CHANDIGARH UNIVERSITY, GHARUAN

Department of Computer Science and Engineering

INTERNSHIP REPORT

Submitted in Partial Fulfilment of the Requirements for the
Degree of
Bachelor of Engineering (B.E.) in Computer Science and Engineering

Internship Program (20 Sep-25 Dec):

Cybersecurity Internship Program – Shentinelix Sphere Pvt Ltd

Task 18

SQL Injection Attacks And Windows x64 Assembly

Internship Report Submitted By:
Vikash Upadhyay

Table of Contents

Executive Summary

I completed a dual-track training module that combined practical offensive techniques for web-application database exploitation (SQL Injection) with foundational reverse-engineering skills on Windows x64 assembly. The SQL Injection track focused on how attackers exploit unsanitized inputs to manipulate database queries, taught automated exploitation with tools like sqlmap and jSQL, and taught manual injection techniques and dictionary creation (Crunch) useful for brute-force or password-guessing scenarios. The Windows x64 Assembly track provided conceptual grounding in reverse engineering — CPU registers, memory layout, calling conventions, and low-level program logic — preparing me to read compiled binaries and reason about vulnerabilities.

Outcomes include demonstrable competency in performing controlled SQL injection tests in lab environments, building dictionaries, and using both automated and manual exploitation approaches; and an improved ability to inspect simple x64 binaries, follow control flow, and map assembly instructions back to higher-level logic. Deliverables produced: step-by-step lab logs, reproduction captures, a set of Crunch dictionary commands/templates, an SOP for safe database testing, and annotated assembly walkthroughs. The combined training strengthened my offensive toolset while improving my ability to reason about defensive mitigations and secure coding practices.

This module reinforced ethical, legal boundaries and the practical reality that discovering a vulnerability is only the first step — responsible disclosure, accurate reporting, and suggested remediations are crucial. It also highlighted operational constraints (test scope, environment fidelity, and hardware/resources) that affect the success of both injection testing and binary analysis.

Introduction

The SQL Injection portion introduced the threat model: how user-controlled inputs that reach database query construction can lead to data exfiltration, unauthorized data modification, authentication bypass, and remote code execution in extreme cases. The curriculum balanced automated scanning (sqlmap, jSQL) with manual techniques (crafting payloads, boolean/time-based injection) to teach learners both speed and deep understanding — speed to triage at scale and manual skills to confirm, exploit, and demonstrate impact responsibly.

The Windows x64 Assembly track emphasized why reverse engineering matters for security: many vulnerabilities (logic flaws, insecure crypto usage, improper input validation) are discovered by reading compiled code when source is unavailable. Lessons covered binary anatomy, registers, instructions, flags, memory layout, calling conventions, and simple reversing exercises using tools that are common in malware analysis and vulnerability research.

Together, these topics help a security practitioner think across layers: from web-app input sanitization and database query logic to how compiled application behavior at the CPU/instruction level can introduce vulnerabilities. The combined perspective improves both

offensive testing (finding complex bugs that surface through interactions between components) and defensive guidance (code fixes, compiler/hardening recommendations).

Overview of the Company

Shentinelix Sphere Pvt. Ltd. is a cybersecurity training and solutions provider that specializes in delivering practical exposure in information security, penetration testing, and cyber defense.

The company focuses on empowering students and professionals with real-world skills through structured labs, Capture-the-Flag (CTF) style challenges, and guided mentorship.

During the internship, I was assigned to the Cybersecurity Research and Training Department The organization strongly emphasizes hands-on learning by offering lab environments, curated challenges, and continuous support from experienced security practitioners, ensuring that interns develop both technical competence and a strong research methodology.

Description of Duties

I performed structured reconnaissance of lab web applications to identify potential injection points (forms, URL parameters, headers). I ran targeted sqlmap scans with safe flags to enumerate databases, tables, and columns in the lab target. Where automation suggested potential issues, I manually validated those findings using crafted payloads (union-based, boolean-based, time-based) to confirm results and measure impact. I created dictionaries using Crunch for scenarios requiring credential/parameter brute forcing.

For manual testing I practiced payload construction, error message analysis, and blind-injection techniques (timing/boolean) to extract data when direct errors were suppressed. I used jSQL and other GUI tools for quick verification, combined with command-line sqlmap for reproducible exploitations and reporting artifacts.

I read simple x64 binaries, traced function calls, examined stack/heap usage, and interpreted register state across calling conventions. I used debuggers/disassemblers to step through small programs, identify control-flow branches, and map assembly constructs to higher-level logic. I documented findings and linked observed behavior to potential security weaknesses (e.g., insecure string handling, implicit trust of inputs).

