International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

ASSESSARC: A MOBILE APPLICATION FOR ANDROID SECURITY ASSESSMENT AND PENETRATION TESTING

Prof. Rushikesh S. Bhalerao[1], Vishal Satle[2], Satyam Katkade[3], Tanvi Sangale[4], Tanishq Medhane[5]
[1] *Assistant Professor, Department of IT, SVIT*
[2]*Bachelors of Engineering, Department of IT, SVIT*
[3]*Bachelors of Engineering, Department of IT, SVIT*
[4]*Bachelors of Engineering, Department of IT, SVIT*
[5]*Bachelors of Engineering, Department of IT, SVIT*

Abstract - In recent times, maintaining safety and academic integrity within educational institutions has become a critical challenge. While CCTV cameras are extensively deployed across campuses and examination halls, they rely heavily on manual monitoring, which is inefficient, prone to human error, and causes delayed responses to critical events. This paper proposes a novel, multi-scenario automated surveillance framework based on advanced deep learning and computer vision techniques. The proposed system is designed to operate in two distinct modes: a Campus Safety mode and an Academic Integrity mode. For violence detection, the architecture integrates YOLOv7 for rapid human detection with a MobileNet-BiLSTM classifier to accurately recognize violent actions. For examination monitoring, the system utilizes Deep Keyframe Detection combined with a Multilayer Perceptron (MLP) enhanced YOLOv8 algorithm (SE-YOLOv8) and a ResNet-based 3D CNN to identify subtle cheating behaviors like passing notes or unauthorized looking. By unifying these dual pipelines into a single deployable framework, this study outlines a scalable solution that minimizes manual invigilation, enhances real-time threat detection, and ensures a secure, fair educational environment.

Key Words: Multi-Scenario Surveillance, Deep Learning, Violence Detection, Smart Proctoring, YOLO Framework, Action Recognition, Deep Keyframe Detection, Computer Vision.

1.INTRODUCTION

The extensive deployment of high-definition video surveillance equipment in educational institutions aims to ensure public safety and maintain the fairness of talent selection during standardized examinations. However, public violence and academic misconduct are both dynamic, fast-occurring events. The current bottleneck in institutional security lies in the excessive reliance on the manual inspection of massive video feeds. This traditional mode of observation leads to a situation of "monitoring without detailed inspection," allowing sporadic cheating behaviors and sudden violent acts to go unnoticed until it is too late.

With the deep penetration of artificial intelligence in the educational sector, leveraging AI to empower electronic surveillance offers a robust solution. This paper proposes an intelligent, multi-scenario surveillance system. Rather than relying on singular object detection algorithms that fail to capture contextual complexity, this framework utilizes a hierarchical approach. It combines the rapid localization capabilities of YOLO-variant models with lightweight temporal classifiers to create an automated, highly responsive environment.

2. BODY OF PAPER

The body of the paper is organized into sections presenting the overview, existing tools, analysis, and proposed concept. In Sec. II-A, Vulnerability Assessment and Penetration Testing (VAPT) is introduced. Existing tools are discussed in Sec. II-B, followed by comparative analysis in Sec. II-C. The research gap is identified in Sec. II-D, and the proposed concept is presented in Sec. II-E.

A. Overview of VAPT

Vulnerability Assessment and Penetration Testing (VAPT) is used to identify and evaluate security vulnerabilities. Most existing tools are desktop-based, limiting mobile accessibility.

B. Existing Tools

Tools such as Nmap, Metasploit, and OpenVAS provide strong capabilities but require desktop environments. Mobile tools like zANTI require root access, while Termux lacks a graphical interface.

C. Comparative Analysis

As discussed in Sec. II-B, desktop tools lack portability, while mobile tools face usability and security limitations.

D. Research Gap

From Sec. II-C, it is observed that no Android-based VAPT tool provides a GUI without requiring root access.

E. Proposed Concept

To address this gap, AssessArc is proposed as a lightweight Android-based VAPT application with a user-friendly interface and no root requirement.

