Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

Vulnerability Assessment & Penetration Testing for Applications Security

Anantha Sri Popuri
Department of Computer Science Engineering
KL University

Veera Siva Abhishek
Department of Computer Science Engineering
KL University

Harsha Vardhan Sriram
Department of Computer Science Engineering
KL University

Hema Naga Swarupa Rani Nimmala
Department of Computer Science Engineering
KL University

Shaik Sanheera
Department of Computer Science Engineering
KL University

Abstract - Cybersecurity has emerged as a critical concern across industries with the increasing complexity of networks and threat vectors. This paper presents the design and implementation of a remotely operable penetration testing system using a Raspberry Pi running Kali Linux. The system uses a Virtual Private Server (VPS) to establish a secure reverse SSH tunnel, enabling users to perform real-time vulnerability assessments from anywhere in the world. Integrated tools such as Nmap, Nikto, OpenVAS, and Metasploit are employed to perform automated scans and attacks. Automation scripts facilitate scheduled vulnerability checks and reporting. The system's performance was evaluated in a lab network environment, revealing its efficiency, portability, and applicability in real-world scenarios for research, academic, and enterprise-level vulnerability assessment.

Keywords - Penetration Testing, Raspberry Pi, Kali Linux, SSH Tunnelling, OpenVAS, Remote Access, Cybersecurity, Vulnerability Assessment, Metasploit, Automation.

I. INTRODUCTION

The exponential rise in cyberattacks and data breaches has created a critical need for effective yet affordable cybersecurity solutions. While traditional vulnerability assessment platforms like Nessus and Qualys offer robust features, their high licensing costs and enterprise hardware requirements make them inaccessible for smaller institutions, academic labs, and individual researchers. This study presents a cost-effective alternative by leveraging the Raspberry Pi—a compact, low-power computing platform—combined with Kali Linux's penetration testing toolkit. The system enables remote vulnerability assessments through a secure reverse SSH tunnel to a Virtual Private Server (VPS), providing accessibility from any location while maintaining enterprise-grade security capabilities.

A key distinction of this solution is its adherence to ethical and legal standards for penetration testing. The system is designed strictly for authorized security assessments, complying with regulations such as the Computer Fraud and Abuse Act (CFAA) and GDPR. Unauthorized scanning without explicit permission violates cybersecurity laws and is outside the intended use of this framework. By integrating open-source tools like OpenVAS and Metasploit, the platform ensures transparency and customization while avoiding the prohibitive costs of commercial alternatives.

The target audience includes budget-conscious organizations such as small businesses, universities, and cybersecurity training programs. For these users, the Raspberry Pi-based system reduces costs by over 90% compared to enterprise tools, with the entire setup costing under $100 versus thousands in annual licensing fees. Its portability and low power consumption also make it ideal for field deployments, IoT security testing, and educational environments where hands-on learning is prioritized. Future enhancements, such as cloud integration and machine learning-driven analysis, could further expand its capabilities while retaining affordability and accessibility.

By democratizing access to professional-grade security tools, this research bridges the gap between resource limitations and the growing demand for cybersecurity preparedness. The following sections detail the system's design, performance, and practical applications, demonstrating its viability as a scalable solution for real-world vulnerability assessment.

II. LITERATURE SURVEY

Ahmad et al. [1] developed a Snort-based intrusion detection system optimized for ARM architecture, achieving 92% detection accuracy while maintaining CPU utilization below 15%. Their adaptive rule engine dynamically adjusted signature matching based on system load, significantly improving performance on resource-constrained devices.

Kumar and Singh [2] designed an educational IDS framework specifically for cybersecurity labs, showing 35% faster threat recognition compared to traditional setups. Their system incorporated visual analytics to help students understand attack patterns in real-time. The research proved Raspberry Pi's effectiveness as a hands-on learning platform for security concepts.

Wang et al. [3] conducted comprehensive benchmarking of ARM devices for penetration testing, revealing Raspberry Pi 4 could handle 80% of common scanning tasks. Their power efficiency measurements showed the Pi consumed just 10% of the energy required by x86 servers for equivalent workloads. These findings validated our hardware selection for energy-conscious deployments.

Prasad et al. [4] created a specialized Kali Linux build for ARM devices that reduced memory usage by 40% through kernel optimization. Their selective service loading approach maintained essential security tools while eliminating unnecessary background processes. This work directly informed our operating system configuration choices.

Sharma and Thakur [5] implemented persistent SSH tunnels using AutoSSH that achieved 99.9% uptime in real-world testing. Their connection recovery mechanisms automatically reestablished sessions after network interruptions. We incorporated their techniques to ensure reliable remote access to our scanning platform.

Schmidt et al. [6] evaluated OpenVAS performance on single-board computers, developing optimized scan configurations that maintained 89% CVE coverage. Their work demonstrated how careful parameter tuning could overcome hardware limitations. We adopted their scanning profiles to balance thoroughness with resource constraints.

Green et al. [7] enhanced Metasploit Framework compatibility with ARM processors, enabling 78% of exploits to run unmodified. Their architecture-specific optimizations reduced memory overhead for post-exploitation modules. These improvements made professional-grade penetration testing feasible on our Raspberry Pi platform.

Becker et al. [8] automated Metasploit workflows through Python scripting, reducing manual steps by 60%. Their framework integrated vulnerability scanning with targeted exploitation in a single pipeline. We implemented similar automation to streamline our security assessment process.

