VulnHunter

Professional User Manual

Enterprise-Grade Vulnerability Assessment and Penetration Testing Framework

Document Version	2.0
Publication Date	July 26, 2025
Classification	Internal Use Only
Target Audience	Security Professionals, Penetration Testers, IT Security Teams
Document Type	User Manual
Total Pages	Auto-Generated

Developed for cybersecurity professionals, by cybersecurity professionals.

VulnHunter Professional User Manual

Enterprise-Grade Vulnerability Assessment and Penetration Testing Framework

Document Version: 2.1
Publication Date: July 26, 2025
Document Classification: Internal Use
Target Audience: Security Professionals, Penetration Testers, IT Security Teams

Table of Contents

1. Executive Summary

1.1 About VulnHunter

VulnHunter is a comprehensive security testing framework designed for professional vulnerability assessment and penetration testing (VAPT) across multiple attack surfaces. Developed for enterprise environments, it provides automated security testing capabilities while maintaining the flexibility needed for manual verification and custom testing scenarios.

1.2 Key Features

- **Enterprise Authentication**: Two-factor authentication (2FA) with email OTP verification

- **Multi-Domain Testing**: Network, Web Application, Cloud, and API security assessments
- **Advanced Integration**: 3,758 Metasploit modules and 46,000+ Exploit-DB entries
- **Professional Reporting**: Executive summaries, technical findings, and remediation roadmaps
- **Automated Tool Management**: Intelligent dependency checking and installation
- **Enterprise Ready**: Session persistence, evidence collection, and compliance reporting

1.3 Target Users

- **Security Consultants**: Professional penetration testers and security auditors
- **Enterprise IT Teams**: Internal security teams conducting regular assessments
- **Compliance Officers**: Organizations requiring security compliance validation
- **Educational Institutions**: Cybersecurity training and research environments

2. Getting Started

2.1 System Requirements

Minimum Requirements: - Operating System: Linux/Unix (Ubuntu 20.04+, Debian 10+, Kali Linux) - Python: Version 3.11 or higher - Memory: 4GB RAM (8GB recommended) - Storage: 2GB free space (5GB recommended for full tool installation) - Network: Internet connection for tool downloads and updates

Recommended Environment: - Kali Linux 2023.1+ or Ubuntu 22.04 LTS - Python 3.11+ with pip3 - 8GB+ RAM for large-scale assessments - SSD storage for improved performance - Dedicated testing network or VPN access

2.2 Pre-Installation Checklist

Before installing VulnHunter, ensure you have:

- [ ] Administrative/root access to the system
- [ ] Updated system packages (`sudo apt update && sudo apt upgrade`)
- [ ] Python 3.11+ installed and configured
- [ ] Git client installed for repository access
- [ ] Network connectivity for dependency downloads
- [ ] Appropriate authorization for target testing

3. Installation Guide

3.1 Quick Installation

Step 1: Clone the Repository

```
git clone https://github.com/narain-karthik/VulnHunter.git
cd VulnHunter
```

Step 2: Install Core Dependencies

```
pip3 install colorama jinja2 tabulate requests reportlab weasyprint markdown
```

Step 3: Launch VulnHunter

```
python3 main.py
```

Step 4: Auto-Install Security Tools

```
# From the main menu, select:
# Option 6: Auto-Install Missing Tools
# Option 1: Auto-install missing tools
```

3.2 Advanced Installation

System Package Installation:

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install core security tools
sudo apt install -y nmap masscan nikto dirb sqlmap jq wget curl \
                   dnsutils netcat-openbsd openssl sslscan

# Install Python development packages
sudo apt install -y python3-dev python3-pip python3-venv build-essential
```

Python Dependencies:

```
# Install all required Python packages
pip3 install colorama jinja2 tabulate requests reportlab weasyprint \
             awscli scoutsuite trafilatura dnspython beautifulsoup4 \
             paramiko scapy python-nmap
```

Verification:

```
# Verify installation
python3 main.py --help
python3 main.py  # Launch interactive mode
```

3.3 Docker Installation (Optional)

```
# Build Docker image
docker build -t vulnhunter .

