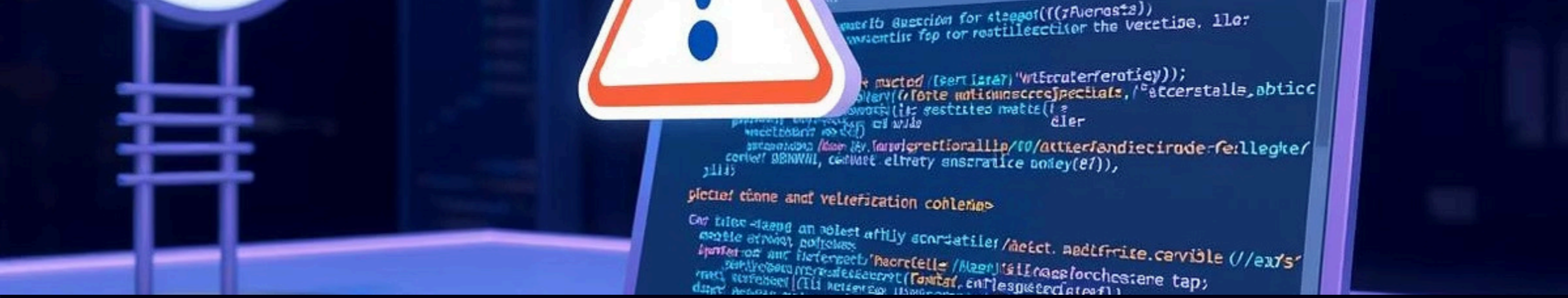
CVE-2025-3248: Critical Vulnerability in Langflow

Hello, I’m Iman Hajibagheri

Github username: ***imanhajibagheri***

EDX username: ***imanhajibagheri***

presenting on CVE-2025-3248, a critical remote code execution vulnerability discovered in June 2025. This case reveals the risks of improper input validation in AI-driven workflows, a vital concern for secure software development.



Overview of CVE-2025-3248

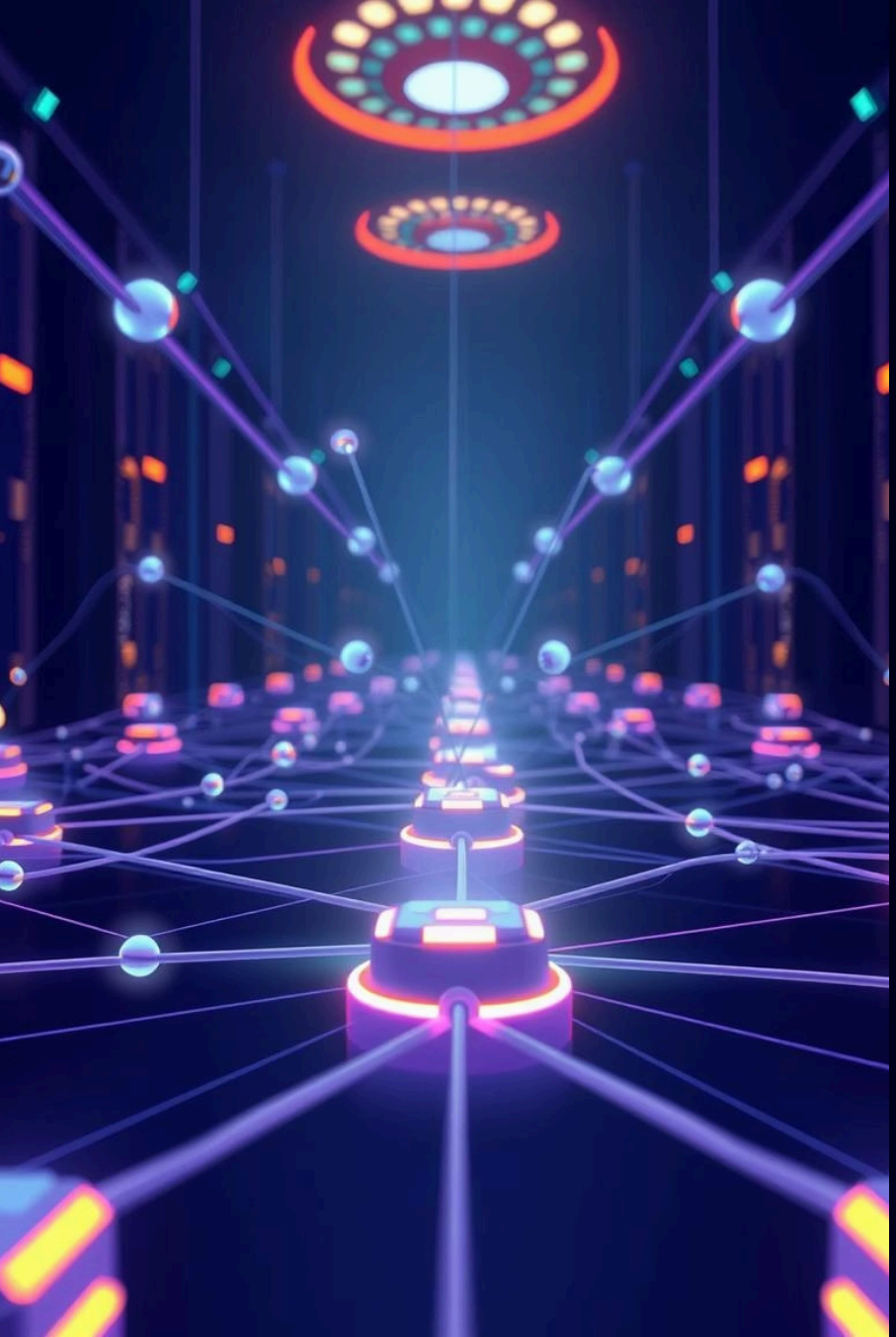
CVE-2025-3248 affects Langflow versions before 1.3.0 with a CVSS score of 9.8, indicating critical severity. It targets the `/api/v1/validate/code` endpoint, allowing unauthenticated attackers to execute arbitrary Python code remotely via crafted HTTP requests. This exploit was actively used by the Flodrix botnet, leading to DDoS attacks and data theft.