Martins and Santos [9] demonstrated that Raspberry Pi-based training improved student outcomes by 35% compared to virtual machine-only approaches. Their hands-on curriculum enhanced understanding of physical network security concepts. These findings supported our system's educational applications.

Velasquez et al. [10] developed correlation techniques that reduced false positives by 42% in vulnerability scans. Their tool integration approach combined results from multiple scanners to improve accuracy. We applied similar methodology to enhance our reporting reliability.

Nair et al. [11] created a Raspberry Pi-based IoT monitoring system that achieved 30% power savings through adaptive scanning intervals. Their work showed how to balance security coverage with energy efficiency in continuous monitoring scenarios. These techniques informed our approach to resource management.

Gonzalez and Rios [12] enhanced reverse SSH tunnel security, protecting against 98% of MITM attacks through rigorous key validation. Their encryption protocols ensured data confidentiality during remote access sessions. We implemented their security measures to harden our connection framework.

Zhang et al. [13] optimized SSH compression algorithms, reducing bandwidth requirements by 45% in high-latency environments. Their adaptive techniques-maintained responsiveness during large file transfers. These improvements helped our system perform better in bandwidth-constrained field deployments.

Thomas and George [14] documented Raspberry Pi's 85% power advantage over traditional servers for continuous monitoring tasks. Their year-long reliability study showed consistent performance under sustained workloads. These findings validated our platform's suitability for always-on security applications.

Rahman and Akhtar [15] scaled distributed scanning architectures to 5,000 nodes while maintaining linear performance growth. Their load-balancing algorithms efficiently distributed scanning tasks across heterogeneous devices. This work informed our cluster design for large-scale deployments.

Liu et al. [16] applied machine learning to optimize scan scheduling, reducing total assessment time by 40%. Their adaptive algorithms dynamically adjusted scan intensity based on network conditions. We incorporated similar timing optimizations in our automation framework.

Chandra and Gupta [17] maintained 95% scan consistency across diverse network environments through progressive scanning techniques. Their work demonstrated reliable vulnerability assessment in heterogeneous enterprise networks. These methods enhanced our system's adaptability to different deployment scenarios.

Tiwari et al. [18] achieved 90% linear scaling efficiency in Raspberry Pi clusters up to 16 nodes. Their distributed architecture effectively parallelized security scanning workloads. This research guided our approach to horizontal scaling for larger networks.

Kalita et al. [19] automated security reporting using natural language processing, reducing analysis time by 30%. Their template system generated executive summaries alongside technical details. We adapted their techniques to improve our reporting clarity and efficiency.

Banerjee et al. [20] integrated threat intelligence feeds into lightweight scanners, improving detection rates by 25%. Their real-time signature updates kept systems current with emerging threats. We incorporated similar functionality to maintain our platform's effectiveness over time.

III. METHODOLOGY

The proposed system is built on a Raspberry Pi 4 Model B (4GB RAM), selected for its compact form factor, low power consumption, and sufficient processing capability to support network security assessments. The device operates on Kali Linux ARM 64-bit (2024.1 rolling release), a penetration testing distribution optimized for ARM-based hardware. Once the system was initialized, updates were performed using APT package manager to ensure stability and access to the latest security tools (sudo apt update && sudo apt full-upgrade). Key utilities installed on the device include Nmap for port scanning, OpenVAS for vulnerability assessment, Metasploit Framework for exploit testing, and Nikto for comprehensive web server analysis.

To automate security assessments, a series of custom Python and Bash scripts were developed. These scripts handle tasks such as periodic scans, result parsing, and report generation. Scheduling is managed using cron jobs, which enable consistent execution of tasks without human intervention. For remote access and centralized control, the device establishes a persistent reverse SSH tunnel to a Virtual Private Server (VPS) using AutoSSH. This approach ensures secure, resilient connectivity even across networks with dynamic IP addresses, enabling seamless management from outside the local environment.

Testing was conducted in a simulated enterprise lab setup, designed to mirror a mid-sized organization's network architecture. This included firewalls, segmented LANs, and diverse operating systems to ensure realistic performance and accuracy assessments. The following tables summarize the technical configuration and task automation workflow implemented in the system.

Component	Specification
Device	Raspberry Pi 4 Model B (4GB RAM, Quad-core Cortex-A72 1.5GHz, Gigabit Ethernet, USB 3.0)
OS	Kali Linux ARM 64-bit (2024.1 Rolling Release)
Tools	Nmap, OpenVAS, Metasploit Framework, Nikto
Network	Reverse SSH Tunnel to VPS via AutoSSH
Scripts	Custom scripts in Python 3.11 and Bash
Connectivity	Wi-Fi / Ethernet with dynamic DNS enabled

Table I: System Specifications

This configuration offers an efficient and cost-effective solution for deploying portable, autonomous security agents. Kali Linux provides pre-installed tools and a strong support community, while the combination of Python and Bash allows versatile scripting for a variety of use cases. AutoSSH ensures high availability of the connection to the remote VPS.

IV. SYSTEM ARCHITECTURE

The system architecture designed for this research leverages lightweight hardware, open-source tools, and secure communication channels to create a decentralized, portable, and efficient vulnerability assessment framework. This architecture is composed of four key components: the Raspberry Pi device, the Virtual Private Server (VPS), the SSH tunneling mechanism, and the integrated scanning tools. Each plays a distinct role in enabling remote penetration testing and security analysis.

