Building a Penetration Testing Linux Distribution from Scratch

bad-antics | June 2024 | Security

Abstract

This paper describes the design and implementation of NullSec, a purpose-built penetration testing Linux distribution. NullSec integrates network analysis, web application testing, exploit development, credential tools, post-exploitation frameworks, and reporting into a cohesive platform optimized for offensive security operations.

1 Introduction

This paper describes the design and implementation of NullSec, a penetration testing Linux distribution built from scratch. NullSec integrates security auditing tools, network analysis capabilities, and exploit development frameworks into a cohesive, purpose-built operating system.

Unlike general-purpose distributions adapted for security testing, NullSec is designed from the ground up with penetration testing workflows as the primary use case. Every system component is selected and configured to support offensive security operations.

The distribution targets both physical hardware and virtual machine environments. A live USB mode provides immediate access to all tools without installation. An installed mode provides persistent storage for ongoing engagements.

2 Base System Design

NullSec uses a minimal Linux kernel with custom configuration optimized for network operations, USB device support, and wireless driver compatibility. The kernel includes modules for packet injection, USB gadget mode, and hardware crypto acceleration.

The init system uses a lightweight process supervisor that starts services in dependency order. Unlike systemd, the supervisor has no external dependencies and boots in under 3 seconds on modern hardware.

The package manager uses a binary package format with cryptographic signatures. Packages are built from source with security-relevant compiler flags: stack canaries, ASLR, position-independent executables, and RELRO.

The base system occupies 2.1 GB on disk. The full installation with all tool categories requires 8.5 GB. The live USB image is 4.2 GB and includes the most commonly used tools.

3 Network Analysis Tools

Network reconnaissance tools include: nmap (port scanner), masscan (high-speed port scanner),

zmap (internet-wide scanner), and netdiscover (ARP-based host discovery). Tools are pre-configured with common scan profiles.

Packet capture and analysis uses tcpdump for command-line capture and Wireshark for graphical analysis. Custom capture filters for common protocols (HTTP, DNS, SMB, Kerberos) are pre-installed.

Wireless network analysis includes aircrack-ng suite (monitoring, injection, cracking), Kismet (wireless detection and sniffing), and wifite (automated wireless auditing). Drivers for common chipsets are pre-installed.

DNS analysis tools include: dig, dnsrecon (DNS enumeration), dnsenum (brute-force subdomain discovery), and fierce (DNS reconnaissance). Custom wordlists for subdomain brute-forcing are included.

4 Web Application Testing

Web vulnerability scanners include: Burp Suite (proxy-based testing), OWASP ZAP (open-source web scanner), nikto (web server scanner), and sqlmap (SQL injection automation).

Web fuzzing tools include: ffuf (web fuzzer), gobuster (directory brute-force), and wfuzz (parameter fuzzing). Wordlists from SecLists and custom NullSec wordlists are pre-installed.

API testing tools include: Postman (API client), httpie (command-line HTTP client), and custom scripts for JWT token manipulation, OAuth flow testing, and GraphQL introspection.

Browser automation for testing uses Selenium and Playwright with pre-configured profiles. Headless browser mode enables automated testing of JavaScript-heavy applications.

5 Exploit Development

Binary analysis tools include: Ghidra (reverse engineering), radare2 (binary analysis framework), and Binary Ninja (decompiler). Debugging tools include GDB with GEF (GDB Enhanced Features) and pwndbg.

Exploit development frameworks include: Metasploit (exploitation framework), pwntools (CTF/exploit library for Python), and ROPgadget (return-oriented programming tool).

Shellcode development tools include: nasm (assembler), msfvenom (payload generator), and custom shellcode templates for Linux x86_64, ARM, and RISC-V architectures.

Fuzzing infrastructure includes: AFL++ (coverage-guided fuzzer), libFuzzer integration, and honggfuzz. Pre-built corpus collections for common file formats accelerate fuzzing campaigns.

6 Password and Credential Tools

Password cracking tools include: hashcat (GPU-accelerated cracking), John the Ripper (CPU-based cracking), and Ophcrack (rainbow table-based). Wordlists include RockYou, SecLists, and custom NullSec dictionaries.

Credential dumping tools extract credentials from running systems: Mimikatz (Windows), LaZagne (multi-platform), and custom scripts for extracting browser-stored passwords and SSH keys.

Active Directory tools include: Impacket (network protocol library), BloodHound (AD attack path analysis), and CrackMapExec (post-exploitation tool). Pre-configured for common AD assessment workflows.

7 Post-Exploitation

Lateral movement tools enable moving between compromised systems: SSH tunneling, proxychains (traffic routing through compromised hosts), and chisel (HTTP tunnel). Configuration templates simplify setup.

Persistence mechanisms include: scheduled task creation, service installation, and registry modification scripts. Each mechanism has detection signatures for defensive testing.

Data exfiltration tools include: encrypted file transfer utilities, steganography tools for covert data hiding, and DNS tunneling for data exfiltration through restrictive firewalls.

8 Reporting and Documentation

Engagement documentation tools include: CherryTree (hierarchical note-taking), Obsidian (markdown-based notes), and custom templates for penetration test reports.

Screenshot and evidence capture is integrated into the desktop environment. Global hotkeys capture screenshots with metadata (timestamp, tool, target). Evidence is organized by engagement and target.

Report generation produces professional penetration test reports from collected evidence. Templates follow OWASP, PTES, and OSSTMM reporting standards. Reports are generated in PDF and HTML formats.

9 Conclusion

NullSec demonstrates that a purpose-built penetration testing distribution provides significant advantages over adapting general-purpose distributions. Integrated tools, pre-configured workflows, and custom kernel support create an efficient security testing platform.

2.1 Kernel Configuration

Network stack configuration enables all packet socket types, raw socket access, and traffic control

features. Netfilter modules for packet manipulation, NAT, and connection tracking are compiled as loadable modules.