Quick duty points:

- Identify inputs and parameters for SQLi testing.
- Run sqlmap with safe enumeration flags; confirm manually.
- Craft manual payloads: ' OR '1'='1' -- , UNION SELECT templates, time-based payloads.
- Generate dictionaries: crunch <min> <max> <charset> -o dict.lst.
- Disassemble simple x64 binaries; follow calling conventions and annotate functions.

Acomplishments
Successfully identified and validated multiple injection vectors in lab applications, enumerated schema objects in authorized test targets, and demonstrated data extraction using both automated (sqlmap) and manual payloads. Produced reproducible attack logs and PoC scripts for safe reporting.
Constructed tailored dictionaries with Crunch for credential/password guessing scenarios relevant to the labs. Used jSQL to quickly map vulnerabilities in small apps and extract data where CLI tooling was less convenient, documenting outputs for reporting.

Completed step-through analysis of provided x64 sample binaries, identified function boundaries, and explained control flow for a simple program (parameters → stack frame → return). Created annotated notes showing how certain assembly patterns could correspond to insecure code practices (e.g., lack of input length checks before memory operations).

Accomplishment bullets:

- End-to-end sqlmap exploitation on lab targets with manual validation.
- Crafted effective Crunch dictionaries for targeted brute-force.
- Cracked/verified simple login flows in lab using dict-based tests.
- Completed assembly walkthroughs and annotated binaries for reporting.

Skills Learned
Practical use of sqlmap and jSQL; manual SQL injection techniques including union-based, boolean-based, error-based, and time-based methods; safe lab testing practices and how to craft precise, minimally-destructive proof-of-concept exploits.

Building dictionaries with Crunch, integrating output into brute-force pipelines, and combining automated discovery with manual confirmation to avoid false positives. Understanding when to escalate an automated finding to manual analysis.x64 assembly basics — register usage (RAX, RBX, RCX, RDX, RSP, RBP, RSI, RDI), stack frames, calling conventions (Windows x64), flags, and instruction patterns. Practical skills in stepping through binaries, identifying branches, and mapping assembly to higher-level logic for vulnerability discovery.

Skills bullet list:

- sqlmap usage and tuning (risk/safety flags, tamper options).
- Manual payload construction and blind injection strategies.
- Crunch dictionary construction and integration.
- Basic x64 reversing: disassembly, function tracing, and stack analysis.
- Reporting: PoC generation, remediation recommendations.

Challenges Faced

Automated tools like sqlmap can produce false positives or flood logs; distinguishing real issues required careful manual validation to avoid misreporting. Fine-tuning requests and understanding web app behavior was necessary.

Many modern apps use parameterized queries, WAFs, or input filtering that obscure direct feedback and frustrate blind injection attempts. Time-based and out-of-band methods were slower and required patience. Similarly, lack of error messages often forced more involved extraction techniques.

Reading assembly is initially slow and requires mental mapping from low-level instructions to program semantics. Identifying where source-level vulnerabilities originate demands practice. Tools help, but interpreting compiler-optimized code was sometimes difficult.

Challenge bullets:

- Managing noisy sqlmap output and tuning to avoid detection.
- Performing blind SQLi when error messages were suppressed or WAFs mitigated direct injection.
- Mapping optimized compiler output back to readable logic in x64 assembly.
- Resource/time constraints for exhaustive brute-force or deep binary analysis.

Conclusion

This combined module delivered practical, ethically-framed training in two complementary cybersecurity domains: web/database exploitation (SQLi) and low-level software understanding (Windows x64 assembly). I gained hands-on experience that improves my ability to find, confirm, and responsibly report SQL injection vulnerabilities, and foundational reversing skills that help reason about software behavior when source code is unavailable. The training emphasized translating offensive capabilities into defensive recommendations — e.g., parameterized queries/ORMs, prepared statements, WAF tuning, input validation, least privilege, and developer education for SQLi; and secure coding practices and compiler-based hardening for binary-level issues.

Recommended next steps: practice timed blind-extraction techniques in controlled labs, explore advanced sqlmap tamper options and WAF evasion research responsibly, deepen assembly knowledge with progressively harder binaries and real-world samples, and study mitigations (parameterized queries, stored procedures, prepared statements, input validation, static analysis tools). Finally, maintain strict adherence to legal/ethical rules: only test with permission, log every action, and provide clear remediation guidance to stakeholders.

Acknowledgments

I would like to thank the team at Shentinelix Sphere Pvt. Ltd. for providing access to lab environments, learning resources, and a supportive environment that encouraged practical learning and skill development. Special thanks to the TryHackMe platform and the authors of

the Research Methodology room for providing structured, hands-on lessons that significantly enhanced my research and problem- solving skills. Finally, I extend my gratitude to my peers and family for their continuous support and encouragement throughout the internship period.