3.LITERATURE REVIEW / RELATED WORK

Vulnerability Assessment and Penetration Testing (VAPT) tools have evolved significantly over the years, predominantly within desktop and server environments. Numerous opensource and commercial solutions exist to identify, assess, and exploit vulnerabilities across networks, applications, and systems. Among these, Nmap remains a foundational utility for network reconnaissance, offering versatile capabilities such as host discovery, port scanning, and OS/service fingerprinting [?]. Its extensible scripting engine further enhances automation and fine-grained probing of network surfaces. Complementing discovery tools, exploitation frameworks like Metasploit have established a standard for structured offensive security testing [?]. Metasploit provides a modular framework for developing,

testing, and executing exploits, integrating payloads, encoders, and post-exploitation modules within a unified interface. This enables security professionals to simulate complex attack chains and validate defensive measures efficiently. Commercial vulnerability scanners, including Nessus and OpenVAS, automate large-scale assessments through continuous scanning, signature-based detection, and compliance auditing [?]. While effective, these tools often demand substantial computational resources, consistent updates, and experienced administration, which limits accessibility for casual users or lightweight environments. On mobile platforms, the development of dedicated VAPT tools has been comparatively limited. Early initiatives such as zANTI and dSploit sought to replicate desktop penetration testing functionality on Android devices [?]. These tools provided features like network mapping, password auditing, and man-in-the-middle (MITM) testing. However, they generally required root access, raising device security and stability concerns. Additionally, the lack of consistent updates and support hindered their long-term viability and compatibility with modern Android versions. Termux introduced a partial solution by enabling a Linuxlike command-line environment on Android [?]. It supports the installation and execution of numerous security tools directly on mobile devices. Nevertheless, its usability remains oriented toward advanced users familiar with terminal operations, and
it lacks the graphical integration necessary for streamlined workflows. The absence of a cohesive interface limits accessibility for non-technical users and impedes productivity in rapid assessment scenarios. Recent research into mobile-based penetration testing and lightweight security frameworks emphasizes modular architectures and hybrid execution models. Studies advocate for offloading resource-intensive computations to external servers or cloud environments while maintaining local control for interaction and data visualization . Such an approach enhances both performance and security by isolating heavy tasks from user-facing components. Building upon these insights, the proposed AssessArc framework introduces a GUI-driven, modular design specifically tailored for Android devices. Unlike prior tools that rely on terminal operations or root permissions, AssessArc focuses on accessibility and ethical automation. It enables users—especially those without desktop environments—to perform essential VAPT operations through an intuitive interface. Moreover, AssessArc's architecture supports simulated and controlled execution models, ensuring responsible testing practices while maintaining system integrity. This positions AssessArc as a practical and educational bridge between mobile usability and professional-grade VAPT capabilitie

TABLE I: Comparative Analysis of Existing Tools

Tool	Platform	Root Required
Nmap/Burp Suite	Desktop/Server	No
Metasploit	Desktop/Server	No
zANTI / dSploit	Android	Often Yes
Termux Utilities	Android (CLI)	Partial (depends on package)
AssessArc(Proposed)	**Android (GUI)**	**No**

4.PROPOSED SYSTEM

A. Architecture

AssessArc adopts a three-layer architecture:

1) User Interface Layer: Android activities/fragments present modules, configuration, and results.

2) Core Engine Layer: Module dispatcher and modulespecific implementations (modular interface for each tool).

3) Result Reporting Layer: Aggregates outputs, stores logs (SQLite), and generates exportable reports (PDF/JSON).

Figure 1 depicts the module interaction and workflow.

B. Implemented Modules

The current AssessArc prototype implements the following modules:

- **Hash Identifier**: Recognizes common hash formats (MD5, SHA-1, SHA-256) using length and pattern heuristics.
- **Hash Cracker:** Dictionary-based and constrained bruteforce methods for offline hash cracking in an authorized environment (prototype simulations used during evaluation).
- **Directory Enumeration (DirSearch):** Wordlist-driven enumeration for discovering hidden directories and files.
- **Password-Protected ZIP Cracker:** Dictionary and iterative attempts to unlock ZIP archives when provided with explicit permission.
- **Brute-force Module: A** configurable credential-testing module for authorized targets only (used in lab-controlled evaluations).

