Interactive Security Testing Framework

A comprehensive security testing framework with modular architecture, designed for security professionals, penetration testers, and system administrators.

1. Features

* Modular Plugin System: Easily extend with new scanning tools
* Multiple Testing Profiles: Network, vulnerability, exploitation, anonymity, auditing, wireless
* Database Integration: Store and query test results and discovered vulnerabilities
* Enhanced Reporting: Generate comprehensive HTML reports with vulnerability analysis
* Stealth Mode: Configure tests for minimal detection
* Anonymity Options: Integrate with Tor and AnonSurf for anonymous testing
* CVE Database: Automatically fetch and cache vulnerability details

1. Prerequisites

* Python 3.6+
* Various security tools (nmap, nikto, metasploit, etc.)
* Suitable permissions (root/admin for many scanner functions)

1. Installation
   1. Clone the repository:

*git clone https://github.com/yourusername/security-testing-framework.git*

*cd security-testing-framework*

* 1. Set up the environment:

*python setup\_environment.py --dependencies*

* 1. Install Python dependencies:

*pip install -r requirements.txt*

1. Usage

Interactive System Tester

Run the main interactive tester with:

*sudo python interactive\_system\_tester.py*

The interactive interface will guide you through:

1. Configuring test settings
2. Selecting targets
3. Running scans
4. Viewing and exporting results

Automated Query Tool

For database-focused operations:

*python automatedquerytool.py*

This tool focuses on querying and analyzing previously collected data.

1. Plugin System

Available Plugins

* NmapScanner: Network discovery and vulnerability scanning
* MasscanScanner: Fast port scanning
* NetcatScanner: Basic connectivity testing
* NiktoScanner: Web server vulnerability scanning
* SQLMapScanner: SQL injection testing
* MetasploitScanner: Exploitation framework integration
* LynisScanner: System auditing
* ChkrootkitScanner: Rootkit detection
* JohnScanner: Password cracking
* AnonSurfScanner: Anonymity verification
* AircrackScanner: Wireless network security testing

1. Creating a New Plugin
   1. Create a new file in the scanners directory, e.g., my\_scanner.py
   2. Implement the BaseScanner interface:

|  |
| --- |
| from scanners import BaseScanner  class MyScanner(BaseScanner):  def scan(self, target):  # Configure options from self.options  verbose = self.options.get('verbose', False)    # Run your scan logic  # ...    # Return structured results  return {  'vulnerabilities': [], # List of vulnerabilities found  'raw\_output': '', # Raw output from the tool  'command': '' # Command that was executed  } |

* 1. The scanner will automatically be discovered and available for use

1. Extending Functionality

Adding a New Profile

1. Update the completer and profile lists in setup\_prompt method
2. Add profile-specific scanners in run\_profile method
3. Update the profile selection logic in configure\_settings

Adding New Report Types

1. Create a new report generator in the reporting directory
2. Integrate it with the export\_reports method

Database Schema

The system uses SQLite with the following tables:

* targets: Discovered systems (IP, MAC, OS, etc.)
* ports: Open ports and services for each target
* test\_results: Results from each test run
* vulnerabilities: Detailed vulnerability information

1. Contributing

Contributions are welcome! Please follow these steps:

1. Fork the repository
2. Create a feature branch: *git checkout -b feature/my-feature*
3. Commit your changes: *git commit -am 'Add my feature'*
4. Push to the branch: *git push origin feature/my-feature*
5. Submit a pull request

License

This project is licensed under the MIT License - see the LICENSE file for details.

Acknowledgments

* Thanks to all the open-source security tools that make this framework possible
* Special thanks to contributors and testers