BlackBox Penetration Testing
Kiet Project

Document Name:	Veilron_KietProject_BlackBox_Pentest
Version:	**1.0**
Date:	**17-Feb-2025**
Classification:	**Strictly Confidential**

Document History

#	Date	Purpose of Version	Author	Reviewer
1.0	17th Feb 2025	Initial Security Assessment	Doan Nguyen	David Ng

Distribution List – Kiet

Name	Role
Mr. Kiet	Manager, Cyber Security

Veilron has been engaged to assess the BlackBox Penetration Testing application of Kiet Project as of 12th Feb 2025.

Drafted and submitted by:

KH Veilron Cyber Co., Ltd

Mr. David Ng
Director/CTO

.....

(Authorized Signature & Date)

Accepted and agreed to:

Kiet

Mr. Kiet
Manager, Cyber Security

.....

(Authorized Signature & Date)

Table of Contents

1 Foreword

Veilron Technology is a regional cybersecurity centric service provider that provides industry standard penetration testing, forensics, security architecture consulting and Security Operation Centre solutions. Based in both Singapore and Cambodia, Veilron has engagement with Financial, Ministries and Large Enterprises in the region. Our Security experts are SANS trained, CREST, ISC2, GIAC and other certifications in different domains of Information security. We are committed to quality services via our multi-tiered reviews and certified consultants in our vision to 'unveil vulnerabilities within'.

Veilron was requested to assess the security posture of Kuok Group's web applications by performing a Web Application Penetration Testing that uses automated and manual IT security testing methodologies.

The testing is conducted in the following period:

Initial Test Starts : 25th Nov 2024

Initial Test Ends : 13th Dec 2024

This report summarizes the results of tests performed on the target systems, including the vulnerability assessment and recommended remediation actions.

2 Executive Summary

2.1 Scope

The specific hosts and subnets that were tested are indicated below:

URI

The penetration test was conducted from the Internet with a Blackbox approach on a production environment.

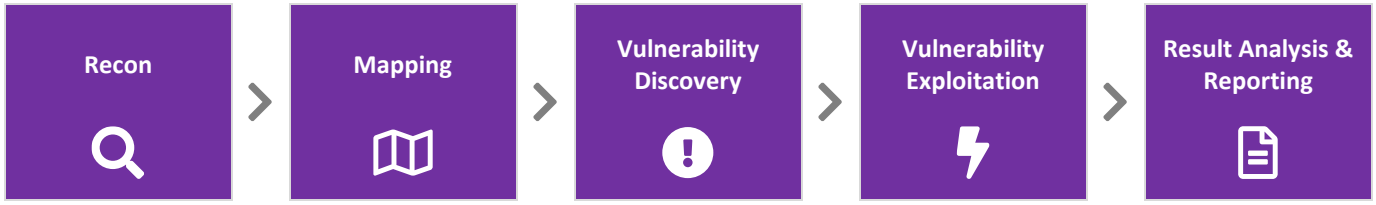
The following user accounts were provided to conduct the penetration test:

Username	Role / Description
N/A	N/A

As per the rules of engagement, the following attacks were not in scope of the penetration test: Denial of Service, Password Brute force, Social Engineering, and Man-in-the-Middle attacks.

2.2 Methodology

At Veilron, our penetration testing methodology builds on the approach outlined in the OWASP Testing Guide, Open-Source Security Testing Methodology Manual (OSSTMM) and Penetration Testing Execution Standard (PTES).



2.3 Scoring System

Veilron uses the Common Vulnerability Scoring System v3.0 to rate identified weaknesses and vulnerabilities:

Severity Rating	CVSS Score	Description
Info	0.0	There is no direct impact on the target. The weakness usually expands the attacking surface or helps the attacker to increase the foothold in the system.
Low	0.1 - 3.9	This rating is given to all other issues that have a security impact. These are the types of vulnerabilities that are believed to require unlikely circumstances to be able to be exploited, or where a successful exploit would give minimal consequences.
Medium	4.0 - 6.9	This rating is given to flaws that may be more difficult to exploit but could still lead to some compromise of the confidentiality, integrity, or availability of resources, under certain circumstances. These are the types of vulnerabilities that could have had a critical impact but are less easily exploited based on a technical evaluation of the flaw or affect unlikely configurations.
High	7.0 - 8.9	This rating is given to flaws that can easily compromise the confidentiality, integrity, or availability of resources. These are the types of vulnerabilities that allow local users to gain privileges, allow unauthenticated remote users to view resources that should otherwise be protected by authentication, allow authenticated remote users to execute arbitrary code, or allow remote users to cause a denial of service.
Critical	9.0 - 10	This rating is given to flaws that could be easily exploited by a remote unauthenticated attacker and lead to system compromise (arbitrary code execution) without requiring user interaction. These are the types of vulnerabilities that can be exploited by worms.