Technical Details of the Vulnerability

Langflow enables no-code AI workflow creation. The critical flaw lies in using Python's `exec()` on unsanitized input without authentication at `/api/v1/validate/code`. Example vulnerable code:

```
@router.post("/validate/code")
async def validate_code(request: Request):
    body = await request.json()
    code = body.get("code")
    exec(code)  # Unsafe execution of untrusted code
    return {"valid": True}
```

Malicious input allows file reads, reverse shells, or installing malware like Flodrix.



Real-World Impact

Exploitation led to full system compromise, large-scale DDoS, and data breaches. Langflow’s popularity in AI prototyping exposed many public instances. EPSS rated probability of exploitation at 92.76% within 30 days, emphasizing high exploitation risk and urgency for patching.

Relation to Course Topics

Highlights the dangers of insecure deserialization and unsanitized input validation.

Emphasizes missing authentication measures in API security.

Demonstrates real-world attack vectors and the urgent need for timely patches.

Recommendations for Mitigation

Upgrade Langflow

Use version 1.3.0+ with authentication on vulnerable endpoints.

Restrict Access

Implement Zero Trust Network Architecture and network segmentation.

Avoid Unsafe Functions

Replace `exec()` with AST parsing and safe evaluators.

Validate & Sanitize

Robustly check all user inputs to prevent injection attacks.

Implement Monitoring

Use WAFs to detect anomalous code submissions.

Maintain Patching

Keep patching programs active for timely software updates.

Secure Coding Lessons from CVE-2025-3248

This vulnerability underscores the fatal consequences of unsafe input handling and missing authentication. It reinforces secure programming fundamentals: never trust user input, and always authenticate sensitive API endpoints — especially in AI platforms where code execution has broad impact.

Understanding the Flodrix Botnet Exploit

The Flodrix botnet leveraged the exploit to deploy malware for DDoS attacks and data exfiltration. Once inside, attackers achieved persistent control over systems running vulnerable Langflow versions, amplifying the scale and impact of attacks globally.

Timely Patching and Incident Response

With public proof-of-concept exploits widely available, the importance of rapid patch deployment cannot be overstated. Integrating active monitoring, incident detection, and automated patching in software lifecycle mitigates risks from such high-severity vulnerabilities.

Key Takeaways

Secure Input Validation

Never execute untrusted user input without strict sanitization and authentication.

Authentication is Essential

Protect API endpoints, especially those executing code or handling sensitive data.

Proactive Defense

Use layered security including Zero Trust, WAFs, and timely patching.

CVE-2025-3248 is a cautionary example highlighting how insecure coding can lead to severe compromise. Vigilance, solid secure development practices, and monitoring are critical in defending modern AI platforms.

Thank you for reading!