

Florida International University

Spring 2026 Senior Design Project



Vulnerd

Team Members: Jonathan Nava-Arenas, Reynaldo Rodriguez, Luisa Faria Novy E Silva, Anthony Whiteman, Rickoy Cunningham

Instructor/Faculty: Masoud Sadjadi Florida International University

Problem Definition

Most cybersecurity tools assume you have a dedicated security team to interpret the output. Small businesses and under-resourced organizations don't have that luxury. They're left with scanner exports full of CVE codes, severity numbers, and technical jargon with no clear guidance on what actually matters or what to do about it. The result? Vulnerabilities go unaddressed, not because the tools don't exist, but because the people who need them most can't make sense of what they're saying. Vulnerd was built to sit in that gap turning raw scan data into clear, prioritized, actionable reporting without requiring a dedicated analyst to translate it.

Key points:

- 43% of cyberattacks target small businesses, yet most lack dedicated security staff
- Existing tools (Nessus, etc.) produce powerful but overwhelming output
- The gap isn't in scanning capability — it's in interpretation and response
- No single detection method catches everything; a layered approach is essential



System Architecture

- Hardware: Raspberry Pi 5 running headless Raspberry Pi OS Lite (64-bit) with USB solid-state storage demonstrating that effective security labs don't require enterprise hardware
- Application: DVWA (Damn Vulnerable Web Application) deployed via Docker Compose with an isolated MariaDB backend, exposed on port 8081 containerization ensures host OS isolation
- Access Control: Tailscale VPN (WireGuard-based) restricts all access to authenticated team members no services are publicly exposed to the internet
- Scanning: Nessus Essentials configured with SSH credentials (vulnerd user) and HTTP form authentication (admin/password) for deep credentialed scanning
- Testing Tools: Browser and terminal (curl) for manual attack execution across 8 DVWA modules at Security Level Low

1. Collect — Users upload a Nessus CSV export through a simple drag-and-drop web interface. That's the only user effort required.
2. Parse — A custom Python engine reads the raw export, cleans it, normalizes the data, and structures it into analyzable input. Nessus exports aren't friendly even to other programs, so this stage handles the heavy lifting.
3. Analyze — Google Gemini AI reads the cleaned findings and generates a full report card. It grades the network, identifies the biggest risks, explains each vulnerability in plain English, and provides prioritized next steps.
4. Comply — The system maps findings to the compliance framework relevant to the user's industry (HIPAA, PCI-DSS, NIST 800-53, FERPA, or CIS Controls) and builds a compliance dashboard with scores, real-world breach stories, and cost comparisons.
5. Deliver — Everything displays in the browser: the report card, compliance dashboard, a built-in AI chat assistant with context on the specific scan, and the option to email a polished PDF report to stakeholders.

Implementation: Automated Scanning

- Basic Network Scan (Credentialed via SSH): Most productive scan and found critical vulnerabilities in system libraries (libnss3, libgnutls30), outdated Vim packages, weak SSH algorithms, and OS/network fingerprinting exposure
- Web Application Tests (HTTP form auth): Found clickjacking vulnerability (missing X-Frame-Options/CSP headers) and cleartext credential transmission, but failed to detect any injection vulnerabilities
- Advanced Scan: Second pass with expanded plugin set and SSH credentials confirmed findings from Basic scan with additional detail
- Advanced Dynamic Scan: Initially failed when Info-level plugins were filtered out, revealed that Nessus requires discovery-stage plugins as prerequisites for higher-severity checks
- Malware Scan: Intentionally skipped, designed for detecting malware on Windows/Unix systems, not application-layer coding vulnerabilities in a containerized DVWA environment

Detection Engineering

The team built a Detection Playbook containing 15 total detections 9 sourced from Nessus scan findings and 6 identified exclusively through manual testing. Each detection follows a consistent structure: detection name, related plugins, threat context, plain-language trigger logic (IF/THEN), priority rating, and recommended response. This playbook turns raw alerts into repeatable "if this happens, do that" rules.