Wireless driver configuration includes: ath9k, ath10k, rtl8188eu, rtl8812au, and mt76. These chipsets are selected for monitor mode and packet injection support. Firmware blobs are included in the initramfs.

USB configuration enables USB gadget mode for emulating USB devices (keyboard, network adapter, storage). HID gadget support enables keystroke injection attacks. Mass storage gadget enables autorun attacks.

Filesystem support includes ext4 (primary), NTFS (Windows evidence), FAT32 (USB drives), LUKS (encrypted volumes), and SquashFS (live image). Filesystem forensic extensions enable deleted file recovery.

Namespace and cgroup configuration provides container isolation for running tools in isolated environments. Network namespaces enable parallel testing without interference. PID namespaces isolate processes.

Performance tuning: TCP buffer sizes are increased for high-speed scanning (wmem_max = 16 MB). The network hash table size is increased for tracking many connections. SYN cookies are enabled.

Security configuration: ASLR is enabled for the kernel. Stack canaries protect kernel functions. KASLR randomizes the kernel's load address. Kernel lockdown mode is available but disabled by default.

Virtualization support includes KVM modules for running virtual machines during testing. VirtIO drivers enable efficient I/O in VM environments. Nested virtualization is supported on compatible CPUs.

Custom kernel patches add: extended Berkeley Packet Filter (eBPF) enhancements for network monitoring, SCTP support for telecom protocol testing, and Bluetooth HCI injection support.

Kernel module signing is configured with the NullSec signing key. Only signed modules can be loaded in secure boot mode. Developer mode allows unsigned module loading for custom tool development.

3.1 Network Reconnaissance Methodology

Host discovery workflows progress from passive (ARP monitoring, DNS queries) to active (ICMP ping, TCP SYN scan). Passive discovery avoids detection. Active discovery provides comprehensive coverage.

Port scanning methodology uses three phases: fast scan (top 1000 ports), comprehensive scan (all 65535 ports), and service version detection (targeted probes on open ports).

Service enumeration identifies running services and versions. NSE (Nmap Scripting Engine) scripts probe for known vulnerabilities, default credentials, and information disclosure.

OS fingerprinting uses TCP/IP stack behavior differences to identify the target operating system and version. Nmap's OS detection analyzes TCP window size, TTL, and option ordering.

Network mapping builds a topology diagram from scan results. The mapper identifies routers, firewalls, and network boundaries. Traceroute data supplements scan results for path analysis.

Vulnerability correlation matches service versions against the CVE database. The correlation tool prioritizes vulnerabilities by CVSS score and exploit availability.

Scan output management stores scan results in structured formats: XML (nmap), JSON (masscan), and SQLite (consolidated). A unified search interface queries across all scan results.

Stealth scanning techniques minimize detection: SYN scan (half-open), FIN scan, idle scan (zombie host), and fragmented packets. Timing options control scan speed to avoid IDS triggers.

IPv6 scanning addresses the challenges of scanning large IPv6 address spaces. Techniques include: multicast discovery, DNS enumeration, and inference from IPv4 results.

Custom NSE script development for NullSec-specific checks: default credential testing, custom vulnerability checks, and organization-specific compliance verification.

4.1 Web Vulnerability Categories

SQL injection testing: sqlmap automates detection and exploitation of SQL injection flaws. Custom tamper scripts bypass WAF rules. Supported databases include MySQL, PostgreSQL, MSSQL, Oracle, and SQLite.

Cross-site scripting (XSS) testing uses custom payloads for reflected, stored, and DOM-based XSS. The XSS toolkit includes context-aware payload generation that adapts to the injection point.

Server-Side Request Forgery (SSRF) testing probes for internal service access through web application requests. The SSRF toolkit includes payloads for cloud metadata endpoints (AWS, GCP, Azure).

Insecure deserialization testing generates crafted serialized objects for Java (ysoserial), PHP, Python (pickle), and .NET. Each generator produces payloads for known deserialization gadget chains.

Authentication bypass testing includes: credential stuffing, brute force with custom wordlists, session fixation, JWT manipulation, and OAuth flow abuse.

Authorization testing verifies that access controls are enforced: horizontal privilege escalation (accessing other users' data), vertical privilege escalation (accessing admin functions), and IDOR (insecure direct object reference).

File upload vulnerability testing submits crafted files: web shells, polyglot files (valid image and PHP), and oversized files. The tester verifies that content type validation and file extension checks are enforced.

API security testing covers: authentication (API keys, OAuth tokens, JWT), authorization (role-based

access), input validation (injection, parameter tampering), and rate limiting.

Content Security Policy (CSP) analysis evaluates the effectiveness of CSP headers. The analyzer identifies: missing directives, overly permissive sources, and bypass techniques.

Subdomain takeover detection identifies DNS records pointing to unclaimed cloud resources. The scanner checks for: dangling CNAME records, expired services, and orphaned DNS entries.

5.1 Reverse Engineering Workflow

Static analysis begins with file format identification (file, binwalk), string extraction, and import/export table analysis. Ghidra's auto-analysis identifies functions, cross-references, and data types.

Dynamic analysis runs the target in a controlled environment with strace (system call tracing), ltrace (library call tracing), and GDB (instruction-level debugging).

Disassembly and decompilation: Ghidra produces C-like pseudocode from machine code. The analyst refines variable names, types, and function signatures to improve readability.

Control flow analysis identifies: loops, conditionals, switch statements, and function calls. The control flow graph visualizes program structure and highlights suspicious patterns.

Data flow analysis tracks how data moves through the program: from input (read, recv) through processing to output (write, send). Data flow reveals buffer operations and potential vulnerabilities.

Patch diffing compares two versions of a binary to identify security patches. The differ highlights changed functions and instructions. Patch analysis reveals the vulnerability being fixed.

Anti-debugging detection identifies techniques used to prevent analysis: debugger detection (ptrace, IsDebuggerPresent), timing checks, code obfuscation, and integrity verification.

