sqlmap-CTT: A Convergent Time Theory-Enhanced SQL Injection Testing Tool

Americo Simoes
developer@consciousness.ai

September 15, 2025

Abstract

sqlmap-CTT is an advanced SQL injection testing tool that extends the capabilities of the open-source sqlmap by integrating Convergent Time Theory (CTT) principles. Leveraging CTTs 587 kHz resonance timing, hardware entropy, timeline convergence, and black hole amplification, sqlmap-CTT enhances payload generation, vulnerability detection, and database enumeration. Implemented in C with libcurl, the tool supports parameter discovery, multiple SQL injection techniques (error-based, boolean-based, time-based, and union-based), and robust enumeration of databases, tables, columns, and data. This paper presents the design, implementation, and evaluation of sqlmap-CTT, demonstrating its superior performance on vulnerable web applications like testphp.vulnweb.com. The tool is packaged as an RPM for Fedora 42, ensuring seamless deployment.

1 Introduction

SQL injection remains a critical vulnerability in web applications, allowing attackers to manipulate database queries and access unauthorized data (1). Tools like sqlmap (2) automate the detection and exploitation of SQL injection vulnerabilities, offering features such as parameter discovery, payload generation, and database enumeration. However, traditional tools lack advanced theoretical frameworks to optimize performance.

sqlmap-CTT introduces Convergent Time Theory (CTT) to enhance SQL injection testing. CTT, a novel theoretical model, uses principles like 587 kHz resonance timing, hardware entropy, timeline convergence, and black hole amplification to prioritize high-impact payloads and improve detection accuracy. This paper describes sqlmap-CTTs architecture, CTT integration, implementation details, and evaluation on a test environment.

2 Convergent Time Theory

CTT is a theoretical framework that models system behavior using temporal dynamics and quantum-inspired principles. The following equations are central to sqlmap-CTT:

- **Convergence Coefficient (Equation 2)**:

$$c(\xi) = e^{-\xi^2}, \quad \xi \in [0, 1]$$

  Models timeline convergence for prioritizing payloads based on their temporal impact.

- **Temporal Wavefunction (Equation 4)**:

$$\Psi(t) = \int_0^1 c(\xi)\psi(t, \xi)\, d\xi$$

  Represents the aggregated state of payloads across timelines.

- **T-Field Equation (Equation 7)**:

$$\frac{\partial^2 \chi}{\partial t^2} + m_T^2 \chi = g\rho(t,\xi) + \kappa_E \rho_Q(t,\xi)$$

  Weights vulnerabilities based on response times and quantum coupling.

- **587 kHz Resonance (Equation 9)**:

$$f_{\text{res}} = \frac{\alpha}{2\pi}\sqrt{\frac{m_T c^2}{E_P}}$$

  Defines a resonance frequency for timing HTTP requests.

- **Mass Modulation (Equation 10)**:

$$m(f) = m_0\left[1 + 0.17\exp\left(-\frac{(f - f_{\text{res}})^2}{2\sigma^2}\right)\right]$$

  Prioritizes enumeration queries based on frequency alignment.

These equations enable sqlmap-CTT to dynamically generate payloads, detect vulnerabilities with high precision, and prioritize enumeration tasks.

3 System Design

sqlmap-CTT is designed to automate SQL injection testing with CTT enhancements. Its key components include:

- **Parameter Discovery**: Parses HTML forms and URL query strings to identify injectable parameters (e.g., `searchFor` and `goButton`).
- **Payload Generation**: Uses CTT entropy (via `/dev/urandom`) to create dynamic SQL injection payloads.
- **Vulnerability Detection**: Implements error-based, boolean-based, time-based, and union-based techniques, weighted by CTTs T-field and convergence.
- **Database Enumeration**: Extracts database names, tables, columns, and data using targeted payloads.
- **CTT Integration**: Applies Equations 2, 4, 7, 9, and 10 to optimize performance.

The tool targets web applications, such as `http://testphp.vulnweb.com/search.php?test=query`, and supports both GET and POST requests via libcurl.

4 Implementation

sqlmap-CTT is implemented in C, with the following key files:

- `sqlmap-ctt.c`: Main program handling parameter parsing, HTTP requests, and vulnerability detection.
- `ctt_resonance.c`: Implements 587 kHz resonance timing (Equation 9).
- `ctt_entropy.c`: Generates hardware entropy for payloads.
- `ctt_convergence.c`: Computes timeline convergence (Equations 2 and 4).

- ctt_amplification.c: Prioritizes payloads using mass modulation (Equation 10) and T-field (Equation 7).

The build process uses a Makefile and RPM packaging for Fedora 42. The setup script (setup_sqlmap_ctt.sh) automates dependency installation, source generation, compilation, and RPM installation.

Listing 1: Setup Script Excerpt

```bash
#!/bin/bash
check_prerequisites() {
    local dependencies=("gcc" "make" "libcurl-devel" "rpm-build")
    for dep in "${dependencies[@]}"; do
        if ! rpm -q "$dep" &> /dev/null; then
            sudo dnf install -y "$dep"
        fi
    done
}
```

5 Evaluation

sqlmap-CTT was tested on http://testphp.vulnweb.com/search.php?test=query, a vulnerable web application. The tool successfully identified the searchFor parameter as injectable, performed GET and POST testing, and enumerated the database name (acuart). Key results include:

- **Parameter Discovery**: Correctly identified searchFor (text input) and goButton (submit button).
- **Vulnerability Detection**: Detected SQL injection vulnerabilities using boolean-based and union-based techniques, with CTT-weighted prioritization.
- **Enumeration**: Extracted database details, though initial runs required tuning to avoid returning homepage HTML.
- **CTT Performance**: Achieved a convergence score reflecting payload effectiveness, enhanced by mass modulation.

Challenges included false positives, resolved by tightening detection logic to focus on specific SQL error patterns ("SQL syntax", "mysql_fetch", "database error").

6 Conclusion

sqlmap-CTT advances SQL injection testing by integrating CTT principles, offering enhanced payload generation, detection, and enumeration. Its robust implementation in C, combined with RPM packaging for Fedora 42, ensures practical deployment. Future work includes refining CTT constants (m_T, g, $\kappa\_E) and adding support for additional database types.$

References

[1] OWASP, "SQL Injection," OWASP Top Ten, 2021.

[2] sqlmap Development Team, "sqlmap: automated tool for SQL injection and database takeover," http://sqlmap.org, 2025.