# Run container
docker run -it --rm -v $(pwd)/results:/app/vapt_results vulnhunter
```

4. Authentication System

4.1 Two-Factor Authentication (2FA)

VulnHunter implements enterprise-grade two-factor authentication to protect access to the security testing framework. The authentication system consists of two layers:

Layer 1: Username and Password Authentication - Secure credential verification using SHA-256 hashing with salt - Failed attempt tracking with automatic lockout protection - 5-minute lockout after 3 failed attempts

Layer 2: Email OTP Verification - 6-digit one-time password sent via email - Professional HTML email templates with security warnings - 5-minute expiration time for enhanced security - 3 OTP verification attempts before authentication failure

4.2 Authentication Process

Step 1: Launch VulnHunter

```
python3 main.py
```

Step 2: Enter Credentials - Username: WhiteDevil - Password: [Configured securely] - System verifies credentials and displays success message

Step 3: OTP Verification - OTP automatically sent to registered email (narainjkans@gmail.com) - Check email (including spam/junk folders) for 6-digit code - Enter OTP within 5 minutes to complete authentication

4.3 Email Configuration

SMTP Settings: - Server: smtp.gmail.com - Port: 587 (TLS encryption) - Authentication: App password for enhanced security

Email Features: - Professional HTML formatting with VulnHunter branding - Session information and timestamps - Security warnings and usage guidelines - Automatic cleanup of expired OTP codes

4.4 Testing Authentication

Test OTP System:

```
python3 test_otp_authentication.py
```

Test Options: 1. **OTP Email System Only** - Test email sending and verification 2. **Complete Authentication** - Test full 2FA flow 3. **Exit** - Exit the test suite

4.5 Security Features

Account Protection: - SHA-256 password hashing with salt - Automatic lockout after failed attempts - Session tracking and audit logging - Secure OTP generation and storage

Email Security: - TLS-encrypted SMTP communication - Professional security notifications - Time-limited verification codes - Automatic cleanup of sensitive data

5. User Interface Overview

5.1 Main Menu

When you launch VulnHunter, you'll see the main menu with these options:

```
╔═══════════════════════════════════════════════════════════════╗
║                          VulnHunter                          ║
║               Professional Security Testing                   ║
╚═══════════════════════════════════════════════════════════════╝

1. Network VAPT        - Infrastructure security assessment
2. Web Application VAPT - Web security testing
3. Cloud VAPT          - Cloud infrastructure assessment
4. API VAPT            - API security validation
5. Tool Dependencies   - Check tool availability
6. Auto-Install Tools  - Install missing security tools
7. Exit               - Exit application

Select an option (1-7):
```

5.2 Navigation Principles

- **Menu-Driven Interface**: All functions accessible through numbered menus
- **Progressive Disclosure**: Information revealed step-by-step through phases
- **Color-Coded Output**: Green (success), Red (errors), Yellow (warnings), Blue (information)
- **Session Management**: Automatic session saving and recovery
- **Confirmation Prompts**: Critical actions require user confirmation

5.3 Output Formatting

VulnHunter uses consistent formatting conventions:

- **[INFO]**: General information and status updates
- **[SUCCESS]**: Successful operations and completions
- **[WARNING]**: Cautions and non-critical issues
- **[ERROR]**: Critical errors requiring attention
- **[RESULT]**: Assessment findings and discoveries

6.## 6. Assessment Types

6.1 Network VAPT

Purpose: Comprehensive network infrastructure security assessment

Key Features: - Port scanning and service enumeration - Vulnerability identification and validation - Network topology mapping - Service banner grabbing and fingerprinting - SSL/TLS security assessment

Typical Use Cases: - Internal network security audits - External perimeter testing - Network segmentation validation - Compliance assessments (PCI DSS, SOX)

Example Workflow:

```
# Interactive mode
python3 main.py
# Select option 1: Network VAPT
# Enter target: 192.168.1.0/24

# Command line mode
python3 main.py --type network --target 192.168.1.0/24
```

5.2 Web Application VAPT

Purpose: OWASP Top 10 focused web application security testing

Key Features: - Automated vulnerability scanning (SQLi, XSS, CSRF) - Directory and file enumeration - Authentication bypass testing - Session management assessment - Input validation testing

Typical Use Cases: - Pre-production security testing - Periodic security assessments - Compliance validation (OWASP ASVS) - DevSecOps integration

Example Workflow:

```
# Interactive mode
python3 main.py
# Select option 2: Web Application VAPT
# Enter target: https://example.com