Firmware extraction and analysis extracts file systems from firmware images using binwalk. The extracted file system is analyzed for: hard-coded credentials, vulnerable services, and debug interfaces.

Protocol reverse engineering uses network captures and binary analysis to document undocumented protocols. The analyst reconstructs message formats, state machines, and authentication mechanisms.

Automated vulnerability discovery uses pattern matching to find common vulnerability patterns: unchecked buffer copies, format string usage, integer overflow, and use-after-free patterns.

6.1 Credential Attack Strategies

Online credential attacks target live authentication endpoints: SSH, RDP, HTTP login forms, and database servers. Hydra and Medusa perform parallel login attempts with configurable timing.

Offline credential attacks process captured password hashes: NTLM, NTLMv2, Kerberos TGT, bcrypt, scrypt, and PBKDF2. Hashcat GPU acceleration achieves billions of hashes per second for

fast algorithms.

Password spray attacks try a small number of common passwords against many accounts. Spraying avoids account lockout thresholds. The spray interval is configured based on the target's lockout policy.

Rule-based password generation creates password candidates by applying transformations to a base wordlist: capitalization, leetspeak, appending numbers/symbols, and combining words.

Kerberoasting extracts Kerberos service ticket hashes for offline cracking. The attack targets service accounts with SPNs (Service Principal Names) and weak passwords.

AS-REP roasting targets Active Directory accounts with pre-authentication disabled. The attacker requests authentication data and cracks the response offline.

Pass-the-hash attacks use captured NTLM hashes to authenticate without knowing the plaintext password. The attack works because NTLM authentication uses the hash directly.

Token impersonation captures and reuses authentication tokens from compromised processes. The attack escalates privileges by impersonating tokens with higher privilege levels.

Certificate-based authentication attacks target PKI infrastructure: extracting private keys, forging certificates, and exploiting certificate validation vulnerabilities.

Multi-factor authentication bypass techniques include: SIM swapping, push notification fatigue, and time-based OTP prediction from weak seed values.

7.1 Command and Control Infrastructure

C2 framework integration includes: Cobalt Strike (commercial), Havoc (open-source), and Sliver (open-source). Each framework is pre-configured with communication profiles and evasion techniques.

C2 communication channels include: HTTPS (standard web traffic), DNS (covert channel through DNS queries), and custom protocols over common ports. Malleable C2 profiles mimic legitimate traffic.

Redirector setup automates the deployment of traffic redirectors that hide the true C2 server address. Redirectors forward legitimate-looking traffic and block direct access.

Payload generation creates platform-specific implants: Windows (exe, dll, PowerShell), Linux (ELF, shell script), and macOS (Mach-O, dylib). Payloads are customized to avoid signature detection.

Evasion techniques include: process hollowing (replacing legitimate process memory), unhooking (removing EDR hooks), AMSI bypass (disabling script scanning), and ETW patching (disabling event tracing).

Persistence across reboots uses: scheduled tasks, service creation, registry modification, and startup folder placement. Each mechanism is tested against common EDR products.

Lateral movement automation scripts execute common lateral movement techniques: PsExec, WMI execution, DCOM, and WinRM. Each technique has different detection signatures and prerequisites.

Privilege escalation automation: LinPEAS (Linux enumeration), WinPEAS (Windows enumeration), and custom scripts that check for common misconfigurations.

Data staging collects and compresses target data before exfiltration. Staging reduces the number of network transfers and makes the operation more efficient.

Operational security (OPSEC) guidelines: use encrypted communications, rotate infrastructure regularly, minimize tool footprint, and clean up artifacts after each operation.

2.2 Desktop Environment

The NullSec desktop uses a lightweight tiling window manager (i3) optimized for penetration testing workflows. Multiple workspaces are pre-configured: reconnaissance, exploitation, post-exploitation, and reporting.

Terminal emulator configuration provides tabbed terminals with custom profiles for different tools. Color schemes distinguish tool categories: red for exploitation, blue for reconnaissance, green for analysis.

Keyboard shortcuts provide rapid access to common tools: Ctrl+Alt+N for nmap, Ctrl+Alt+B for Burp Suite, Ctrl+Alt+M for Metasploit. All shortcuts are configurable through a central configuration file.

System tray indicators show: VPN connection status, active network interfaces, running scans, and resource utilization. Critical alerts (scan complete, exploit successful) are displayed as desktop notifications.

Multi-monitor support arranges workspaces across multiple displays. A common setup places reconnaissance tools on one monitor and exploitation tools on another.

Virtual desktop integration provides isolated desktops for different engagement phases. Each desktop has its own set of terminals and tools. Desktop switching is instant with keyboard shortcuts.

File manager integration provides quick access to common directories: wordlists, exploits, engagement data, and tool output. Context menu actions include 'Open in terminal' and 'Hash file'.

Clipboard management stores clipboard history for the last 50 entries. Sensitive data (passwords, hashes) is automatically cleared from the clipboard after 30 seconds.

Screenshot integration captures the current window or full screen with a single keystroke. Screenshots are saved with metadata (tool, target, timestamp) for report generation.

Theme and appearance use a dark color scheme to reduce eye strain during long engagements. Syntax highlighting in terminals uses colors optimized for dark backgrounds.

8.1 Evidence Management

Evidence chain of custody tracks all collected evidence from acquisition to reporting. Each evidence item has a unique identifier, hash, acquisition timestamp, and handler identity.

Evidence storage uses encrypted containers for sensitive data. Each engagement has its own encrypted volume. Encryption keys are derived from the engagement passphrase.

Evidence categorization classifies items by type: network captures, screenshots, logs, credentials, and exploits. Categories map to report sections for automated report generation.

Evidence integrity verification computes SHA-256 hashes at acquisition time and re-verifies before inclusion in reports. Hash mismatches indicate evidence tampering.

Timeline construction orders evidence items chronologically to reconstruct the attack narrative. The timeline shows: discovery time, exploitation time, and access duration for each target.