C. Design Principles

- **Modularity:** Each module implements a common interface and can be extended or replaced independently.
- **Safety and Authorization:** The application is designed to operate only on targets explicitly supplied by the user. Heavy or potentially offensive actions are intended to be executed on controlled backends rather than on-device whenever appropriate.
- **Efficiency:** Algorithms are optimized for CPU, memory, and battery constraints typical of mid-range Android devices.

5.METHODOLOGY

A.Development Environment

AssessArc was developed using:

- Android SDK (Kotlin and/or Java)
- SQLite (local result storage)
- Retrofit/OkHttp for optional server communication • PDF generation libraries or Android's PdfDocument API for report export

B. Implementation Detail

Modules implement a consistent workflow:

1) Validate user input (target IP, URL, file path, or hash).
2) Preprocess inputs (normalize URLs, check hash format).
3) Execute module logic (local simulation or API call to server-side engine).
4) Aggregate outputs and annotate severity labels.
5) Persist results and create a human-readable report. To keep the mobile client safe and distributable, AssessArc supports two runtime modes:

• Local Simulation Mode:
Modules return simulated or deterministic results for UI testing and educational demonstrations.

• Server-Assisted Mode: The mobile client dispatches jobs to an authenticated server that runs vetted tools in a controlled environment. The server returns sanitized results to the mobile client for display and storage.

C. Ethical and Legal Controls

AssessArc enforces explicit consent dialogs, logs user authorization for each target, and includes a clear disclaimer that only authorized testing is permitted. The application design avoids embedding exploit payloads or code that would facilitate unauthorized attacks.

Charts

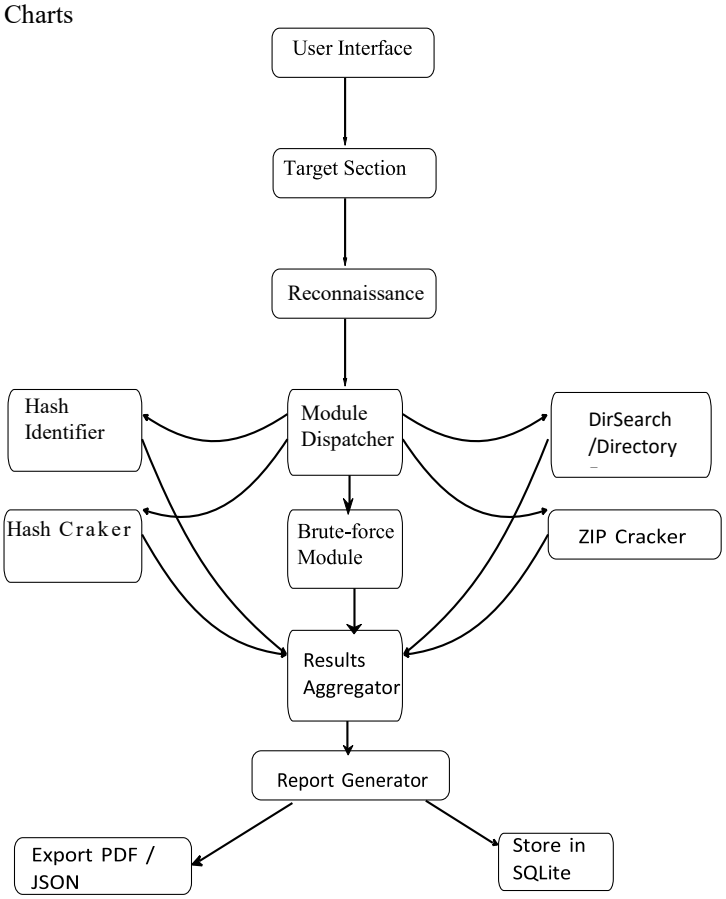

Fig -1: AssessArc module interaction and workflow.