VULNERD PROJECT	
Detection Playbook — Version 2	
Purpose The purpose of the detection playbook is basically a guide for what we want to catch and how we react to it. Instead of treating every Nessus finding or log entry as random noise, the playbook lists the key patterns we care about (for example, "web app leaking version info" or "failed SSH credentials"), explains why they matter, and describes what should happen when we see them. It turns raw technical alerts into clear "if this happens, then do that" rules. This helps the team stay consistent, focus on the most important signals, and respond faster when something suspicious shows up in our scans.	
Structure Each detection entry in this playbook follows a consistent structure: • Detection name • Related Nessus plugins (by name or ID) • What it means (threat / risk) • Trigger logic (in plain IF/THEN terms) • Priority (Low / Medium / High / Critical) • Recommended response	
Data Sources This playbook was built from findings across multiple scan types and manual attack testing performed against the VULNERD lab environment (DVWA on Raspberry Pi 5, accessed via Tailscale VPN on port 6881). The data sources include: • Basic Network Scan (Credentialed via SSH): 1 Critical, 2 High, 6 Medium, 1 Low, 72 Info findings • Web Application Tests Scan: 0 Critical, 0 High, 1 Medium, 1 Low, 24 Info findings • Advanced Dynamic Scan: 1 Critical, 0 High, 0 Medium, 0 Low, 11 Info findings • Manual Attack Testing: Brute Force, Command Injection, SQL Injection, XSS (Reflected, Stored, DOM), File Inclusion, File Upload — all successfully exploited on DVWA (Security Level Low)	
Detections	
Detection 1 — Critical System Library Vulnerability (libnss3)	
Field	Details
Plugins	300454 — Debian dsa-6149, libnss3 - security update
What it means	The Network Security Services (NSS) library on the host has a known critical vulnerability. NSS handles SSL/TLS and cryptographic operations. A compromised NSS library could allow an attacker to intercept encrypted communications, forge certificates, or execute arbitrary code. This was detected by both the Basic Network Scan and the Advanced Dynamic Scan, confirming it is a real and persistent issue.
Trigger logic	IF a credentialed scan detects that the installed libnss3 package version is older than the version specified in Debian Security Advisory DSA-6149, THEN raise a Critical alert.
Priority	Critical
Recommended response	Apply the Debian security update immediately using: <code>sudo apt update && sudo apt upgrade libnss3</code> . Verify the update was applied by re-scanning. This library is fundamental to encrypted communications on the system.
Detection 2 — Outdated System Libraries with Known Vulnerabilities	
Field	Details
Plugins	299347 — Debian dsa-6138, libpng-dev - security update (High) 299381 — Debian dsa-6140, gnutils-bin - security update (Medium)
What it means	Multiple system libraries (libpng for image processing, GnuTLS for secure communications) have known vulnerabilities with available patches. These outdated libraries could be exploited to cause denial of service, information disclosure, or in some cases remote code execution. In a small organization without dedicated security staff, unpatched libraries are one of the most common attack vectors.
Trigger logic	IF a credentialed scan identifies installed packages with versions older than those specified in Debian Security Advisories (DSA-6138, DSA-6140), THEN raise an alert at the highest severity level among the affected packages.
Priority	High
Recommended response	Run <code>sudo apt update</code> and <code>sudo apt upgrade</code> . Re-scan and verify.

