HACK KRMU 5.0

- PROBLEM STATEMENT ID : PS1
- TEAM NAME: SHUBHAM RATHORE
- TEAM ID : HK-197
- TEAM MEMBERS : ADITYA TRIPATHI, SAURAV KUMAR, YASHIKA BATRA

PROBLEM & SOLUTION

The Modern Application Stack

SPA Frameworks

Heavy reliance on React, Vue, Angular with dynamic client-side rendering creates "black boxes" for static crawlers.

API-First Design

Microservices & REST/GraphQL APIs often lack proper schema documentation exposure.

Where Legacy Scanners Fail

- **Inability to Render JavaScript**
  Traditional spiders miss 60%+ of the attack surface in SPAs.
- **Auth & Session Blindness**
  Cannot maintain complex token-based sessions (JWT/OAuth) to test deep internal logic.
- **Logic Flaws Missed**
  Broken Access Control (BAC) & BOLA are undetectable by signature-based matching.

STANDARD ALIGNMENT

NIST SP 800-53 OWASP ASVS

CRITICAL RISK CONTEXT

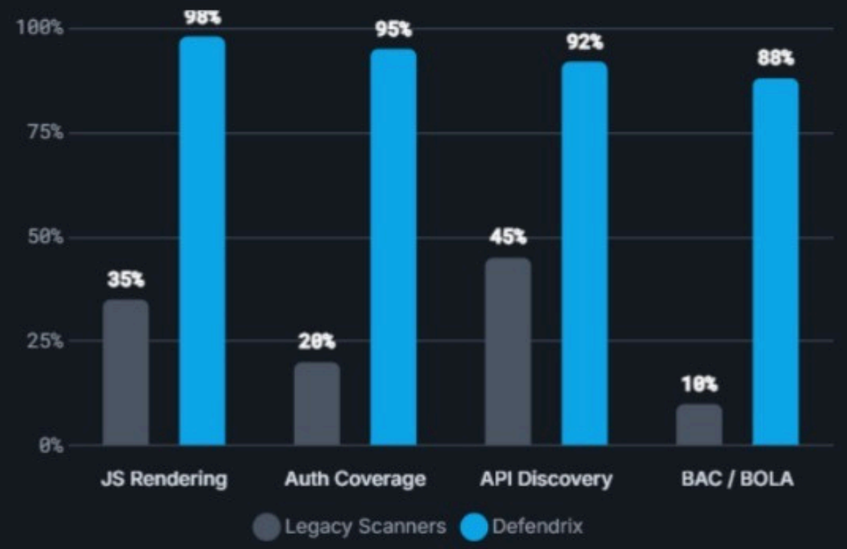
OWASP Top 10 (2021)

#1 Risk: **Broken Access Control**

01

Coverage Gap Analysis

Legacy DAST vs. Defendrix Advanced



FLOW OF SOLUTION

Defendrix Advanced is a **Dynamic Application Security Testing (DAST)** platform engine designed to map, render, and exploit vulnerabilities in modern, API-driven architectures.

01
Intelligent Rendering
Headless Chromium engine executes complex JavaScript to discover hidden SPA routes and DOM states.

02
Authenticated Mapping
Manages complex session states, JWTs, and OAuth flows to test deep application logic.

03
Auto-BAC Testing
Identifies Broken Access Control by replaying privileged requests across lower-privilege roles.

04
OAST Detection
Detects blind vulnerabilities via out-of-band DNS and HTTP interaction callbacks.

05
Payload Mutation
Context-aware fuzzing engine that adapts attack payloads based on WAF filtering responses.

06
Behavioral Correlation
Analyzes response timing, error signatures, and content length deviations to confirm flaws.

07
Multi-Vector Logic
Chains distinct low-severity findings to demonstrate complex, high-impact attack paths.

08
Cross-Validation
Reduces false positives by verifying findings through secondary detection methods.

COMPREHENSIVE VULNERABILITY COVERAGE

SQL Injection | XSS (Reflected/Stored/DOM) | SSTI / CSTI | RCE | **Broken Access Control** | **BOLA**

VALUE PROPOSITION

Enterprise-grade coverage, engineered for **SME-ready** operation and cost.