6. RESULTS / PERFORMANCE ANALYSIS

Evaluation used mid-range Android devices to measure performance and resource usage:

- **Device A:** Snapdragon 720G, 6 GB RAM, Android13
- **Device B:** MediaTek Helio G85, 4 GB RAM, Android 12

All active tests were run in a controlled lab network where permission to test was granted.

B. Module Performance

Table II summarizes average execution times and observed success rates for prototype runs. Note: these figures reflect the current prototype behaviour (including local simulations and server-assisted runs where applicable).

C. Resource Utilization

Observed metrics on Device A during active scanning:

- Average CPU utilization: 35–55% (spikes during intensive tasks)
- Memory usage: 150–320 MB (depending on module)
- Battery drain (10 minutes active scanning): ≤ 6%

Module	Average Execution Time	Success Rate
Hash Identifier	1.2 s	98%
Hash Cracker (dictionary)	1-10s per candidate(varies)	90–95% (lab wordlists)
Dir Search (1000-entry wordlist)	10-18s	92%
ZIP Cracker (dictionary)	3-12 spar attempt range	88–92%
Brute-force Module	Depends on wordlist size	80-90% (lab tested)

These results indicate that the implemented modules are practical for mobile-first execution with appropriate throttling and server-assist strategies for heavy tasks.

7. CONCLUSION AND FUTURE SCOPE

AssessArc demonstrates that essential VAPT functionality can be integrated into an Android application with acceptable performance and strong safety controls. The implemented modules (hash identifier, hash cracker, directory enumeration, ZIP cracker, and brute-force module) provide a practical foundation for portable, authorized security assessments.

Future Work:

- Expand local capabilities for non-offensive analysis (e.g., SSL/TLS configuration checks, header analysis).
- Integrate server-assisted scanning for resource-intensive tasks while preserving mobile UX.
- Add machine-learning-based anomaly detection to prioritize findings.
- Provide collaboration features and centralized report management for teams.
- Implement authenticated, auditable workflows and policy enforcement for enterprise usage.

By combining portability with safety controls and modular extensibility, AssessArc offers a feasible path to widen access to security testing for authorized users and learners.

ACKNOWLEDGMENT

The author acknowledges the contributions of testers and peer reviewers who validated prototype modules in controlled laboratory environments. Special thanks to mentors and the academic department for guidance and infrastructure.

REFERENCES

[1] M. Aydos, C. Aldan, E. Cos¸kun, and A. Soydan, "Security testing of Web applications: A systematic mapping of the literature," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 6775–6792, 2022.

[2] K. Vimala and S. Fugkeaw, "VAPE-BRIDGE: Bridging OpenVAS results for automating Metasploit framework," in *IEEE Xplore, 14th International Conference on Knowledge and Smart Technology (KST)*, 2022.

[3] M. Qasaimeh, A. Shamlawi, T. Khairallah, and Jo, "Black Box Evaluation of Web Application Scanners: Standards Mapping Approach," *Journal of Theoretical and Applied Information Technology*, vol. 31, 2018.

[4] CNCS, "National Cybersecurity Reference Framework (QNRCS) –

Portugal," 2021.

[5] J. Couto, "Auditoria de Ciberseguranc¸a: um Caso de Estudo," Ph.D. dissertation, 2018.

[6] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," 2015.

[7] T. Nuno, "Auditoria de Seguranc¸a em Aplicac¸oes na World Wide Web˜ Portuguesa," 2011.

[8] T. Vieira and C. Serrao, "Web applications security and vulnerability˜ analysis financial Web applications security audit – a case study," *International Journal of Innovative Business Strategies*, vol. 2, 2016.

[9] OWASP, "WSTG - Web Security Testing Guide," 2023. [Online].