Finding severity classification uses CVSS v3.1 scoring. Each finding includes: base score, temporal score, environmental score, and a descriptive severity rating (Critical, High, Medium, Low, Informational).

Remediation recommendations accompany each finding. Recommendations are specific, actionable, and prioritized by risk. Each recommendation includes implementation difficulty and estimated effort.

Executive summary generation produces a high-level overview for non-technical stakeholders. The summary covers: engagement scope, key findings, overall risk assessment, and top recommendations.

Technical appendix generation includes detailed evidence for each finding: reproduction steps, tool output, screenshots, and affected system details.

Report review workflow requires peer review of findings before report delivery. The review checks: accuracy of findings, completeness of evidence, and clarity of recommendations.

3.2 Wireless Attack Techniques

WPA/WPA2 handshake capture uses aircrack-ng to place the wireless adapter in monitor mode, deauthenticate a connected client, and capture the four-way handshake when the client reconnects.

WPA2 PMKID attack captures the PMKID from the access point's first message without waiting for a full handshake. The PMKID is cracked offline with hashcat.

Evil twin attack creates a rogue access point with the same SSID as the target. Connected clients are redirected to a captive portal that captures credentials.

WPS PIN brute force attacks the Wi-Fi Protected Setup PIN. The attack exploits the WPS design flaw that allows the PIN to be cracked in 11,000 attempts instead of 100 million.

Bluetooth reconnaissance uses hcitool and bettercap to discover Bluetooth devices, enumerate services, and identify vulnerabilities. BLE (Bluetooth Low Energy) scanning targets IoT devices.

RFID/NFC analysis tools include: Proxmark3 for RFID cloning and sniffing, libnfc for NFC communication, and custom scripts for badge cloning and access control bypass.

Software-defined radio (SDR) integration includes: GNU Radio for signal processing, gqrx for spectrum analysis, and custom demodulators for common wireless protocols.

Wireless signal analysis measures signal strength, identifies access points and clients, and maps wireless coverage areas. Airodump-ng provides real-time monitoring of all wireless activity.

Rogue DHCP server deployment assigns malicious DNS servers to clients joining the network. DNS responses redirect traffic through the attacker's proxy for credential capture.

Karma attack responds to all wireless probe requests, tricking devices into connecting to the attacker's access point. The attack captures credentials and traffic from connected devices.

5.2 Exploit Payload Development

Return-oriented programming (ROP) chain construction identifies gadgets (short instruction sequences ending in ret) and chains them to execute arbitrary operations without injecting code.

Heap exploitation techniques include: use-after-free exploitation, heap overflow with controlled adjacent allocation, and tcache poisoning for modern glibc.

Format string exploitation uses format specifier vulnerabilities to read and write arbitrary memory. The exploit constructs write-what-where primitives using %n specifiers.

Kernel exploitation development uses dedicated kernel debugging VMs. Techniques include: stack buffer overflow, integer overflow, race condition (double fetch), and use-after-free in kernel space.

Shellcode encoding transforms raw shellcode to avoid bad characters (null bytes, newlines) and bypass basic filters. Encoders include: XOR, Alpha-numeric, and polymorphic.

Payload staging divides the exploit payload into stages. The first stage is small (fits in the buffer overflow) and downloads the larger second stage from the attacker's server.

Anti-virus evasion techniques for payloads include: custom encryption, code obfuscation, process injection, and living-off-the-land binaries (LOLBins).

Exploit reliability testing runs the exploit multiple times against the target to measure success rate. Factors affecting reliability: ASLR, stack canaries, and timing dependencies.

Cross-platform payload compilation generates payloads for multiple architectures from a single source. The compilation framework supports x86_64, ARM, MIPS, and RISC-V.

Exploit documentation standards require: affected software and version, root cause analysis, exploitation steps, reliability assessment, and detection signatures.

7.2 Pivoting and Tunneling

SSH tunneling creates encrypted tunnels through compromised hosts. Local port forwarding (-L) accesses remote services. Remote port forwarding (-R) exposes attacker services to the target network.

Dynamic SOCKS proxy through SSH (-D) creates a SOCKS5 proxy on the attacker's machine. Traffic routed through the proxy reaches the target network through the compromised host.

Chisel provides a fast TCP tunnel over HTTP. Chisel client runs on the compromised host and connects to the attacker's chisel server. The tunnel supports SOCKS proxy and port forwarding.

Proxychains routes tool traffic through a chain of proxies. The chain can include SOCKS4, SOCKS5, and HTTP proxies. Each hop adds a layer of indirection.

Double pivoting accesses networks separated by two hops from the attacker. The first pivot reaches the intermediate network. The second pivot, through a host on the intermediate network, reaches the target network.

Metasploit autoroute adds routes through compromised sessions. Traffic to the target network is automatically routed through the session. Multiple routes support complex network topologies.

DNS tunneling encapsulates TCP traffic in DNS queries and responses. The tunnel operates even through restrictive firewalls that only allow DNS. Throughput is limited by DNS packet size.

ICMP tunneling encapsulates data in ICMP echo request and reply packets. ICMP tunneling bypasses firewalls that block all TCP and UDP but allow ping.

Network segmentation testing verifies that network segments are properly isolated. The tester attempts to reach each segment from every other segment. Successful cross-segment access indicates segmentation failures.

Tunnel detection testing verifies that the target's monitoring can detect tunneling activity. The tester runs tunnels and checks whether security alerts are generated.

6.2 Hash Identification and Cracking

Hash type identification analyzes the hash format (length, character set, prefix) to determine the algorithm. Common formats: MD5 (32 hex), SHA-1 (40 hex), SHA-256 (64 hex), bcrypt ($2b$), NTLM (32 hex).

GPU-accelerated cracking uses hashcat with NVIDIA or AMD GPUs. A single RTX 4090 achieves: 164 billion MD5/s, 68 billion NTLM/s, 22 billion SHA-1/s, and 164 thousand bcrypt/s.