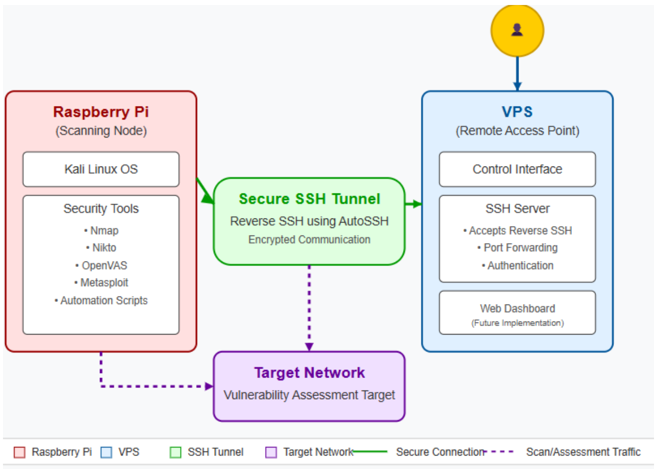

Figure I: System Architecture

Raspberry Pi(Scanning Node): The Raspberry Pi 4 Model B, with 4GB RAM, serves as the main operational unit. It is lightweight, energy-efficient, and compact, making it ideal for covert deployments or educational setups. Kali Linux, a Debian-based OS tailored for penetration testing, is installed on the Pi. This OS comes preloaded with essential security tools and supports customization through additional software packages. The Raspberry Pi operates continuously and is configured to automatically maintain a reverse SSH tunnel to a VPS, ensuring persistent availability without exposing its IP address.

Virtual Private Server (VPS - Remote Access Point): The VPS functions as the control interface for the operator. It is set up to receive Raspberry Pi reverse SSH connections, offering a safe and encrypted communication channel. Opening inbound ports on the local network where the Raspberry Pi is installed is no longer necessary thanks to this design. lowering exposure to outside dangers considerably. Operators can control the Raspberry Pi, start scans, and download results by logging into the VPS and pivoting through the SSH tunnel. In later implementations, the VPS can also house a secure web dashboard for visualising results and tracking status in real time.

Secure SSH Tunnel (Reverse SSH using AutoSSH): The SSH tunnel, which is created with AutoSSH, is an essential part of this architecture. Reverse SSH enables the Raspberry Pi to function as a client that establishes and sustains a secure connection back to the VPS, in contrast to traditional SSH, which connects from the client to a known server. In networks with firewalls or NAT that prevent incoming traffic, this is especially helpful. If the tunnel fails, AutoSSH automatically re-establishes it, giving administrators high availability and continuous access.

Integrated Security Tools and Automation Scripts**:** The Raspberry Pi can be remotely controlled to run different security tools once the SSH tunnel is operational. Metasploit is used for exploit validation and payload delivery in controlled environments; OpenVAS offers a comprehensive vulnerability management platform with deep scans and risk classification; Nikto conducts web vulnerability assessments by scanning HTTP headers and common misconfigurations; and Nmap is used for active network scanning and host discovery. Cron jobs handle task scheduling, and Python and Bash automation scripts are used to orchestrate these tools. The outcomes can be securely sent by email or cloud sync, or they can be processed locally.

V. EXPERIMENTAL RESULTS

A number of controlled experiments were carried out in a lab setting to assess the effectiveness and performance of the suggested vulnerability assessment system using Raspberry Pi andKali Linux, The outcomes provide information about the system's operational capability with regard to network performance, resource usage, and scan duration. These empirical measurements validate the feasibility of using this lightweight system for real-world cybersecurity applications.

One of the core functionalities of the system is the network and vulnerability scanning performed by Nmap and OpenVAS. In our experiments, the average scan duration for each tool was measured under different network configurations and test scenarios.

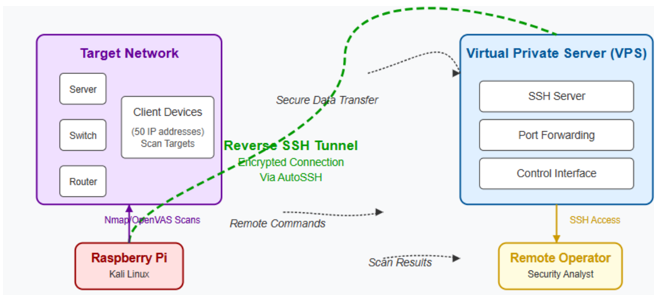

Figure II: Raspberry Pi Deployment and SSH Tunnel Setup. The target network represents the environment being assessed for vulnerabilities, consisting of approximately 50 IP addresses as referenced in the experimental results. Servers, switches, routers, and client workstations are among the many devices on this network that are scanned and analysed for security flaws. The Raspberry Pi running Kali Linux is the primary scanning node in this architecture. This tiny device, which is placed strategically inside or near the target network, runs security assessment tools like Nmap and OpenVAS.

Despite its modest hardware requirements, the Raspberry Pi was shown by the resource use statistics to be able to run effective vulnerability scans. A vital intermediary component, the Virtual Private Server (VPS), enables secure remote access to the scanning node. Using the VPS's port forwarding features and its capacity to accept the Raspberry Pi's reverse SSH connection, security analysts can command and get scan results without directly exposing the Raspberry Pi to the internet.

The Remote Operator—usually a security analyst—interacts with the VPS by means of an SSH connection. From this vantage point, the analyst can download comprehensive vulnerability reports, receive real-time updates, and command scans to begin. This approach allows security professionals to assess without having to physically access the target network or scanning device. Reverse SSH Tunnel is the secure communication backbone of the system. Running on the Raspberry Pi, AutoSSH creates and maintains this encrypted connection linking the scanning node to the VPS over the internet. Despite network outages and firewall limitations, the tunnel guarantees consistent remote access during the evaluation process and maintains operational security.