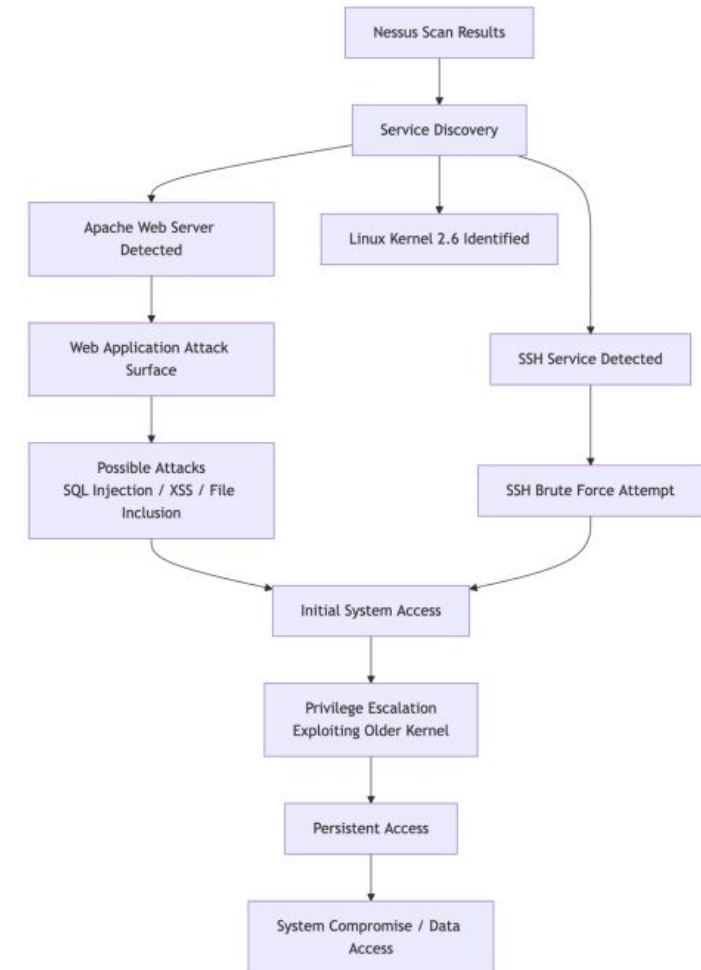
Implementation: Manual Testing

- Brute Force: Automated password guessing via curl loop with no rate limiting or lockout, succeeded on default credentials, demonstrating weakest-link authentication risk
- Command Injection: Used semicolon operators (127.0.0.1;whoami) to execute arbitrary system commands, retrieved /etc/passwd and kernel version, proving full server control
- SQL Injection: Injected ' OR '1'='1 payloads to dump the entire user database including password hashes, OWASP Top 10 vulnerability, primary cause of real-world data breaches
- XSS (Reflected, Stored, DOM): Three distinct attack vectors tested — Stored XSS persists in the database and fires for every visitor; DOM-based XSS exploits client-side JavaScript
- File Inclusion (LFI): Path traversal (../../etc/passwd) exposed the entire server file system; PHP leaked internal file paths through error messages
- File Upload: Uploaded a PHP webshell with zero validation, server accepted and executed it, giving persistent remote backdoor access that survives restarts

Implementation: CTI Analysis & VulNerd Tool

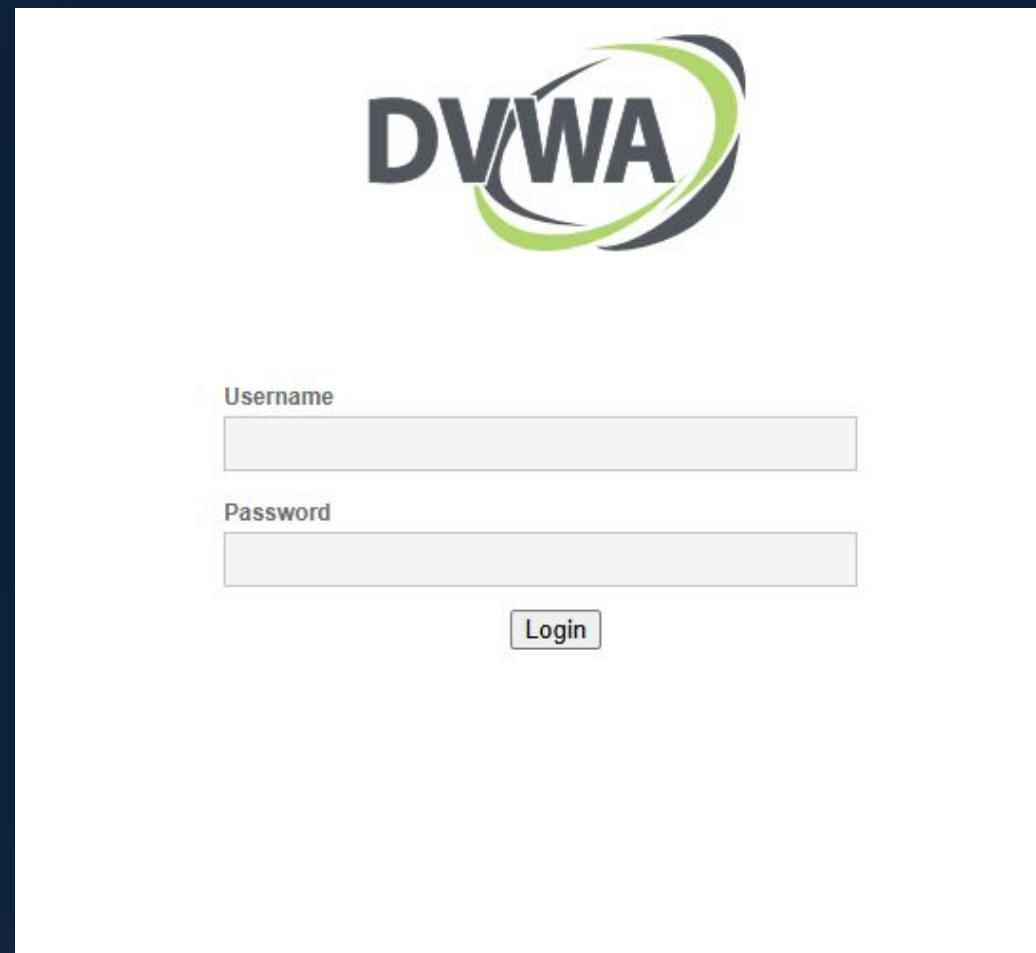
- **Cyber Threat Intelligence:** Luisa Faria analyzed scan results through a threat intelligence lens mapping discovered services to potential attack paths and real-world attacker behavior
- **MITRE ATT&CK Mapping:** Findings mapped to 5 framework techniques T1595 (Active Scanning), T1046 (Network Service Discovery), T1190 (Exploit Public-Facing App), T1110 (Brute Force), T1059 (Command Execution)
- **Attack Chain Modeling:** CTI report modeled a full attack scenario from reconnaissance through service discovery, exploitation, privilege escalation, and persistent access
- **VulNerd Web Tool:** Parses Nessus CSV exports through a 5-stage pipeline to Collect, Parse, Analyze (Google Gemini AI), Comply (industry-specific frameworks), and Deliver (report card with grades, compliance dashboard, and interactive AI chat)
- **Compliance Coverage:** Maps findings to HIPAA, PCI-DSS, NIST 800-53, FERPA, and CIS Controls based on industry type, includes cost comparison of fixing vs. breach impact