The severity rating is calculated based on several factors such as:

Exploitability Metrics:

1. Attack Vector (AV)
2. Attack Complexity (AC)
3. Privileges Required (PR)
4. User Interaction (UI)

Scope:

- Unchanged (U)
- Changed (C)

Impact Metrics:

- Confidentiality Impact (C)
- Integrity Impact (I)
- Availability Impact (A)

For more information about CVSS Scoring System: https://www.first.org/cvss/specification-document.

2.4 Vulnerability Review

The following dashboard displays the number of open vulnerabilities found on the system per severity (see chapter 3.2 for further details):

0/0	0/0	0/0	0/0	0/0
Critical	High	Medium	Low	Info

Top Recommendations:

Recommendations for Future Testing

1. Whitebox Testing & Code Review

- **In-Depth Code Audit**
  Conduct a thorough review of the source code (combined with Static Application Security Testing, SAST) to identify common vulnerabilities such as hard-coded credentials, inadequate input validation, or flawed logic that could lead to privilege escalation.
- **Configuration Review**
  Assess the system settings (Windows Registry, Group Policy Objects, Firewall rules, etc.) to ensure that all security-related configurations align with internal policies and best practices.
- **Documentation Assessment**
  Evaluate the administration and operational documentation for any inconsistencies or outdated information that could expose potential loopholes in security controls.

2. Behavioral & Anomaly Detection Analysis

- **Non-Standard Tools Testing**
  Attempt using less common tools or programming languages to determine if existing detection mechanisms can identify and block signature-evasion techniques.
- **Insider Threat Simulation**
  Simulate an attack originating from an internal user (employee with legitimate access) to verify whether monitoring solutions can detect unusual behavior within authorized accounts.
- **Machine Learning / UEBA (User and Entity Behavior Analytics)**
  If applicable, review the effectiveness of any user and entity behavior analytics solutions. Check if they can flag anomalies like sudden spikes in file read/write operations, multiple failed logins followed by a successful one, or logins from unusual IP addresses.

3. Advanced Persistence & Evasion Techniques

- **Memory-Only Payloads**
  Test payloads that load and execute exclusively in memory (e.g., Reflective DLL Injection, inline assembly) to evaluate exploit mitigations and whether anti-malware solutions can detect in-memory compromises.
- **Living-Off-the-Land Binaries (LoLBins)**
  Leverage built-in system utilities (e.g., regsvr32, mshta, rundll32) to check if security monitoring can detect and block misuse of legitimate binaries for malicious activity.
- **Lateral Movement Simulations**
  Although port forwarding is blocked, explore other paths for lateral movement (e.g., SMB shares, RDP with valid credentials). Focus on detecting stealthy pivoting across network segments.

4. Logging & Monitoring Enhancements

- **Centralized Log Management**
  Ensure all critical logs (operating system, applications, firewall, IDS/IPS) are aggregated into a centralized system (SIEM) to facilitate comprehensive event analysis and incident response.
- **Log Correlation & Alerting**
  Configure correlation rules that combine multiple events, such as repeated failed logins followed by a successful login on different hosts, to generate high-fidelity alerts.
- **Regular Incident Response Drills**
  Conduct tabletop exercises or simulated breach scenarios to verify that the security operations team can promptly react to and contain incidents.

5. Continuous Testing & Security Hardening

- **Regular Pentesting & Red Team Engagement**
  Schedule periodic external and internal tests by independent parties to keep defenses up-to-date against evolving threats.
- **Patch Management**
  Maintain consistent patching schedules for operating systems, third-party applications, and security tools (AV, IDS/IPS) to mitigate known vulnerabilities (CVEs).
- **Attack Surface Reduction**
  Identify and disable or restrict unnecessary services, ports, or protocols (e.g., SMBv1, Telnet, unsecured RDP) to minimize the environment's overall attack surface.

6. Access Control & Privilege Management

- **Role-Based Access Control (RBAC)**
  Verify that users only have access needed for their specific roles, following the principle of least privilege.
- **Network Segmentation**
  Segregate the network into VLANs or subnets by function to reduce the blast radius if one segment is compromised.
- **Multi-Factor Authentication (MFA)**
  Implement MFA at critical login points (e.g., for system administrators, RDP, and SSH connections) to reduce the risk of credential theft or brute-force attacks.