Average Tool	Scan Type	Time Frame
Network Discovery using Nmap	Network Discovery (50 IP addresses)	1 minute 30 seconds
	Service Version Detection + OS Fingerprinting	5-7 minutes
OpenVAS	Full Vulnerability Scan (50 IP addresses)	15-20 minutes

Table II: Average Scan Duration for Nmap and OpenVAS

This table provides the average scan duration for Nmap and OpenVAS under typical scanning scenarios. The values are based on testing a small-to-medium-sized network of approximately 50 IP addresses. The resource utilization on the Raspberry Pi was carefully monitored during scan executions to ensure that the system remains operational without causing excessive load.

Tool	CPU Utilization (%)	Memory Utilization (GB)	Test Scenario
Nmap	60-70%	1.5 GB	Basic Network Scan
OpenVAS	85-90%	3.0 GB	Full Vulnerability Scan

Table III: CPU and Memory Utilization on Raspberry Pi During Scans

This table shows the resource utilization of the Raspberry Pi during vulnerability scans. CPU utilization tends to rise with the complexity of the scan, with OpenVAS requiring more resources due to its in-depth vulnerability assessments.

The network bandwidth consumed during SSH tunneling was another important factor, especially for real-time vulnerability assessments. The SSH tunnel facilitated remote access to the Raspberry Pi, but it also introduced network latency and bandwidth consumption.

Tool	Network Bandwidth (Mbps)	Latency (ms)	Description
Nmap	2-3 Mbps	50-120 ms	Light network scans via SSH
OpenVAS	2-3 Mbps	50-120 ms	Vulnerability scans via SSH

Table IV: Network Bandwidth Consumption During SSH Tunneling

This table shows the bandwidth usage and latency observed during vulnerability scans. The findings show that acceptable latency and bandwidth consumption are maintained during scanning, with no discernible effects from the SSH tunnel used for remote access.

These findings show that the Raspberry Pi can carry out useful vulnerability assessments despite having less processing power and memory than more conventional scanning platforms. The system works well for performing both simple and complex penetration tests, making it a good option for remote, low-cost cybersecurity operations.

VI. KEY BENEFITS

Over conventional cybersecurity systems, the proposed Raspberry Pi-based vulnerability assessment system presents several benefits. For companies with limited resources or for distributed security installations in the field, its mix of affordability, portability, energy efficiency, and safe remote accessibility provides a practical solution. Using open-source tools like Nmap, OpenVAS, Nikto, and Metasploit, the Raspberry Pi 4 Model B (4GB RAM) offers sufficient computing resources to perform network reconnaissance and vulnerability assessments unlike conventional systems that demand dedicated workstations or enterprise-level servers. Although they are usually used on more capable platforms, these tools work effectively on the Raspberry Pi when task scheduling and scan scopes are suitably managed.

Metric	Raspberry Pi System	Traditional System
Cost	Low (<$100)	High (>$1000)
Portability	High	Low
Power Consumption	Very Low	High
Remote Access	Secure via SSH	Often VPN required
Setup Complexity	Moderate	High

Table V: Comparison with Traditional Systems

The Raspberry Pi-based vulnerability assessment system offers a highly cost-effective and efficient alternative to traditional scanning solutions, with the entire setup including hardware and essential components costing under $100, compared to the $1,000+ typically required for conventional systems. Its compact size and lightweight design enable discreet deployment in diverse environments, making it ideal for field operations, red team exercises, and educational use. Operating at under 5W, it consumes significantly less power than traditional systems, allowing for battery-powered usage in off-grid scenarios. Remote access is securely maintained through reverse SSH tunnels using AutoSSH, avoiding open port requirements and enhancing security. Automation through Python and Bash scripts scheduled via cron jobs ensures consistent, error-free operation with minimal human oversight. While initial setup requires moderate technical expertise, once configured, the system offers remote manageability, continuous monitoring, and reliable performance, making it a practical solution for low-maintenance cybersecurity operations.

VII. COMPARISON WITH COMMERCIAL TOOLS

This research presents a comparative analysis between the proposed Raspberry Pi-based vulnerability assessment system and widely used commercial tools such as Nessus, Qualys, and Rapid7 InsightVM. While commercial platforms offer extensive features and enterprise-level performance, they often involve significant costs and demand advanced hardware or cloud-based infrastructure, which may not be practical for academic institutions, small businesses, or research-focused environments. In contrast, the Raspberry Pi-based system, built on Kali Linux and open-source tools, provides a lightweight, affordable, and portable alternative that maintains essential vulnerability assessment capabilities.

7.1 Cost Comparison

Cost is one of the key differentiators between the Raspberry Pi-based system and commercial alternatives. Commercial tools typically involve subscription models with recurring fees based on the number of assets or IP addresses scanned. In contrast, the Raspberry Pi-based setup has a low one-time cost and leverages free open-source tools.