V. Example of Threat Scenario



Demonstration Overview

- **Live Lab Access:** Connect to DVWA through Tailscale VPN, showing the containerized environment running on the Raspberry Pi 5
- **Nessus Scan Comparison:** Walk through scan results across all four scan profiles, highlighting what each found and missed
- **Manual Attack Demo:** Execute a live command injection or SQL injection attack against DVWA at Security Level Low, demonstrating the vulnerability that all automated scans missed
- **Detection Playbook v2:** Show the 15-detection playbook with IF/THEN trigger logic, priority ratings, and recommended responses
- **VulNerd Tool Demo:** Upload a Nessus CSV → watch AI analysis generate a graded report card → view compliance dashboard → interact with the built-in chat assistant



Testing and Evaluation Results



The Central Finding

40% of all detections including 3 of 4 Critical-priority findings — were identified exclusively through manual testing and were completely missed by every automated Nessus scan. This validates the project's core thesis: no single detection method catches everything.

Automated Scanning Results (Nessus)

Four scan types were run, producing 9 of the 15 total detections. The credentialed Basic Network Scan was the most productive by a significant margin, uncovering system-level exposures like OS fingerprinting, PHP version leakage, web application exposure on non-standard ports, and SSH configuration issues. The Web Application Tests failed to detect any injection vulnerabilities despite being properly authenticated against DVWA a notable limitation of automated web scanning. The Advanced Dynamic Scan revealed that filtering out Info-level plugins broke the scan pipeline entirely, because higher-severity plugins depend on those low-level discovery stages to function.

Manual Testing Results

Eight targeted attacks were executed against DVWA, producing 6 detections that no automated scan found. These represent some of the most dangerous vulnerability categories in web application security:

- **Command Injection (Critical)** — Arbitrary system commands executed through user input
- **SQL Injection (Critical)** — Full database contents extracted through form fields
- **Unrestricted File Upload (Critical)** — Malicious files uploaded to the server without restriction
- **Cross-Site Scripting** — Reflected, Stored, and DOM variants all successfully exploited
- **File Inclusion (High)** — Server-side files accessed through manipulated parameters
- **Brute Force Authentication (High)** — Weak credentials compromised through repeated login attempts

Detection Breakdown by Source

- **Nessus only: 9 detections (60%)** — mostly Medium and Low priority, focused on system configuration and information disclosure
- **Manual only: 6 detections (40%)** — included 3 Critical and 2 High priority findings targeting application-layer vulnerabilities

Challenges and Lessons Learned

Challenge 1: Nessus Plugin Dependencies

When configuring the Advanced Dynamic Scan, the team attempted to filter out Info-level plugins to reduce noise and focus on higher-severity findings. This unexpectedly broke the entire scan pipeline. Info-level plugins handle foundational discovery tasks like service detection and port identification that higher-severity plugins depend on as prerequisites. The lesson: severity filtering in automated scanners must account for plugin dependency chains, not just individual severity ratings.