Rainbow table attacks use pre-computed hash-to-password mappings. Tables for common algorithms and character sets are included. Table lookup is near-instantaneous but requires significant storage.

Combinator attacks combine words from multiple wordlists. Each word from the first list is combined with each word from the second list. The combinator is effective for compound passwords.

Mask attacks generate candidates matching a pattern: ?u?l?l?l?d?d?d?d (uppercase, three lowercase, four digits). Masks target password policies that require specific character types.

Hybrid attacks combine wordlists with masks: append digits to each word (wordlist + ?d?d?d?d) or prepend special characters (?s + wordlist). Hybrids target common password modification patterns.

Distributed cracking across multiple machines uses hashcat's --session and --restore features. Each machine processes a different portion of the keyspace. Results are collected centrally.

Password analysis after cracking examines patterns: most common base words, common modifications, password length distribution, and character set usage. Analysis improves future wordlists.

Custom rule creation for hashcat uses the rule engine to define transformations: capitalize first letter, reverse string, duplicate, rotate characters, and toggle case.

Time estimation for cracking operations: MD5 with 8-character lowercase takes 3 minutes on a single GPU. SHA-256 with 8-character mixed case takes 6 hours. bcrypt with 8-character any charset takes 34 years.

8.2 Report Templates

Executive report template includes: engagement overview, scope, methodology summary, key findings (top 5), risk rating distribution, and strategic recommendations. Length: 3-5 pages.

Technical report template includes: methodology details, all findings with evidence, detailed remediation steps, tool output logs, and technical appendices. Length: varies by finding count.

Vulnerability finding template includes: title, CVSS score, description, affected systems, reproduction steps, evidence (screenshots, logs), impact analysis, and remediation recommendation.

Compliance mapping template maps findings to compliance frameworks: PCI DSS requirements, HIPAA security rules, SOC 2 criteria, and CIS benchmark controls.

Retest report template documents the remediation verification. Each finding is retested and marked: remediated, partially remediated, or not remediated. Evidence of the current state is included.

Debriefing presentation template provides slides for the post-engagement debriefing. Slides cover: engagement summary, attack path visualization, critical findings, and Q&A.

Finding database stores all findings from all engagements. The database enables: trend analysis, recurring vulnerability identification, and industry benchmarking.

Automated report generation uses Jinja2 templates populated from the finding database. Reports are rendered in PDF (via LaTeX) and HTML. Formatting is consistent across all reports.

Quality assurance checklist verifies report completeness: all findings have evidence, CVSS scores are calculated correctly, remediation recommendations are actionable, and the executive summary is accurate.

Client portal provides secure report delivery. Reports are encrypted with the client's public key. The portal provides download tracking and read receipts.

2.3 Build System

The NullSec build system uses Docker containers for reproducible builds. Each package is built in an isolated container with only the declared dependencies. Build artifacts are cached for incremental builds.

Cross-compilation support builds packages for x86_64, ARM64, and RISC-V from a single build host. Cross-compilation enables running NullSec on Raspberry Pi and RISC-V single-board computers.

ISO image generation combines the base system, tool packages, and live system infrastructure into a bootable ISO image. The live image uses SquashFS compression for efficient storage.

Continuous integration builds the ISO image on every commit to the main branch. Automated tests verify: boot success, tool availability, network connectivity, and regression tests.

Package repository hosting provides a signed APT repository for package updates. The repository supports multiple release channels: stable, testing, and development.

Version management tracks tool versions and updates. Security-critical tools (nmap, metasploit, burp) are updated within 48 hours of upstream releases. Other tools follow a weekly update cycle.

Custom tool packaging wraps NullSec-specific tools in Debian packages. Each package includes: binary, configuration files, documentation, and integration scripts.

Build reproducibility ensures that the same source code produces identical packages. Reproducible builds prevent supply chain attacks by enabling independent verification.

License compliance tracking identifies the license of each included tool. The distribution complies with GPL, BSD, MIT, and commercial license terms. License information is included in the ISO.

Size optimization reduces the ISO image size through: unused dependency removal, documentation compression, shared library deduplication, and tool selection based on usage statistics.

3.3 Protocol-Specific Testing

SMB/CIFS testing uses smbclient and enum4linux to enumerate shares, users, and groups on Windows systems. CrackMapExec automates SMB credential testing across the network.

SNMP testing uses snmpwalk and onesixtyone to enumerate SNMP-enabled devices. Default community strings (public, private) are tested. SNMP v1/v2c strings are transmitted in cleartext.

LDAP enumeration queries Active Directory for: users, groups, computers, GPOs, and trust relationships. ldapsearch provides raw LDAP queries. Custom scripts extract security-relevant attributes.

SSH testing verifies: supported algorithms, key exchange methods, and authentication mechanisms.

The tester checks for weak algorithms (DES, RC4), small key sizes, and default credentials.

RDP testing verifies: NLA (Network Level Authentication) enforcement, encryption level, and vulnerability to BlueKeep and similar RDP exploits. Rdesktop and xfreerdp provide client functionality.

SMTP testing includes: open relay detection, user enumeration (VRFY, RCPT TO), and SPF/DKIM/DMARC verification. Swaks provides SMTP transaction testing.

FTP testing checks for: anonymous access, writable directories, version-specific vulnerabilities, and cleartext credential transmission.

Database testing for MySQL, PostgreSQL, MSSQL, and Oracle checks: default credentials, excessive privileges, SQL injection through stored procedures, and data exposure.

VoIP testing includes: SIP enumeration (svmap, svwar), RTP interception, and toll fraud detection. VoIP-specific tools target PBX misconfiguration and credential weakness.

Industrial control system (ICS) testing tools handle Modbus, DNP3, and OPC UA protocols. Specialized tools query PLCs, RTUs, and SCADA systems. Testing requires explicit authorization.