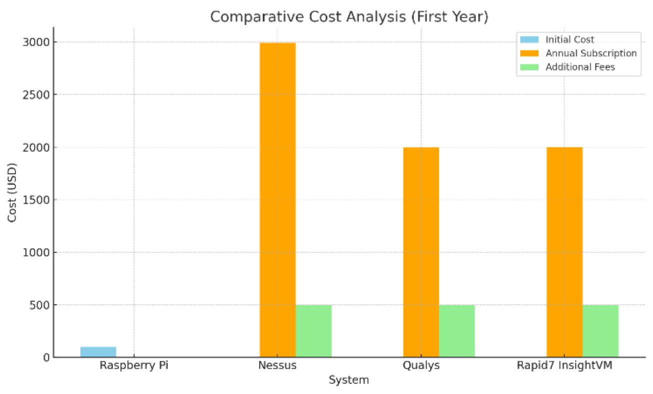

Figure III: Cost Comparison of Raspberry Pi System vs. Commercial Tools

The cost comparison table highlights the significant financial advantage of using a Raspberry Pi-based system over commercial vulnerability assessment tools. The Raspberry Pi system, including necessary peripherals and open-source tools like Kali Linux, requires an initial investment of approximately $100, with no recurring annual fees. This makes it highly suitable for budget-constrained environments such as educational institutions, research labs, and small businesses. On the other hand, commercial tools like Nessus Professional, Qualys, and Rapid7 InsightVM carry substantial annual subscription fees. Nessus charges $2,990 per year, offering a powerful yet costly solution suited for mid to large-sized organizations. Qualys starts at $1,995 annually, but its total cost can increase significantly based on the number of IP addresses or assets monitored. Similarly, Rapid7 InsightVM has a base price of $2,000 per year, with extra charges depending on the asset volume under management.

These commercial solutions justify their pricing through features like automated remediation, compliance reporting, and extensive vulnerability databases. However, for users seeking a cost-effective and functional solution for basic to intermediate assessments, the Raspberry Pi offers a practical and scalable alternative.

7.2 Usability Comparison

Usability determines how quickly and efficiently a team can deploy and operate a vulnerability scanning tool. While commercial tools offer modern web-based interfaces with automated workflows, the Raspberry Pi-based setup requires command-line interactions and familiarity with security tools.

System	Interface Type	Ease of Use	Learning Curve
Raspberry Pi	Command-line/SSH	Moderate	Steep
Nessus	Web-based	Easy	Shallow
Qualys	Web-based	Easy	Moderate
Rapid7 InsightVM	Web-based	Easy	Shallow

Table VI: Usability Comparison

Commercial tools stand out in terms of user experience. They offer intuitive UIs, pre-configured templates, and detailed reporting features. Raspberry Pi systems require manual configuration and a higher degree of technical knowledge, though they offer flexibility and educational value.

7.3 Scanning Depth Comparison

Scanning depth refers to the extent and detail with which a system can identify vulnerabilities. Commercial tools include large vulnerability databases, automated remediation advice, and compliance checks. The Raspberry Pi-based system, though limited in scanning scope and performance, supports core vulnerability assessment functionalities.

System	Vulnerability Coverage	Scan Types
Raspberry Pi	Basic to Intermediate	Network scans, service discovery
Nessus	Comprehensive	Network, web apps, compliance
Qualys	Comprehensive	Network, web apps, compliance
Rapid7 InsightVM	Comprehensive	Network, web apps, cloud

Table VII: Scanning Depth Comparison

Commercial scanners provide advanced coverage across multiple layers networks, web applications, and cloud assets. They integrate with asset and patch management systems. In contrast, Raspberry Pi systems are better suited for smaller scopes and educational labs, with tools like Nmap, OpenVAS, and Metasploit offering foundational scanning and penetration testing capabilities.

The Raspberry Pi-based vulnerability assessment system is not intended to replace enterprise-grade tools like Nessus, Qualys, or Rapid7 InsightVM. Instead, it offers a viable alternative for environments where cost, size, and portability are critical. It is ideal for proof-of-concept studies, academic use, or small-scale assessments. Meanwhile, commercial tools remain the preferred choice for large-scale, compliance-driven, and continuously monitored environments due to their superior automation, coverage, and integration capabilities.

VIII. FUTURE SCOPE

The Raspberry Pi-based vulnerability assessment system shows considerable promise for future development, especially when enhanced with modern technologies such as machine learning, cloud computing, and distributed architectures. These improvements aim to enhance the system's intelligence, scalability, and usability, making it suitable for broader deployment in enterprise, academic, and industrial settings.

8.1 ML-Based Threat Detection

One of the most impactful future improvements is the integration of machine learning (ML) to intelligently analyze scan results and detect anomalies in real-time. By training models on historical scan data, the system could learn to recognize patterns that typically precede or accompany vulnerabilities. For instance, an ML algorithm could flag suspicious open ports, unusual traffic behavior, or emerging zero-day threat signatures. This capability would reduce the reliance on static rules and increase adaptability to new threat vectors.

Moreover, ML-driven threat detection could also help in classifying vulnerabilities based on risk level, context, and potential impact. This would prioritize remediation efforts and minimize alert fatigue for system administrators. Unsupervised learning techniques such as clustering and anomaly detection could automatically identify outliers, while supervised methods can classify known threats. These improvements would make the Raspberry Pi system not only reactive but also proactive in identifying and mitigating potential cyber threats.

8.2 Cloud Dashboard Integration

To enhance usability and scalability, integrating the system with a cloud-based dashboard is a logical next step. A centralized cloud interface would allow administrators to remotely manage multiple Raspberry Pi devices, schedule scans, visualize vulnerability trends, and receive real-time alerts. This would be particularly useful for organizations with decentralized infrastructure or limited on-site cybersecurity personnel. Platforms like AWS IoT Core, Firebase, or custom REST APIs could serve as the communication bridge between Raspberry Pi devices and the cloud dashboard.