Available: https://owasp.org

[10] Greenbone, "OpenVAS - Open Vulnerability Assessment Scanner," 2022. [Online]. Available: https://www.openvas.org

[11] Rapid7, "Metasploit — penetration testing software, pen testing security," 2023. [Online]. Available: https://www.metasploit.com

[12] OWASP, "Framework OWASP Top 10," 2017. [Online]. Available: https: //owasp.org/www-project-top-ten/

[13] OWASP, "Free for Open-Source Application Security Tools," 2023. [Online]. Available: https://owasp.org

[14] OWASP, "Project OWASP ZAP," 2020. [Online]. Available: https:// owasp.org/www-project-zap/

[15] A. Ahamed, N. Sadman, A. Khan, I. Hannan, F. Sadia, and M. Hasan, "Automated testing: Testing top 10 OWASP vulnerabilities of government WA in Bangladesh," 2022.

[16] P. E. Black, E. Fong, V. Okun, and R. Gaucher, "Software assurance tools," 2008.

[17] A. Alzahrani, A. Alqazzaz, Y. Zhu, H. Fu, and N. Almashfi, "Web application security tools analysis," in *IEEE Xplore*, pp. 237–242, 2017.

[18] K. Jatinkushwah, S. Dutt, R. Jhunjhunwala, and T. Duggal, "Web application security using VAPT," *IJAEM*, vol. 2, p. 389, 2020.

[19] S. Shah and B. Mehtre, "An automated approach to vulnerability assessment and penetration testing using net-nirikshak 1.0," in *IEEE Xplore*, pp. 707–712, 2015.

[20] A. Reddy, C. A. Kumar, P. Rukmani, and S. Ganapathy, "A new compromising security framework for automated smart homes using VAPT," pp. 337–366, 2022.

[21] E. A. Altulaihan, A. Alismail, and M. Frikha, "A survey on Web application penetration testing," *Electronics*, vol. 12, p. 1229, 2023.

[22] Y. Khera, D. Kumar, Sujay, and N. Garg, "Analysis and impact of vulnerability assessment and penetration testing," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019.

[23] J. N. Goel and B. Mehtre, "Vulnerability assessment and penetration testing as a cyber defence technology," *Procedia Computer Science*, vol. 57, pp. 710–715, 2015.

[24] X. Qiu, S. Wang, Q. Jia, C. Xia, and Q. Xia, "An automated method of penetration testing," in *IEEE Xplore*, pp. 211–216, 2014.

[25] F. Abu-Dabaseh and E. Alshammari, "Automated penetration testing: An overview," *Computer Science and Information Technology*, 2018.

[26] OWASP, "How to use the OWASP Top 10 as a standard -OWASP Top 10:2021," 2021. [Online].
Available: https://owasp.org/ www-project-top-ten/

[27] OWASP, "ASVS Application Security Verification Standard," [Online]. Available: https://owasp.org/ www-project-application-security-verification-standard/

[28] O. Valea and C. Oprisa, "Towards pentesting automation using the Metasploit framework," in *IEEE Xplore*, pp. 171–178, 2020.

[29] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "Towards automated penetration testing for cloud applications," in *IEEE Xplore*, pp. 24–29, 2018.

[30] O. K. Akram, D. J. G. Franco, N. F., A. Mohammed Jamil, and S. Ismail, "How to Guide Your Research Using ONDAS Framework," Beja, Portugal, 2018.

[31] D. J. Franco, "Privacy Optimization and Intrusion Detection in MODBUS/TCP Network-Based SCADA in Water Distribution Systems," Ph.D. dissertation, Universiti Putra Malaysia, 2021.

[32] O. K. Akram, D. J. Franco, and A. Lee, "Undergraduate and Graduate Students' Challenges: A Qualitative Study with ONDAS Framework Across Multiple Disciplines and Innovative Research Methodologies," *The Qualitative Report*, vol. 28, no. 10, pp. 2887–2915, 2023.

[33] A. Anwar, "What is Average Precision in Object Detection and Localization Algorithms and how to calculate it," 2022.

[34] Python Software Foundation, "About Python," 2023. [Online]. Available: https://www.python.org/about/

[35] Offensive Security, "Get Kali — Kali Linux," 2023. [Online]. Available: https://www.kali.org/get-kali/#kali-platforms

[36] Microsoft, "Windows 10 Professional," 2023. [Online]. Available: https: //www.microsoft.com/en-us/software-download/windows10