# Command line mode
python3 main.py --type web --target https://example.com
```

5.3 Cloud VAPT

Purpose: Multi-cloud infrastructure security evaluation

Key Features: - AWS, Azure, GCP configuration assessment - IAM policy analysis - Storage bucket security evaluation - Network security group assessment - Compliance reporting (CIS benchmarks)

Typical Use Cases: - Cloud migration security validation - Regular cloud posture assessment - Compliance auditing - DevOps security integration

Example Workflow:

```
# Interactive mode
python3 main.py
# Select option 3: Cloud VAPT
# Enter cloud provider: aws
# Enter configuration file: examples/cloud-config.json

# Command line mode
python3 main.py --type cloud --target aws --config examples/cloud-config.json
```

5.4 API VAPT

Purpose: REST/GraphQL API security testing and validation

Key Features: - Authentication and authorization testing - Input validation and injection testing - Rate limiting and DoS assessment - API endpoint discovery - Business logic testing

Typical Use Cases: - API security validation - Mobile application backend testing - Microservices security assessment - Third-party API integration testing

Example Workflow:

```
# Interactive mode
python3 main.py
# Select option 4: API VAPT
# Enter target: https://api.example.com

# Command line mode
python3 main.py --type api --target https://api.example.com
```

6.## 6. VAPT Methodology

VulnHunter follows a structured 5-phase methodology based on industry standards (PTES, OWASP Testing Guide, NIST SP 800-115).

6.1 Phase 1: Planning and Scope Definition

Objectives: - Define assessment scope and boundaries - Identify target systems and applications - Select appropriate testing methodology - Configure tool settings and parameters

User Interactions: - Target specification and validation - Assessment type selection - Scope confirmation and authorization - Tool configuration review

Deliverables: - Scope definition document - Testing methodology confirmation - Tool readiness verification - Session initialization

6.2 Phase 2: Reconnaissance and Information Gathering

Passive Reconnaissance: - OSINT collection and analysis - Domain and subdomain enumeration - Public record searches - Social media intelligence gathering

Active Reconnaissance: - Port scanning and service detection - Technology stack fingerprinting - Network topology mapping - Service banner enumeration

Key Activities:

```
# Example reconnaissance commands generated by VulnHunter:
nmap -sS -sV -O -A 192.168.1.0/24
dirb https://example.com /usr/share/dirb/wordlists/common.txt
dnsrecon -d example.com -t axfr
```

6.3 Phase 3: Vulnerability Assessment

Automated Scanning: - Comprehensive vulnerability detection - Service-specific security testing - Configuration assessment - Compliance validation

Manual Verification: - False positive elimination - Impact assessment - Exploitability validation - Business context analysis

Risk Classification: - CVSS v3.1 scoring - Business impact assessment - Exploitability rating - Remediation complexity evaluation

6.4 Phase 4: Penetration Testing and Exploitation

Exploit Selection: - CVE-to-exploit mapping - Success rate analysis - Target compatibility assessment - Impact prediction

Manual Testing: - Guided exploitation procedures - Custom payload development - Multi-stage attack chains - Privilege escalation attempts

Post-Exploitation: - System enumeration - Data collection - Lateral movement testing - Persistence establishment

Evidence Collection: - Screenshot capture - Command output logging - Configuration file extraction - Network traffic analysis

6.5 Phase 5: Reporting and Remediation

Report Generation: - Executive summary creation - Technical findings documentation - Risk assessment and prioritization - Remediation recommendations

Deliverables: - Executive summary (business focus) - Technical report (implementation details) - Evidence package (proof-of-concept) - Remediation roadmap (action items)

7.## 7. Tool Management

7.1 Tool Dependencies Overview

VulnHunter integrates 57 security tools across multiple categories:

Network Tools: - nmap: Network discovery and security auditing - masscan: High-speed port scanner - netdiscover: ARP reconnaissance tool - arp-scan: ARP-based network discovery

Web Application Tools: - nikto: Web server scanner - dirb: Directory/file brute forcer - sqlmap: SQL injection testing tool - gobuster: Directory/DNS busting tool

SSL/TLS Tools: - sslscan: SSL/TLS cipher suite scanner - testssl.sh: SSL/TLS security tester - openssl: Cryptographic toolkit

Cloud Tools: - aws-cli: Amazon Web Services CLI - scout-suite: Cloud security auditing - prowler: AWS security assessment

7.2 Automatic Tool Installation

VulnHunter includes an intelligent tool management system:

Features: - Automatic dependency detection - System package installation (apt-get) - Python package installation (pip3) - Installation verification and testing - Comprehensive reporting

Usage:

```
# From main menu
python3 main.py
# Select option 6: Auto-Install Missing Tools
# Choose installation mode:
#   1. Auto-install missing tools
#   2. Generate installation script
#   3. Continue without installing
```

Installation Process: 1. **Detection**: Scan system for missing tools 2. **Analysis**: Categorize tools by installation method 3. **Installation**: Execute installation commands 4. **Verification**: Test tool functionality 5. **Reporting**: Provide detailed installation results

7.3 Manual Tool Installation

For tools requiring manual installation:

Git-based Tools:

```
# Example: Installing additional security tools
git clone https://github.com/projectdiscovery/nuclei
cd nuclei && go build -o nuclei main.go
sudo mv nuclei /usr/local/bin/
```

Complex Tools:

```
# Example: Installing Burp Suite Community
wget -O burpsuite.jar https://portswigger.net/burp/releases/download
java -jar burpsuite.jar
```

8.## 8. Advanced Features

8.1 Metasploit Integration

VulnHunter includes complete Metasploit Framework integration with 3,758 modules:

Module Categories: - **Exploits**: 2,212 modules for vulnerability exploitation - **Auxiliary**: 867 modules for scanning and enumeration - **Post-Exploitation**: 368 modules for post-compromise activities - **Payloads**: Multiple payload options for different platforms

Usage Examples:

```
# CVE-based exploit search
# In penetration testing phase, enter: cve:CVE-2021-44228
# VulnHunter will find related exploits and provide guidance

# Manual exploit selection
# Choose from categorized exploit list
# Get step-by-step exploitation guidance
```

8.2 Exploit-DB Integration

Access to 46,000+ real-world exploits:

Features: - CVE-to-exploit mapping - Manual testing procedures - Target-specific commands - Success rate indicators

Usage:

```
# Recent exploit browsing
# In penetration testing phase, enter: edb
# Browse recent Exploit-DB entries
# Exploit download and customization
# Download exploit code for manual review
# Customize for specific target environment
```

8.3 Session Management

Persistent Sessions: - Automatic session saving - Progress tracking across phases - Session recovery after interruption - Multi-session management

Session Features:

```
# Session storage location
./vapt_sessions/<session_id>/
├── session_metadata.json
├── phase_1_planning.json
├── phase_2_reconnaissance.json
├── phase_3_vulnerability_assessment.json
├── phase_4_penetration_testing.json
└── phase_5_reporting.json
```

8.4 Configuration Management

Configuration Files: - JSON-based configuration - Assessment-specific settings - Tool parameter customization - Report format preferences

Example Configurations:

```
// Network assessment configuration
{
  "target": "192.168.1.0/24",
  "scan_type": "comprehensive",
  "ports": "1-65535",
  "timing": "T4",
  "scripts": ["vuln", "default", "discovery"],
  "output_formats": ["html", "pdf"],
  "exploit_integration": true
}
```

9.## 9. Report Generation

9.1 Report Types

Executive Summary: - Business impact analysis - Risk level distribution - Compliance status - Strategic recommendations

Technical Report: - Detailed vulnerability findings - Proof-of-concept evidence - Exploitation procedures - Technical remediation steps

Evidence Package: - Command outputs - Screenshots and visual evidence - Configuration files - Network diagrams

9.2 Output Formats

PDF Reports: - Professional formatting - Executive-ready presentation - Embedded evidence - Custom branding options

HTML Reports: - Interactive navigation - Searchable content - Embedded media - Responsive design

JSON Export: - Machine-readable format - API integration ready - Custom processing - Database import compatible

Plain Text: - Console-friendly format - Log file compatible - Script processing ready - Minimal formatting

9.3 Report Customization

Template Modification:

```
# Edit HTML template
templates/report_template.html

# Custom CSS styling
# Modify report appearance and branding
# Add organization logos and colors
```

Report Configuration:

```
{
  "report_settings": {
    "format": ["pdf", "html"],
    "executive_summary": true,
    "technical_details": true,
    "evidence_collection": true,
    "custom_branding": true,
    "classification": "Internal Use"
  }
}
```

10.## 10. Configuration Management

10.1 Global Configuration

Tool Configuration (config/tools.json):