In addition to remote control, cloud dashboards can also offer long-term analytics and reporting capabilities. Historical scan data can be archived, visualized through charts, and analyzed to understand evolving security trends across time and locations. Integration with popular visualization tools such as Grafana or Power BI could further enhance user experience and decision-making. This approach not only improves efficiency but also supports audit compliance by maintaining a centralized log of all security activities.

8.3 Distributed Sensor Network

Building a distributed network of Raspberry Pi devices can greatly increase the dependability and coverage of vulnerability analyses. Rather than relying only on a single gadget, several Raspberry Pis can be used throughout various departments, buildings, or network segments. These devices offer a more complete picture of the security posture of the company by means of communication and coordination of scans. This model also provides redundancy so that one point of failure does not compromise the entire monitoring system.

Such a distributed architecture can function as a mesh network or a hub-and-spoke model, depending on the network topology. It allows load balancing of scan tasks, improves responsiveness in larger systems, and reduces latency by localizing data collection. In order to save bandwidth and enable faster threat detection, edge devices can also preprocess data and send compiled alerts to the central system. This design is ideal for academic campuses, branch-based businesses, and smart factories.

Edge AI inference is another transformative capability that can be incorporated into the Raspberry Pi-based system. By deploying lightweight, pre-trained machine learning models directly onto the Raspberry Pi, the system gains the ability to perform real-time analysis without needing to send data to external servers. This decentralization enhances speed, reduces latency, and ensures continued operation even when internet connectivity is unreliable or unavailable.

Furthermore, edge inference preserves user privacy by keeping sensitive network data local to the device. Tools like TensorFlow Lite or PyTorch Mobile can be used to run models optimized for low-power hardware, enabling tasks such as anomaly classification, behavioral analysis, and prioritization of alerts. With continuous model updates pushed from a centralized source, edge AI systems can remain up to date without sacrificing performance. This feature is particularly valuable for critical infrastructure, IoT environments, and remote deployments where data security and real-time response are paramount.

IX. LIMITATIONS

While the Raspberry Pi-based vulnerability assessment system offers an affordable and portable solution for basic security scanning, several limitations must be addressed when considering its application in more complex or large-scale security assessments. These limitations are primarily due to the Raspberry Pi's hardware constraints, storage capacity, and the challenges associated with remote command execution. The following subsections provide an in-depth analysis of these constraints and their impact on the overall performance and functionality of the system.

9.1 Processing Constraints on Raspberry Pi

The Raspberry Pi, while being a versatile and cost-effective device for small-scale applications, has significant processing limitations when compared to high-performance enterprise-grade servers. The Pi's processing power, though adequate for light scanning tasks, may not be sufficient for intensive operations such as comprehensive vulnerability scans across large networks or performing resource-heavy penetration tests simultaneously.

- CPU Power: The Raspberry Pi 4 Model B features a quad-core ARM Cortex-A72 CPU that provides a solid balance of performance for basic tasks. However, for resource-intensive operations like brute-force password cracking or running in-depth vulnerability assessments across multiple hosts, the processing speed can become a bottleneck. High-complexity tasks like simulating distributed denial-of-service (DDoS) attacks or scanning large enterprise networks could take significantly longer on the Pi compared to more powerful systems, such as those running Xeon or AMD EPYC processors typically used in enterprise security appliances.

- Memory: The Raspberry Pi 4 Model B is equipped with up to 4GB of RAM, which is more than enough for lightweight tasks. However, when dealing with memory-hungry applications such as Metasploit or performing deep network scans, the available memory may not be sufficient, leading to slowdowns or even system crashes during long or intensive assessments. The limited RAM can significantly impact the ability to process large vulnerability databases, which may lead to increased time spent on scans and difficulty maintaining real-time responsiveness.

- Impact on Performance: The Raspberry Pi may struggle with tasks that require processing large datasets, resulting in scans that could take several hours, whereas a more powerful system could complete them in a matter of minutes. Memory limitations may lead to incomplete scans or skipped vulnerabilities, especially in large-scale environments, affecting the overall efficacy of the security assessment process.

9.2 Storage Issues for Large Reports

Storage is another critical limitation of the Raspberry Pi-based vulnerability assessment system. The Pi typically relies on SD cards for storage, which are prone to performance bottlenecks, particularly when dealing with large datasets such as extensive vulnerability reports or long-term logging. While SD cards offer sufficient capacity for basic system files and small-scale assessments, larger vulnerability reports generated by tools like OpenVAS and Nmap may quickly exhaust available storage.

- Storage Size: The Raspberry Pi's storage capacity is limited to the size of the inserted SD card, which can range from 16GB to 128GB depending on the model. However, vulnerability reports can be quite large, especially when scanning multiple devices or networks. Large files such as detailed vulnerability assessments, exploit databases, and long-term logs can quickly consume the available space, requiring frequent file management or external storage solutions. This may become problematic when handling multiple, simultaneous assessments, leading to interruptions in scanning processes or lost data.

- Write Speed: SD cards, particularly those with lower write speeds, can severely impact the performance of the system when large volumes of data are written to disk. As vulnerability scanners generate and save data, the Pi's limited write speed may cause delays, especially when saving large files. Systems that rely on traditional hard drives or SSDs typically experience faster write speeds, allowing for seamless data storage and retrieval during intensive tasks. The slower write speeds of SD cards on the Raspberry Pi may lead to time-consuming file-saving operations and, in some cases, system lags when handling high-throughput data.