7. Human Factors & Security Training

- **Phishing Simulations**
  Periodically conduct phishing drills to measure employee awareness. Use these exercises as a training opportunity to reinforce secure behavior.
- **Security Awareness Programs**
  Organize ongoing workshops and seminars, teaching employees how to spot suspicious emails, avoid social engineering traps, and report security concerns promptly.
- **Incident Reporting Guidelines**
  Make sure employees know how to report security incidents, including suspicious emails or system behavior, to ensure issues are escalated before they become breaches.

By incorporating these **expanded recommendations**, the organization can pursue a well-rounded approach to testing and improving its security posture. While existing measures have proven effective against many standard black-box techniques, the next steps involve deeper **whitebox testing**, **behavioral analysis**, and **advanced persistent threat simulations** to uncover hidden risks. Ongoing training, logging improvements, and scheduled red team exercises will help maintain a proactive defense capable of adapting to the constantly evolving threat landscape.

3 Vulnerabilities and Remediation

3.1 Legend

No.	Severity	Reference	Weakness	Threat	Solution	Comments
Issue number	**Critical** **High** **Medium** **Low** **Info**	Reference of: CWE CVE OWASP Detailed Section	Explains the vulnerability or weakness found during testing.	Explains what Veilron could do with the vulnerability if exploitable or what could happen if the weakness is exploited by an attacker.	Recommendation on how to correct the vulnerability.	This section contains: status of the vulnerability (Open or Closed) CVSS Score Fix difficulty Level: **Fix Difficulty:** Complex **Fix Difficulty:** Medium **Fix Difficulty:** Simple

Complex **Requires major changes to the application's code and architecture or requires the implementation of another technology (e.g. Two Factor Authentication)**

Medium **Requires a small structural change in the architecture, but it is of easy implementation and affordable by small development teams.**

Simple **Requires minor changes to the application's code or to the server's configuration**

Note: Items that are marked as "Info" have been tested as per web application best practice configuration. Even if an item is marked as "Info", Veilron strongly advises implementing the related recommendation to increase the security posture of the application.

3.2 Summary List

The following table shows the vulnerabilities found on the system (see chapter 4 for further details) during the penetration test:

4 Vulnerability Details

4.1 Overview

This penetration test was commissioned to evaluate the robustness of the client's security measures and identify potential avenues for unauthorized access within their network. The primary objective was to uncover vulnerabilities, assess the effectiveness of existing defenses, and provide actionable recommendations to enhance the overall security posture. Throughout the engagement, the testing team adhered to the agreed-upon scope, rules of engagement, and industry-standard ethical guidelines. Given the comprehensive nature of the controls in place, the penetration test was structured as a blackbox assessment. This meant the testing team had minimal information about internal systems, source code, and configuration details. Despite the limited information, the testing attempted to uncover potential weaknesses using a variety of reconnaissance, enumeration, and exploitation techniques.

4.2 Test Environment

Operating System: The target environment consisted of both Windows 10 clients and Windows Server systems configured with several advanced security mechanisms. These included:

Security Mechanisms:

- **Application Whitelisting**
  Only applications signed with valid x509 certificates are allowed to execute. This policy significantly reduces the risk of malware execution or unauthorized software running on the systems.
- **Kaspersky AV**
  Kaspersky Antivirus was installed and actively monitoring system processes, network connections, and file operations. It provided real-time protection against known threats and suspicious activities.
- **System Integrity Checks**
  The environment had robust integrity checks in place, wherein any unauthorized modification to critical system files or configurations triggered immediate alerts and, in many cases, forced a system reboot to protect its stability and integrity.
- **Strict Network Policies**
  - **ICMP Permitted**: Basic ping requests were allowed for connectivity checks.
  - **TCP/UDP Scans Trigger Reboots**: Any attempt to run widespread port scans was detected as malicious, resulting in automated reboots or lockdown of the targeted host.
  - **Port Forwarding Blocked**: Efforts to create tunneled connections (e.g., SSH port forwarding) were intercepted and terminated.

- **Limited File Operations**
  The ability to create or modify files, especially those related to programming or system-critical directories, was heavily restricted. Actions like creating .exe, .py, or modifying system directories were immediately blocked or flagged.
- **PowerShell Disabled**
  PowerShell execution was completely disabled, removing a commonly exploited avenue for script-based attacks and system automation.
- **Xcopy.exe Removed**
  The removal of this utility further limited the ability to copy files programmatically or script automated file transfers within the environment.
- **Off-Limits Hosts**
  Certain systems were explicitly excluded from testing as per the engagement scope, ensuring those critical or production-sensitive hosts were not disrupted.