4.2 Web Application Proxy Configuration

Burp Suite proxy configuration intercepts all browser traffic. The proxy captures requests and responses for manual inspection. Match-and-replace rules modify traffic in real time.

Certificate handling installs the proxy's CA certificate in the browser trust store. The proxy generates per-site certificates signed by the CA. This enables HTTPS interception.

Scope definition limits proxy activity to target domains. Out-of-scope traffic is passed through without interception. Scope prevents accidental testing of non-target systems.

Request repeater enables manual modification and replay of captured requests. The tester modifies parameters, headers, and body content to test for vulnerabilities.

Intruder module automates parameter fuzzing. The tester marks injection points in a request template. The intruder iterates through payloads and records responses.

Scanner configuration tunes the active scanner's aggressiveness. Low aggressiveness avoids denial-of-service but misses some vulnerabilities. High aggressiveness provides thorough coverage.

Extension integration adds community and custom extensions: JWT manipulation, GraphQL testing, authorization testing, and custom authentication handling.

Session handling rules maintain authenticated sessions during scanning. Rules define: login macros, session validation checks, and token extraction patterns.

Collaborator integration detects out-of-band vulnerabilities: blind SSRF, blind XSS, and blind SQL injection. The collaborator server receives callbacks from payloads that trigger external requests.

Traffic logging stores all intercepted traffic in a project file. The project file enables: session review, finding correlation, and report generation.

9.1 Engagement Methodology

Pre-engagement preparation includes: scope definition, rules of engagement, emergency contacts, communication channels, and legal authorization. A signed statement of work documents all agreements.

Information gathering (OSINT) collects publicly available information: DNS records, WHOIS data, social media, job postings, document metadata, and exposed credentials in breach databases.

Threat modeling identifies likely attack vectors based on the target's technology stack, industry, and threat landscape. The model prioritizes testing efforts on the highest-risk areas.

External testing assesses the target's internet-facing infrastructure: web applications, VPN endpoints, email servers, and cloud services. External testing simulates an attacker without internal access.

Internal testing simulates an attacker who has gained access to the internal network. Testing covers: Active Directory attacks, network segmentation, internal services, and lateral movement.

Social engineering testing evaluates human security: phishing emails, phone pretexting, and physical access attempts. Results measure employee security awareness.

Wireless testing assesses wireless network security: WPA/WPA2 configuration, rogue access points, and wireless client attacks. Testing covers both corporate and guest wireless networks.

Physical security testing assesses: badge cloning, tailgating, lock picking, and server room access. Physical testing requires explicit authorization and is conducted with site escorts.

Cloud security testing evaluates: IAM policies, storage bucket permissions, serverless function security, and cloud service configuration. Testing covers AWS, Azure, and GCP environments.

Post-engagement activities include: evidence cleanup, findings debrief, report delivery, and remediation support. The engagement formally closes with client acknowledgment.

5.3 Mobile Application Testing

Android application testing uses: apktool (decompilation), jadx (Java decompilation), Frida (dynamic instrumentation), and drozer (security assessment framework).

iOS application testing uses: Clutch (app decryption), class-dump (Objective-C metadata), and Frida for runtime manipulation. Testing requires a jailbroken device or simulator.

Static analysis for mobile apps examines: embedded secrets (API keys, credentials), insecure data storage, weak cryptography, and hardcoded URLs.

Dynamic analysis intercepts network traffic, monitors file system changes, and traces function calls. The analysis identifies: certificate pinning bypass opportunities, insecure communication, and data

leakage.

API endpoint discovery extracts API URLs from the application binary and network traffic. Discovered endpoints are tested for authentication, authorization, and input validation vulnerabilities.

Root/jailbreak detection bypass circumvents protections that prevent running on rooted/jailbroken devices. Frida scripts hook detection functions and return false results.

Local data storage analysis examines: SQLite databases, shared preferences, keychain entries, and cached data. The analysis identifies sensitive data stored without encryption.

Binary protections analysis checks for: code obfuscation, anti-tampering, and debugger detection. The analysis assesses the difficulty of reverse engineering the application.

Push notification testing verifies that sensitive data is not included in push notification payloads. Notifications are captured and analyzed for information disclosure.

Mobile app signing verification checks that the application is signed with the expected certificate. Re-signing with a different certificate indicates tampering.

7.3 Privilege Escalation

Linux privilege escalation checks: SUID/SGID binaries, writable cron jobs, kernel vulnerabilities, misconfigured sudo rules, and writable PATH directories.

Windows privilege escalation checks: unquoted service paths, writable service binaries, AlwaysInstallElevated, stored credentials, and kernel exploits.

Docker container escape techniques include: mounting the host filesystem, exploiting privileged containers, and leveraging Docker socket access.

Kubernetes privilege escalation exploits: misconfigured RBAC, pod security policies, service account tokens, and etcd access.

Database privilege escalation: SQL Server xp_cmdshell, PostgreSQL COPY to file, MySQL FILE privilege, and Oracle Java stored procedures.

Active Directory privilege escalation: unconstrained delegation, constrained delegation abuse, DCSync (replicating AD credentials), and Golden Ticket (forging Kerberos tickets).

Cloud privilege escalation: IAM role assumption, instance metadata service exploitation, and cross-account access through trust relationships.

Automated enumeration tools: LinPEAS scans Linux systems for privilege escalation vectors. WinPEAS scans Windows systems. Both tools produce color-coded output highlighting critical findings.

Custom privilege escalation scripts target organization-specific software: internal tools with elevated privileges, custom services running as root/SYSTEM, and application-specific vulnerabilities.

Privilege escalation documentation records: initial access level, escalation technique, evidence of elevated access, and the specific misconfiguration that enabled escalation.

2.4 Security Hardening

Full disk encryption protects data at rest. NullSec uses LUKS2 with AES-256-XTS. The encryption password is required at boot. A secondary key slot supports hardware security keys.

Network anonymization routes all non-engagement traffic through Tor. A firewall rule prevents traffic leakage. The Tor configuration uses bridge relays to avoid Tor detection.