Challenge 2: Automated Scanner Blind Spots

The Nessus Web Application Tests were properly configured with DVWA authentication credentials, yet they failed to detect any injection vulnerabilities. SQL injection, command injection, XSS, and file inclusion all went undetected. This was initially surprising but became one of the project's most important findings. Automated web scanners interact with applications differently than a human attacker would, and many application-layer vulnerabilities require contextual understanding of form behavior, input handling, and application logic that scanners simply don't replicate well.

Challenge 3: Credentialed vs. Uncredentialed Scanning

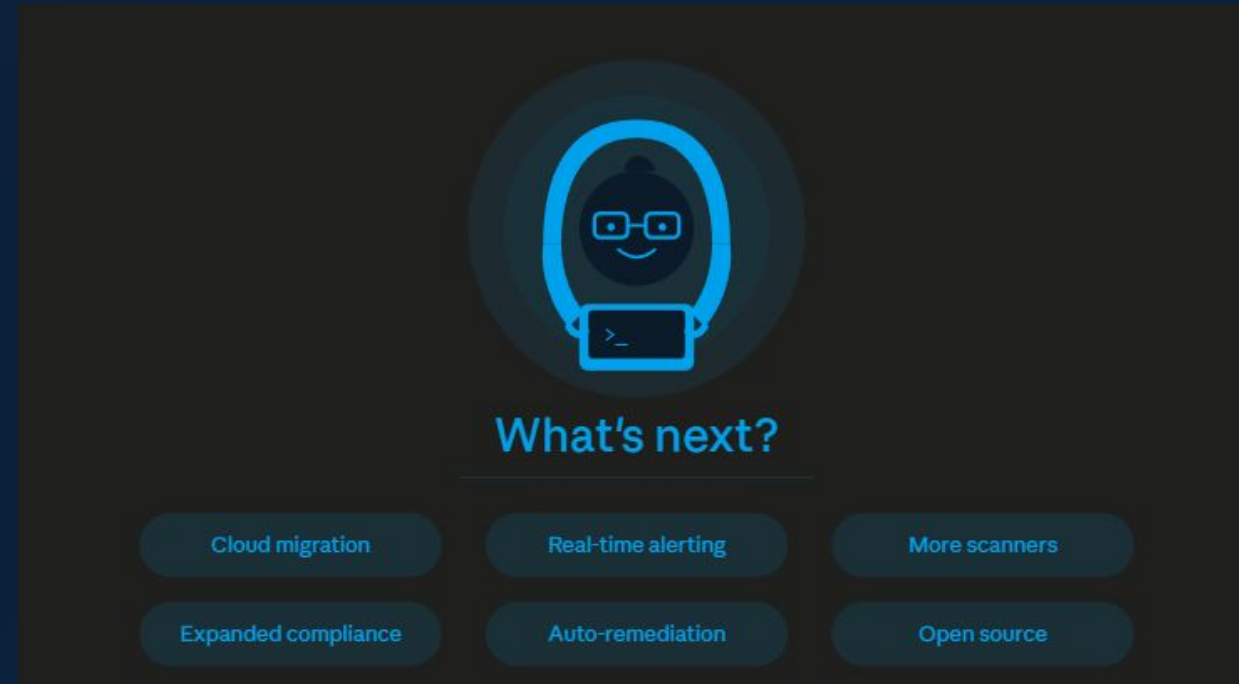
Early unauthenticated scans returned mostly surface-level information disclosure findings. Once credentialed SSH access was configured for the Basic Network Scan, the depth and quality of results improved dramatically. The lesson reinforced that scan configuration matters as much as scan selection the same tool produces vastly different results depending on the access level it's given.

Overarching Lesson

No single tool, scan type, or methodology provides complete coverage. The project repeatedly demonstrated that combining approaches and understanding the limitations of each is what produces comprehensive security visibility. This is the core message for organizations that rely on a single scanner and assume they're covered.

Future Work

- Expand VulNerd to accept scan formats beyond Nessus CSV. OpenVAS, Qualys, and Burp Suite exports would broaden accessibility for organizations using different scanners
- Integrate real-time log monitoring and automated alerting based on the Detection Playbook's IF/THEN trigger logic turning static playbook rules into active detection capabilities
- Develop role-based views so different team members (IT admin, manager, compliance officer) see findings relevant to their responsibilities rather than a raw vulnerability dump
- Possibly explore integrating dedicated DAST tools (Burp Suite, OWASP ZAP) into the pipeline to close the application-layer detection gap demonstrated by this project



Thank You



VULNERD

FIU