```
{
  "network_tools": {
    "nmap": {
      "priority": 1,
      "install_hint": "apt-get install nmap",
      "test_command": "nmap --version",
      "features": ["port_scanning", "service_detection", "os_detection"]
    }
  }
}
```

10.2 Assessment-Specific Configuration

Network Assessment:

```
{
  "target": "192.168.1.0/24",
  "scan_type": "comprehensive",
  "ports": "1-65535",
  "timing": "T4",
  "scripts": ["vuln", "default", "discovery"],
  "exclude_hosts": ["192.168.1.1", "192.168.1.254"],
  "output_formats": ["html", "json"],
  "exploit_integration": true,
  "compliance_checks": ["pci_dss", "nist"]
}
```

Web Assessment:

```
{
  "target": "https://example.com",
  "scope": [
    "https://example.com/*",
    "https://api.example.com/*"
  ],
  "authentication": {
    "type": "cookie",
    "credentials": "session=abc123; token=xyz789"
  },
  "tests": ["owasp_top10", "custom_payloads"],
  "recursion_depth": 3,
  "rate_limiting": {
    "requests_per_second": 10,
    "concurrent_threads": 5
  },
  "report_format": "pdf"
}
```

10.3 User Preferences

Personal Configuration:

```
{
  "user_preferences": {
    "default_output_dir": "./vapt_results",
    "preferred_report_format": "pdf",
    "verbosity_level": "normal",
    "color_output": true,
    "auto_save_sessions": true,
    "confirmation_prompts": true
  }
}
```

11.## 11. Best Practices

11.1 Pre-Assessment Planning

Authorization and Scope: - Obtain written authorization before testing - Clearly define scope boundaries - Identify critical systems requiring special handling - Establish communication protocols - Plan for emergency procedures

Technical Preparation: - Verify tool functionality and updates - Prepare isolated testing environment - Configure backup and recovery procedures - Test network connectivity and access - Review target system documentation

11.2 During Assessment

Methodology Adherence: - Follow structured 5-phase approach - Document all activities and findings - Maintain detailed evidence collection - Regularly backup session data - Monitor system impact and performance

Safety and Ethics: - Avoid destructive testing methods - Respect system availability requirements - Maintain confidentiality of findings - Follow responsible disclosure practices - Document all testing activities

11.3 Post-Assessment

Report Quality: - Provide clear executive summaries - Include detailed technical findings - Offer prioritized remediation recommendations - Attach supporting evidence - Ensure accuracy and completeness

Knowledge Management: - Archive assessment data securely - Update tool configurations based on lessons learned - Share knowledge with team members - Maintain assessment methodology documentation - Plan follow-up assessments

11.4 Compliance Considerations

Regulatory Requirements: - Understand applicable compliance frameworks - Map findings to compliance requirements - Provide compliance-specific reporting - Maintain audit trails and documentation - Support compliance validation activities

Industry Standards: - Follow OWASP Testing Guide recommendations - Adhere to PTES methodology standards - Apply NIST Cybersecurity Framework principles - Consider ISO 27001 requirements - Implement CIS Controls where applicable

---

12.## 12. Troubleshooting

12.1 Common Issues

Installation Problems:

Issue: Python dependency installation fails

```
# Solution: Update pip and install system dependencies
sudo apt update
sudo apt install python3-dev build-essential
pip3 install --upgrade pip
pip3 install colorama jinja2 tabulate requests
```

Issue: Tool installation permissions denied

```
# Solution: Use sudo for system-wide installation
sudo apt install nmap masscan nikto
# Or install to user directory
pip3 install --user package_name
```

Runtime Errors:

Issue: "Tool not found" errors during assessment

```
# Solution: Check tool availability and install
python3 main.py
```

```
# Select option 5: Tool Dependencies Check
# Select option 6: Auto-Install Missing Tools
```

Issue: Session data corruption

```
# Solution: Clear session cache and restart
rm -rf ./vapt_sessions/<session_id>
python3 main.py
# Start new assessment session
```

12.2 Performance Issues

Slow Network Scanning: - Adjust timing templates (T1-T5) - Reduce port range scope - Increase thread count (if system allows) - Use target-specific configurations

Memory Usage: - Monitor system resources during large assessments - Split large target ranges into smaller chunks - Increase virtual memory if needed - Close unnecessary applications

12.3 Network Connectivity

Proxy Configuration:

```
# Set proxy environment variables
export http_proxy=http://proxy.company.com:8080
export https_proxy=https://proxy.company.com:8080
python3 main.py
```

Firewall Issues:

```
# Check firewall rules
sudo ufw status
# Allow outbound connections for security tools
sudo ufw allow out 80,443,53
```

12.4 Getting Help

Log Files: - Check VulnHunter logs in `./logs/` directory - Review tool-specific output files - Monitor system logs for errors

Community Support: - GitHub Issues: Report bugs and feature requests - Documentation Wiki: Detailed guides and tutorials - Community Forums: User discussions and tips

Professional Support: - Enterprise support contracts available - Custom development and integration services - Training and certification programs

---

13.## 13. Appendices

Appendix A: Tool Reference

Network Security Tools: | Tool | Purpose | Installation | Usage | |------|---------|--------------|--------| | nmap | Port scanning, service detection | `apt install nmap` | `nmap -sS -sV target` | | masscan | High-speed port scanning | `apt install` `masscan` | `masscan -p1-65535 target` | | netdiscover | ARP reconnaissance | `apt install netdiscover` | `netdiscover -r 192.168.1.0/24` | | arp-scan | ARP-based discovery | `apt install arp-scan` | `arp-scan -l` |

Web Security Tools: | Tool | Purpose | Installation | Usage | |------|---------|--------------|--------| | nikto | Web server scanning | `apt install nikto` | `nikto -h target` | | dirb | Directory brute forcing | `apt install dirb` | `dirb target wordlist` | | sqlmap | SQL injection testing | `apt install sqlmap` | `sqlmap -u target` | | gobuster | Directory/DNS busting | `apt install gobuster` | `gobuster dir -u target -w wordlist` |

Appendix B: Configuration Examples

Complete Network Assessment Configuration:

```
{
  "assessment_type": "network",
  "target": "192.168.1.0/24",
  "scan_configuration": {
    "ports": "1-65535",
    "timing": "T4",
    "scripts": ["vuln", "default", "discovery", "safe"],
    "os_detection": true,
    "service_detection": true,
    "version_detection": true
  },
  "exclusions": {
    "hosts": ["192.168.1.1", "192.168.1.254"],
    "ports": ["25", "143", "993"]
  },
  "advanced_options": {
    "fragment_packets": false,
    "source_port": null,
    "spoof_source": null,
    "decoy_scan": false
  },
  "reporting": {
    "formats": ["pdf", "html", "json"],
    "executive_summary": true,
    "technical_details": true,
    "evidence_collection": true,
    "compliance_mapping": ["nist", "pci_dss"]
  },
  "integration": {
    "metasploit": true,
    "exploitdb": true,
    "cve_mapping": true,
    "risk_assessment": true
  }
}
```

Appendix C: Legal and Compliance

Legal Disclaimer: VulnHunter is designed for authorized security testing only. Users must: - Obtain written permission before testing any systems - Respect scope limitations and boundaries - Follow applicable laws and regulations - Maintain confidentiality of findings - Report vulnerabilities responsibly

Compliance Frameworks: - **OWASP ASVS**: Application Security Verification Standard - **NIST SP 800-115**: Technical Guide to Information Security Testing - **PTES**: Penetration Testing Execution Standard - **ISO 27001**: Information Security Management - **PCI DSS**: Payment Card Industry Data Security Standard

Appendix D: Troubleshooting Reference

Common Error Messages:

"Permission denied" errors: - Run with appropriate privileges (sudo if needed) - Check file permissions on target directories - Verify network access permissions

"Tool not found" errors: - Install missing tools using auto-install feature - Check PATH environment variable - Verify tool installation location

"Target unreachable" errors: - Verify network connectivity - Check firewall rules - Confirm target system availability

Performance Tuning:

For large networks: - Use parallel scanning options - Implement target segmentation - Optimize timing parameters - Monitor system resources

For web applications: - Configure rate limiting - Implement session management - Use appropriate thread counts - Enable progress monitoring

---

Document Information: - **Version**: 2.0 - **Last Updated**: July 26, 2025 - **Authors**: VulnHunter Development Team - **Classification**: Internal Use - **Next Review Date**: October 26, 2025