MAC address randomization changes the wireless MAC address on each connection. Randomization prevents tracking across wireless networks. The original MAC can be restored for specific engagements.

Memory encryption (where supported) protects against physical memory attacks. AMD SEV and Intel TME encrypt memory contents. The encryption is transparent to the operating system.

Secure deletion of engagement data uses multiple overwrite passes followed by trim (for SSDs). Deleted data cannot be recovered with forensic tools.

Anti-forensics tools help detect whether the NullSec system has been examined. The tools check for: forensic tool artifacts, file access time modifications, and unauthorized copies of engagement data.

VPN configuration provides pre-configured OpenVPN and WireGuard profiles. VPN connections encrypt all traffic between the tester and the engagement network.

Firewall configuration defaults to deny-all inbound and allow-all outbound. Specific inbound rules are added for reverse shells and callback listeners during engagements.

Audit logging records all tool executions with timestamps and parameters. Audit logs provide evidence of testing activities for legal protection and engagement documentation.

Automatic screen lock activates after 5 minutes of inactivity. The lock screen requires the user password. Critical engagements can enable immediate lock on lid close.

4.3 API Security Assessment

REST API testing verifies: authentication mechanisms (API keys, OAuth, JWT), authorization (role-based access), input validation, and rate limiting. Each endpoint is tested for OWASP API Top 10 vulnerabilities.

GraphQL security testing targets: introspection disclosure, nested query depth attacks (denial of service), batch query abuse, and injection through query variables.

WebSocket security testing verifies: authentication during handshake, message validation, origin checking, and encryption (WSS). Message fuzzing tests input handling.

gRPC security testing examines: service reflection disclosure, authentication (mTLS, token-based),

and input validation of Protocol Buffer messages.

API documentation analysis (Swagger/OpenAPI) identifies: undocumented endpoints, excessive data exposure, and missing authentication on sensitive operations.

API rate limiting verification tests: per-client limits, per-endpoint limits, and bypass techniques (header manipulation, IP rotation). Exceeding limits should return 429 status.

API versioning security checks for: deprecated endpoints still accessible, version-specific vulnerabilities, and inconsistent security across versions.

Token security analysis examines: JWT algorithm confusion (none, RS256 to HS256), token lifetime, refresh token rotation, and token revocation.

API error handling analysis checks that error responses do not leak: stack traces, database queries, internal paths, or system information.

API business logic testing verifies: workflow enforcement, transaction integrity, race conditions, and abuse of discount/referral systems.

6.3 Wireless Credential Capture

Evil portal captures credentials through a fake authentication page. The portal mimics the target organization's login page. Captured credentials are stored in an encrypted database.

WPA Enterprise (EAP) attacks intercept RADIUS authentication. Hostapd-wpe creates a rogue RADIUS server that captures MSCHAPv2 hashes. Hashes are cracked offline with hashcat.

Cookie capture through wireless interception uses ARP spoofing to redirect traffic through the attacker's machine. Session cookies are extracted from HTTP traffic for session hijacking.

MITM certificate interception presents fraudulent certificates for HTTPS connections. SSLstrip downgrades HTTPS to HTTP. HSTS bypass techniques defeat strict transport security.

Credential relay attacks forward captured authentication to legitimate services. NTLM relay forwards Windows authentication to access shares, execute commands, or escalate privileges.

NetNTLM hash capture uses Responder to poison LLMNR, NBT-NS, and mDNS queries. Clients send NetNTLM hashes when connecting to the attacker's spoofed services.

Kerberos ticket capture intercepts Kerberos authentication on the network. Captured TGT and service tickets enable pass-the-ticket attacks.

VPN credential testing checks for: default VPN credentials, certificate-based authentication weaknesses, and split-tunnel leakage that exposes internal traffic.

Clear-text protocol monitoring captures credentials transmitted in clear text: FTP, Telnet, HTTP Basic Auth, SNMP community strings, and SMTP authentication.

Credential spray via wireless uses the captured wireless credentials to test access to other services.

Password reuse across services is common and often provides additional access.

3.4 Passive Reconnaissance

DNS passive reconnaissance queries public DNS records: A, AAAA, MX, NS, TXT, SOA, and SRV records. Certificate Transparency logs reveal subdomains from issued certificates.

WHOIS analysis extracts domain registration information: registrant name, organization, email, registration date, and name servers. Historical WHOIS records track ownership changes.

Search engine dorking uses Google/Bing advanced operators to find: exposed documents, directory listings, login pages, and error messages. Custom dork lists target specific technologies.

Social media OSINT collects information from LinkedIn, Twitter, GitHub, and other platforms. Employee names, job titles, technology stacks, and organizational structure inform the assessment.

Code repository analysis searches GitHub, GitLab, and Bitbucket for: exposed credentials, API keys, internal URLs, and infrastructure details in the target organization's repositories.

Document metadata extraction from published documents (PDF, DOCX, XLSX) reveals: author names, software versions, internal paths, and printer names.

Breach database queries check whether the target's email addresses appear in known data breaches. Exposed credentials are tested against the target's authentication endpoints.

Shodan and Censys searches identify internet-facing devices associated with the target. Search results reveal: open ports, running services, SSL certificate details, and device types.

BGP and ASN analysis maps the target's IP address ranges. BGP looking glasses and RIR databases identify owned address space. Broader address ranges reveal additional attack surface.

Email header analysis from publicly available emails reveals: internal mail server names, IP addresses, and mail routing paths.

8.3 Engagement Workflow Automation

Automation scripts coordinate multi-phase engagements. Phase transitions trigger: tool configuration updates, notification emails, and status dashboard updates.

Scan scheduling runs reconnaissance scans during specified time windows. Scheduling avoids peak business hours and complies with rules of engagement. Results are collected automatically.

Finding deduplication removes duplicate findings from parallel testing activities. Deduplication compares: vulnerability type, affected host, and evidence. Unique findings are retained.