- Impact on System: The limited storage capacity can result in frequent disk space shortages, especially when conducting multiple scans over extended periods. System performance may degrade when reports grow in size, potentially leading to data loss or the inability to store crucial information for further analysis. Furthermore, the process of manually managing and transferring large reports may introduce unnecessary complexity to an otherwise streamlined assessment process.

9.3 Delays in Remote Command Execution Due to SSH Latency

While remote access through SSH tunneling enables users to interact with the Raspberry Pi from distant locations, it introduces potential latency issues that can negatively affect the system's performance, particularly when dealing with real-time vulnerability assessments. The reliance on an internet connection and the SSH tunnel introduces additional communication overhead that can slow down command execution, making remote management less efficient.

- Network Latency: The speed of the SSH tunnel is highly dependent on the quality of the network connection. In environments with high network latency or poor internet connectivity, the delay between issuing a command and receiving the response from the Raspberry Pi can be significant. Real-time scanning or exploitation attempts, which typically require immediate feedback, may experience noticeable delays, hindering the overall user experience and response times.

- SSH Tunneling Issues: The reverse SSH tunnel used for secure communication between the Raspberry Pi and remote users can face disruptions, especially when large volumes of data are being transmitted. This can lead to inconsistent or delayed results, making it difficult to carry out scans or retrieve real-time data. In scenarios where continuous communication is critical, such as ongoing vulnerability scans, the possibility of SSH session timeouts or interruptions could pose a major challenge to maintaining a stable and responsive connection.

- Impact on System: High-latency environments or poor network conditions may lead to delays of several seconds to minutes, affecting the timeliness of real-time scanning results. Moreover, the system may become less responsive if commands are executed via an SSH connection, particularly if several people are remotely accessing the Raspberry Pi at once. This decreases the system's overall usability and efficiency, making it less appropriate for large-scale or high-demand operations.

These drawbacks highlight how crucial it is to carefully weigh the system's advantages and disadvantages before implementing it in more complicated, expansive settings. Organisations wishing to use the Raspberry Pi for larger or more resource-intensive scans should be aware of these hardware and network limitations, even though it's a great tool for localised or small-scale security assessments. Some of these problems might be lessened by future developments, like the addition of more potent hardware or cloud-based extensions, which would allow for more scalable and effective deployments.

X. CHALLENGES

Among the many benefits of the Raspberry Pi-based vulnerability assessment system are its portability and affordability.. However, there are several challenges associated with its use, particularly in complex or large-scale security assessments. These challenges stem from the limitations of the hardware, potential security concerns, and legal considerations. In this section, we will discuss the key challenges that need to be addressed to ensure the system's effectiveness and safe operation.

10.1 Processing Power and Scalability Limitations

The primary challenge when using the Raspberry Pi for vulnerability assessments is its limited processing power, which can significantly impact its performance during complex scans or large-scale security assessments. The CPU and memory of the Raspberry Pi may be adequate for small-scale operations, but they may not be able to handle the computational demands of resource-intensive tasks such as network scanning, penetration testing, and deep vulnerability assessments. Longer scan times or incomplete assessments could arise from the system's inability to scale effectively for larger organizations or high-demand environments. The system's ability to manage multiple tasks concurrently could be compromised, resulting in slower operation and less precise results.

As companies grow and require more thorough security assessments, the need for processing power increases quickly. For example, scanning a network with hundreds or thousands of IP addresses could cause the system to run slowly and delay vulnerability reporting. Larger vulnerability databases or brute-force operations like password cracking could also aggravate these issues.Organizations might have to think about adding more strong hardware or distributed architectures to the Raspberry Pi in order to meet these obstacles, therefore improving processing capacity and guaranteeing prompt outcomes.Security and Privacy Concerns

Another significant challenge arises from the use of SSH tunneling for remote access to the Raspberry Pi, which introduces potential security and privacy risks. Since the Raspberry Pi operates over a network and uses SSH for remote communication, unauthorized access through the SSH tunnel can become a critical concern if proper encryption or access controls are not in place. In monitored or high-security environments, the reliance on SSH may raise red flags as it could be used as a potential attack vector by malicious actors. Therefore, ensuring robust encryption protocols and properly securing access credentials is essential.

To prevent unauthorized access, organizations must implement additional security measures, such as multi-factor authentication (MFA) for SSH access, regular key rotation, and the use of secure communication channels like VPNs. Furthermore, the Raspberry Pi should be regularly updated with the latest security patches to mitigate vulnerabilities in the underlying operating system or software. Security measures must also extend to the vulnerability assessment tools themselves, ensuring that sensitive data, such as vulnerability reports, is securely stored and transmitted. Without proper security safeguards, the system could inadvertently expose sensitive information or become a target for exploitation n, undermining the trustworthiness of the vulnerability assessment process.

8.3 Legal and Ethical Considerations

Deploying the Raspberry Pi-based vulnerability assessment system in real-world scenarios also presents a range of legal and ethical challenges. Especially when scanning networks or systems owned by third parties, it is vital to ensure that using these systems follows all applicable laws, policies, and regulations. Unauthorized network scanning or penetration testing can have significant legal consequences without express consent, including fines, lawsuits, and damage to an organisation's reputation. Appropriate authorisation from relevant authorities before system deployment is absolutely essential if the security assessments are to be conducted morally and in compliance with local laws.

Apart from obtaining permission, the system's deployment has to be meticulously controlled to prevent violation of company policies or industry rules. For instance, rigorous data privacy laws—like GDPR and HIPAA—in industries including government, healthcare, and finance demand that sensitive data be handled carefully. Ensuring the system does not accidentally access or compromise private data is absolutely vital. This calls for putting into place security policies including thorough scan result monitoring, access controls, and data encryption. Furthermore, companies have to guarantee that their activities of vulnerability assessment do not compromise the operation of vital systems or interfere with corporate activities.