TECH STACK & APPROACH

TECHNOLOGY STACK

- Python 3 – Modular scanning engine
- PySide6 (Qt) – Desktop GUI dashboard
- Requests + BeautifulSoup – HTTP handling & HTML parsing
- Custom Payload Engine – SQLi, XSS, SSTI detection
- VirusTotal API – Passive threat intelligence enrichment

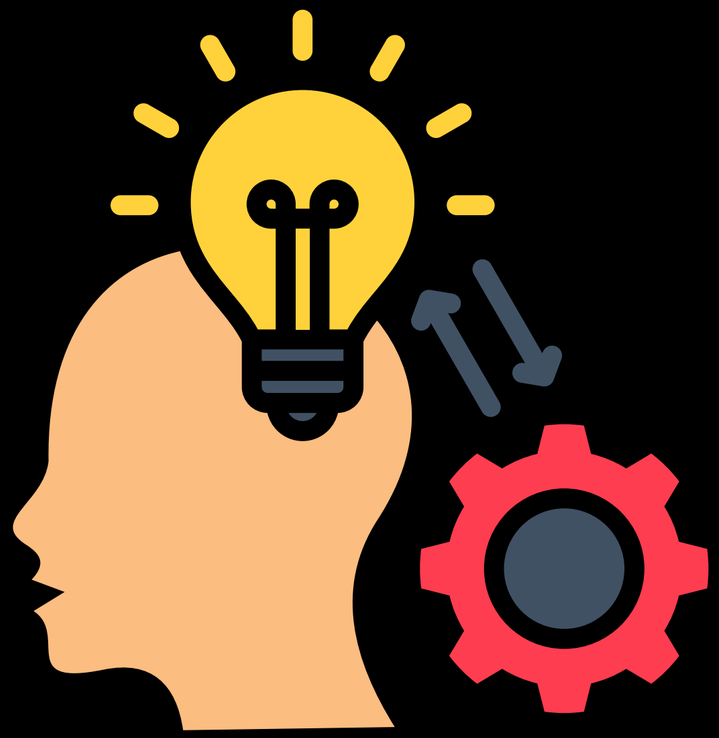
ARCHITECTURE PRINCIPLES

- Hybrid Active + Passive Scanner
- Modular Layered Design
- GUI-Independent Core Engine
- Low False-Positive Detection Model
- Expandable Plugin-Based Structure

TECHNICAL APPROACH

- Discover – Depth-based crawling & dynamic endpoint extraction
- Map – Attack surface identification (parameters, forms, headers)
- Simulate – Controlled payload injection (non-destructive testing)
- Analyze – Reflection detection, error matching, response delta comparison
- Classify – Severity scoring with confidence rating & risk aggregation

UNIQUENESS & INNOVATION FACTOR



Advanced Dynamic Scanning
Integration of JavaScript rendering and Out-of-Band Application Security Testing (OAST) to support modern dynamic web applications and detect complex vulnerabilities.

Enterprise-Scale Infrastructure
Implementation of distributed scanning and automated OWASP Top 10 coverage for large-scale enterprise environments.

AI-Assisted Detection
Incorporation of behavior-based anomaly detection and intelligent vulnerability prediction to improve accuracy and minimize false positives.

DevSecOps Integration
CI/CD pipeline integration and REST API exposure to enable automated security testing within development workflows.

Scalable Cloud-Ready Architecture
Cloud deployment support with role-based reporting dashboards to transform SentinelLite into an enterprise-grade security platform.

FEASIBILITY & CHALLENGES

Modular and Lightweight Architecture
Built using a Python-based modular design with a focused dependency stack, ensuring practical implementation and manageable resource usage.

Stable and Controlled Scanning
Depth-limited crawling and non-destructive payload injection ensure stable performance within realistic computational constraints.

Flexible and Scalable Engine Design
GUI-independent scanning engine enables flexible deployment and future scalability without major architectural changes.

Technical Challenges in Web Scanning
Handling dynamic JavaScript-driven applications, session-based authentication complexity, and minimizing false positives remain core technical challenges.

Accuracy and Ethical Considerations
Careful payload design, intelligent response analysis, and iterative refinement are required to balance detection accuracy, performance, and ethical non-destructive testing.