Status dashboard provides real-time engagement progress: hosts tested, vulnerabilities found, exploitation status, and time remaining. The dashboard is accessible through a web browser.

Notification system alerts the team of: critical findings, tool completion, and scope changes.

Notifications are sent via Slack, email, or desktop alerts.

Evidence collection automation captures screenshots, tool output, and network traffic at predetermined points. Automatic collection ensures comprehensive evidence without manual intervention.

Tool orchestration coordinates multiple tools running in parallel. The orchestrator manages: tool dependencies, resource allocation, and output aggregation.

Engagement calendar tracks scheduled engagements, team assignments, and resource allocation. The calendar integrates with project management tools.

Findings tracking database stores all findings with their lifecycle: discovered, confirmed, reported, remediation planned, and verified. The database supports filtering and searching.

Post-engagement cleanup automation removes: test accounts, uploaded files, persistence mechanisms, and network configurations. Cleanup verification confirms complete removal.

9.2 Legal and Ethical Considerations

Authorization documentation includes: signed penetration testing agreement, scope definition, emergency contacts, and rules of engagement. Testing without authorization is illegal.

Scope boundaries define exactly which systems, networks, and applications are included in the assessment. Out-of-scope systems are explicitly listed. Accidental out-of-scope testing must be reported immediately.

Data handling procedures define how sensitive data encountered during testing is handled: encryption requirements, retention period, and destruction method.

Third-party systems encountered during testing (cloud providers, CDN, shared hosting) require additional authorization from the third-party provider.

Responsible disclosure procedures apply when testing reveals vulnerabilities in third-party software. The vulnerability is reported to the vendor before public disclosure.

Evidence chain of custody maintains the integrity of testing evidence from collection through reporting. Each evidence item is hashed and timestamped.

Professional liability insurance protects the testing team against claims arising from testing activities. Insurance coverage includes: data breach, system damage, and professional errors.

Compliance with local laws ensures that testing activities comply with: Computer Fraud and Abuse Act (US), Computer Misuse Act (UK), and equivalent legislation in other jurisdictions.

Ethical guidelines prohibit: accessing data beyond what is necessary, causing intentional damage, exfiltrating real data, and disclosing findings to unauthorized parties.

Continuous professional development requires testers to maintain certifications (OSCP, GPEN,

GXPN) and stay current with evolving attack techniques and defense mechanisms.

2.5 Tool Integration Framework

Unified output format converts tool-specific output into a common JSON schema. The schema includes: target host, finding type, severity, evidence, and timestamp. All tools export to the common format.

Tool chain automation connects tool outputs to tool inputs. Nmap scan results feed into vulnerability scanners. Credential dumps feed into lateral movement tools. The chain reduces manual data transfer.

Result aggregation merges findings from multiple tools into a unified view. Aggregation deduplicates findings, correlates related evidence, and presents a consolidated target picture.

Tool wrapper scripts add: logging, evidence capture, and output format conversion around each tool. Wrappers provide consistent behavior without modifying the underlying tools.

Plugin architecture allows custom tools to integrate with the NullSec framework. Plugins implement a standard interface: initialize, execute, and report. The plugin manager handles lifecycle.

Tool dependency resolution ensures that required tools and libraries are available before execution. Missing dependencies are installed automatically from the NullSec repository.

Concurrent tool execution runs independent tools in parallel on separate CPU cores. The executor manages resource limits (CPU, memory, network bandwidth) to prevent interference.

Tool update notification alerts the user when newer versions of tools are available. Critical security updates are highlighted. The update process preserves custom configurations.

Tool documentation integration provides man pages, usage examples, and NullSec-specific tutorials for every included tool. Documentation is searchable from the desktop environment.

Tool benchmarking measures execution time and resource usage for common operations. Benchmarks identify performance bottlenecks and guide optimization efforts.

9.3 Community and Ecosystem

NullSec community forums provide a platform for users to share techniques, custom tools, and engagement experiences. Forums are moderated to prevent discussion of unauthorized testing.

Tool contribution process allows community members to submit new tools for inclusion. Submissions are reviewed for: code quality, security (no backdoors), license compatibility, and usefulness.

Training materials include: video tutorials, hands-on labs, and certification preparation guides. Labs use purpose-built vulnerable environments (HackTheBox-style challenges).

Bug bounty integration connects NullSec tools with popular bug bounty platforms (HackerOne, Bugcrowd). Scope files are imported automatically. Findings are formatted for platform submission.

Conference presentations share NullSec development progress, new features, and security research. Annual NullSec conference brings together users and developers.

Industry partnerships with security vendors enable tool integration, beta testing, and feature requests. Partnerships ensure NullSec stays current with commercial tool developments.

Academic collaboration provides NullSec as a teaching platform for university security courses. Educational licenses are free. Course materials include lab exercises using NullSec tools.

Localization support translates the NullSec interface and documentation into multiple languages. The translation process uses community contributions and professional review.

Accessibility features include screen reader compatibility, keyboard-only navigation, and high-contrast themes. Accessibility ensures all security professionals can use NullSec effectively.

Roadmap planning involves community input through feature requests and voting. The development team prioritizes features based on community demand, security impact, and implementation feasibility.

Plugin marketplace provides a curated collection of community-developed plugins. Each plugin is reviewed for security and compatibility. Popular plugins are promoted on the NullSec website.

Automated environment setup provisions vulnerable lab environments for practice. Environments include: intentionally vulnerable web applications, Active Directory forests, and network simulations.

References

[1] Offensive Security. Kali Linux Documentation. 2024.
[2] OWASP Foundation. OWASP Testing Guide v4. 2020.
[3] PTES. Penetration Testing Execution Standard. 2014.
[4] NIST. SP 800-115: Technical Guide to Information Security Testing. 2008.
[5] Weidman, G. Penetration Testing: A Hands-On Introduction. No Starch Press, 2014.