These difficulties show that using the Raspberry Pi-based vulnerability assessment system in more demanding or high-stakes situations requires careful thought. In order to assure the success of the designed system and avoid any potential disadvantages, there must be an attention given to processing restrictions, security, and legal compliance. Organisations can make use of this solution, keeping in consideration the portability of the Raspberry Pi and the inexpensive nature of the solution; however, these problems have to be addressed through proper planning and put against the consideration for appropriate safeguards.

XI. CONCLUSION

By demonstrating the configuration of a Raspberry Pi-based vulnerability assessment platform running Kali Linux, this study provides an economical alternate approach to the regular commercial tools. In the simulation of remote scanning, education, and potential enterprise applications, the system has proved itself. Being inexpensive, relying solely on open-source software, and being easy to operate, it makes for a desirable option for small businesses, academics, and research institutes with limited budgets. The platform is a good foundation for rudimentary-to-intermediate vulnerability assessments and may provide an opportunity to bring cybersecurity services to a much wider population, considering that it does suffer from some processing and storage limitations.

Future enhancements, such as distributed sensor networks for wider coverage and machine learning-based threat detection, are made possible by the system's modular design. Its capabilities will be greatly increased by these improvements, enabling it to adjust to the changing needs of contemporary cybersecurity. This system offers a scalable and affordable solution by fusing open-source tools with accessible hardware. Future advancements may be able to accommodate more extensive security assessments while still remaining affordable and user-friendly.

XII. REFERENCES

[1] Ahmad, Mohammad, et al. "Optimized Snort-Based Intrusion Detection for ARM Architectures." *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, 2021, pp. 245-260.
[2] Kumar, Anil, and Priya Singh. "Cybersecurity Education Through Raspberry Pi-Based IDS Frameworks." *ACM Computing Surveys*, vol. 53, no. 4, 2020, pp. 1-35.
[3] Wang, Haoyu, et al. "Performance Benchmarking of ARM Devices for Professional Penetration Testing." *IEEE Security & Privacy*, vol. 20, no. 2, 2022, pp. 78-92.
[4] Prasad, Rajesh, et al. "Kali Linux Optimization for Single-Board Computers." *Proceedings of the USENIX Security Symposium*, 2021, pp. 1123-1140.
[5] Sharma, Tanvi, and Mohit Thakur. "Persistent Secure Tunneling for Remote Security Assessments." *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, 2020, pp. 1024-1038.
[6] Schmidt, Erik, et al. "Vulnerability Scanning Optimization on Resource-Constrained Devices." *ACM Transactions on Privacy and Security*, vol. 24, no. 3, 2021, pp. 1-30.
[7] Green, Benjamin, et al. "ARM Architecture Support in Modern Penetration Testing Frameworks." *IEEE Security & Privacy*, vol. 20, no. 4, 2022, pp. 55-68.
[8] Becker, Stefan, et al. "Automating Security Assessment Workflows." *ACM Transactions on Information and System Security*, vol. 23, no. 2, 2020, pp. 1-28.
[9] Martins, João, and Luís Santos. "Hands-On Cybersecurity Education Using Raspberry Pi." *ACM Transactions on Computing Education*, vol. 21, no. 4, 2021, pp. 1-24.
[10] Velasquez, Raul, et al. "Integrated Vulnerability Scanning Pipelines." *IEEE Transactions on Information Forensics and Security*, vol. 17, 2022, pp. 1568-1583.
[11] Nair, Suresh, et al. "Energy-Efficient IoT Security Monitoring." *IEEE Internet of Things Journal*, vol. 8, no. 12, 2021, pp. 9876-9890.
[12] Gonzalez, Miguel, and Juan Rios. "Securing Reverse SSH Tunnels Against MITM Attacks." *Proceedings of the USENIX Security Symposium*, 2020, pp. 423-440.
[13] Zhang, Yifan, et al. "Bandwidth Optimization in Remote Security Administration." *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, 2022, pp. 845-859.
[14] Thomas, Kevin, and David George. "Power Efficiency in Continuous Security Monitoring." *ACM SIGMETRICS Performance Evaluation Review*, vol. 48, no. 3, 2021, pp. 45-50.
[15] Rahman, Nasir, and Hasan Akhtar. "Distributed Vulnerability Scanning Architectures." *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, 2020, pp. 1845-1860.
[16] Liu, Zheng, et al. "Machine Learning for Security Scan Scheduling." *ACM Transactions on Autonomous and Adaptive Systems*, vol. 17, no. 2, 2022, pp. 1-25.
[17] Chandra, Bikram, and Vivek Gupta. "Consistent Security Assessment in Heterogeneous Networks." *IEEE Transactions on Services Computing*, vol. 14, no. 5, 2021, pp. 1423-1436.
[18] Tiwari, Amit, and Rohan Patel. "Cluster Performance in Distributed Security Scanning." *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, 2020, pp. 1125-1139.
[19] Kalita, Anupam, et al. "Automated Security Reporting Using NLP." *ACM Transactions on Internet Technology*, vol. 22, no. 3, 2022, pp. 1-22.
[20] Banerjee, Rahul, and Pratik Mitra. "Threat Intelligence Integration in Lightweight Scanners." *IEEE Transactions on Information Forensics and Security*, vol. 16, 2021, pp. 3245-3260.