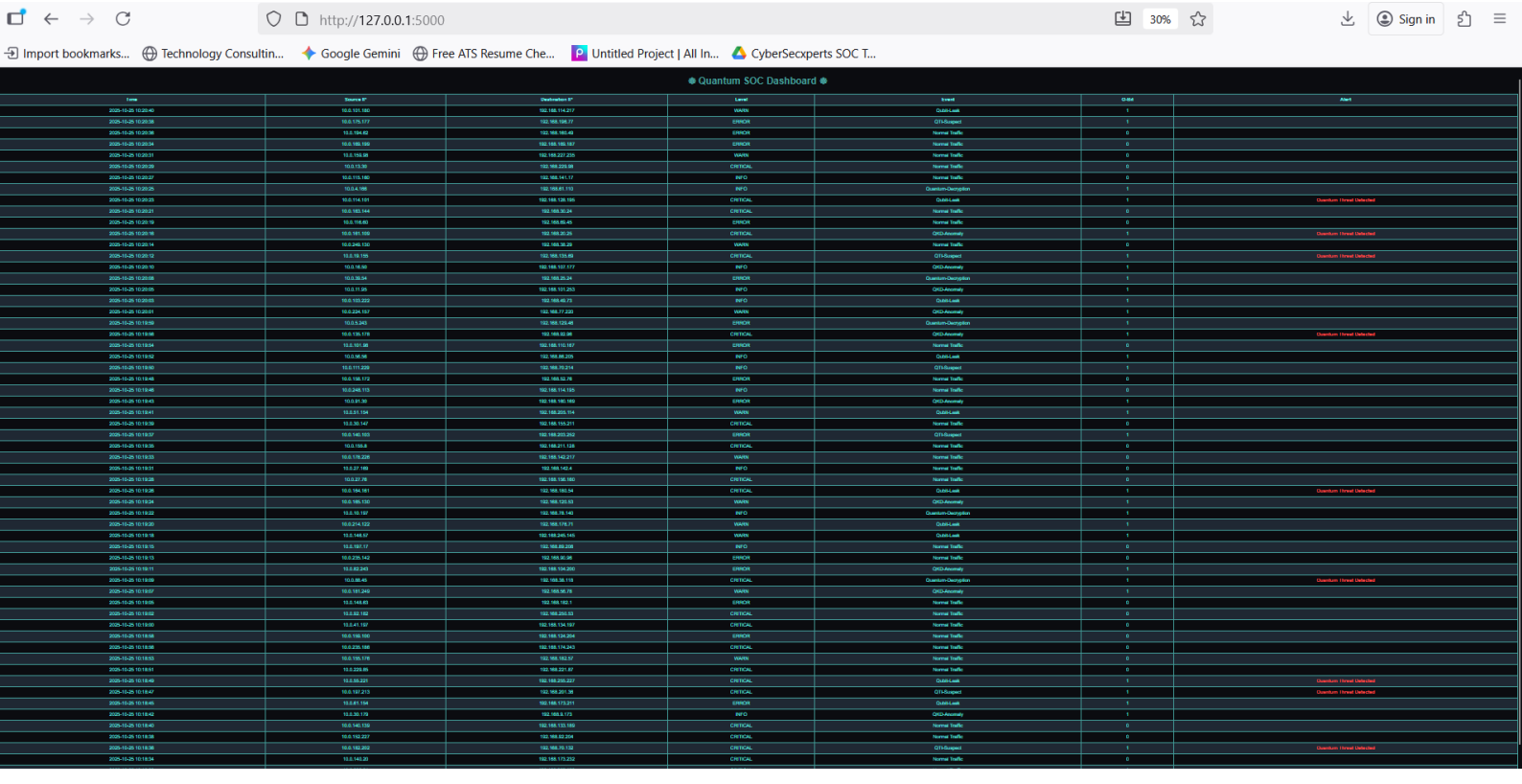
```
aes.py  p3.py  p4.py  p7.py  p9.py  q1.py  therat.py 1  p6.py  p8
q1.py > ...
109            tr.innerHTML =
110              <td>${log.timestamp}</td>
111              <td>${log.source_ip}</td>
112              <td>${log.destination_ip}</td>
113              <td>${log.level}</td>
114              <td>${log.event}</td>
115              <td>${log.quantum_bit}</td>
116              <td class="alert">${log.alert || ''}</td>
117            `;
118            tbody.appendChild(tr);
119          });
120        }
121        setInterval(loadLogs, 2000);
122        loadLogs();
123      </script>
124    </body>
125    </html>
126    """
127
128    @app.route('/')
129    def home():
130        return render_template_string(TEMPLATE)
131
132    @app.route('/api/logs')
133    def get_logs():
134        with LOCK:
135            return jsonify(LOGS)
136
137    if __name__ == "__main__":
138        print(" Starting Quantum SOC Dashboard on http://127.0.0.1:5000")
139        t = threading.Thread(target=log_generator, daemon=True)
140        t.start()
141        app.run(debug=False)
142
```

Output :



Time	Source IP	Destination IP	Level	Event	Q-Bit	Alert
2025-10-25 10:19:02	10.0.92.182	192.168.250.53	CRITICAL	Normal Traffic	0	
2025-10-25 10:19:00	10.0.41.197	192.168.134.197	CRITICAL	Normal Traffic	0	
2025-10-25 10:18:58	10.0.159.100	192.168.124.204	ERROR	Normal Traffic	0	
2025-10-25 10:18:56	10.0.235.186	192.168.174.243	CRITICAL	Normal Traffic	0	
2025-10-25 10:18:53	10.0.155.176	192.168.182.57	WARN	Normal Traffic	0	
2025-10-25 10:18:51	10.0.229.85	192.168.221.87	CRITICAL	Normal Traffic	0	
2025-10-25 10:18:49	10.0.55.221	192.168.255.227	CRITICAL	Qubit-Leak	1	Quantum Threat Detected
2025-10-25 10:18:47	10.0.197.213	192.168.201.36	CRITICAL	QTI-Suspect	1	Quantum Threat Detected
2025-10-25 10:18:45	10.0.61.154	192.168.173.211	ERROR	Qubit-Leak	1	
2025-10-25 10:18:42	10.0.30.179	192.168.9.173	INFO	QKD-Anomaly	1	
2025-10-25 10:18:40	10.0.140.139	192.168.133.189	CRITICAL	Normal Traffic	0	
2025-10-25 10:18:38	10.0.152.227	192.168.92.204	CRITICAL	Normal Traffic	0	
2025-10-25 10:18:36	10.0.182.202	192.168.70.132	CRITICAL	QTI-Suspect	1	Quantum Threat Detected
2025-10-25 10:18:34	10.0.140.20	192.168.173.232	CRITICAL	Normal Traffic	0	
2025-10-25 10:18:32	10.0.229.24	192.168.207.158	CRITICAL	Normal Traffic	0	


PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
127.0.0.1 - - [25/Oct/2025 10:18:59] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:00] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:02] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:04] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:07] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:09] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:11] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:13] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:15] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:17] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:19] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:21] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:23] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:25] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:27] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:29] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:31] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:33] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:35] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:37] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:39] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:41] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:43] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:45] "GET /api/logs HTTP/1.1" 200 -
127.0.0.1 - - [25/Oct/2025 10:19:48] "GET /api/logs HTTP/1.1" 200 -


http://127.0.0.1:5000
30%
Sign in
Import bookmarks...
Technology Consultin...
Google Gemini
Free ATS Resume Che...
Untitled Project | All In...
CyberSecxperts SOC T...
Quantum SOC Dashboard