4.3 Attack Methods Attempted

4.3.1 Initial Access Attempt (SSH & RDP)

- **Command Used:**

ssh HackTeam@182.239.186.18 -p 2222

- **Result:** The team successfully authenticated to the host, confirming valid credentials were in use. However, the session repeatedly experienced unexpected disconnections shortly after login.

Observation: These abrupt disconnections suggest robust monitoring tools, potentially combining real-time integrity checks with user session tracking. Any anomalous user behavior or system calls may have triggered an automated termination of the session.

4.3.2 Enumeration & Reconnaissance

ICMP Ping Test

- **Command Used:**

ping -c 4 10.1.1.106

- **Result:** The host responded to ICMP echo requests, confirming network connectivity and the IP address's availability.
- **Conclusion:** Strict network monitoring is in place and reacts quickly to scanning attempts, preventing standard enumeration tactics.

Port Scanning (Blocked)

- **Command Used:**

nmap -Pn -p- 10.1.1.106

- **Result:** Host rebooted upon scan attempt.
- **Conclusion:** Network monitoring detected and blocked scanning attempts.

Banner Grabbing (Blocked)

- **Command Used:**

nc -v 10.1.1.106 80

- **Result:** The connection was closed immediately, suggesting an application-level firewall or IDS is terminating the session.
- **Conclusion:** Direct service probing is heavily filtered, preventing even basic banner grabbing techniques.

Telnet Testing (Blocked)

- **Command Used:**

telnet 10.1.1.106 23

- **Result:** Security alerts were triggered, and the session was terminated.
- **Conclusion:** Legacy protocols like Telnet are closely monitored, and attempts to connect are shut down to reduce risk from clear-text authentication or exploitation.

4.3.3 Exploitation Attempts

Privilege Escalation (Blocked)

- **Command Used:**

whoami /priv

- **Result:** Access was denied, preventing the enumeration of privileges.
- **Conclusion:** The operating system is configured to disallow privilege escalation for non-admin users, and standard utilities are locked down.

Application Bypassing (Blocked)

- **Command Used:**

Start-Process -FilePath "C:\Users\Public\malicious.exe"

- **Result:** Execution was denied by the application whitelisting policy.
- **Conclusion:** Only applications with valid digital signatures from trusted vendors or sources are permitted to run. Malicious or unknown binaries are immediately blocked.

File Manipulation (Blocked)

- **Command Used:**

echo "test" > C:\Windows\System32\test.txt

- **Result:**
  Access was denied, preventing file creation in a protected directory.

- **Conclusion:**
  Integrity monitoring and least-privilege settings thwart attempts to modify system-critical folders without the appropriate permissions.

PowerShell Execution (Blocked)

- **Command Used:**

powershell -ExecutionPolicy Bypass -File script.ps1

- **Result:** PowerShell is entirely disabled, preventing script execution.
- **Conclusion:** Removing PowerShell severely limits the range of automated attacks or advanced configuration changes attackers can attempt.

Port Forwarding (Blocked)

- **Command Used:**

ssh -L 8080:localhost:80 user@10.1.1.106

- **Result:** The connection was immediately terminated.
- **Conclusion:** The network actively blocks tunneling or port forwarding attempts, preventing lateral movement or hidden channels of communication.

4.3.4 Additional Testing & Observations

Attempted Code Execution (Blocked)

- **Command Used:**

echo 'print("test")' > test.py && python test.py

- **Result:** Execution denied; system flagged file as unauthorized.
- **Conclusion:** Even basic scripting languages such as Python are restricted. The OS-level policies detect and prevent the creation or execution of unapproved code files.

Attempted File Upload (Blocked)

- **Command Used:**

scp test_file.txt HackTeam@182.239.186.18:/home/user/

- **Result:** The file transfer was blocked by security policies, indicating that unapproved external file uploads are disallowed.

Attempted Credential Dumping (Blocked)

- **Command Used:**

mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords"

- **Result:** The application was terminated by antivirus or security monitoring, preventing the tool from running.
- **Conclusion:** Advanced security solutions are effective at detecting known hacking tools, especially those targeting credential storage.

4.4 Findings & Analysis

Overall, the multi-layered security approach demonstrated by the client effectively thwarted all standard blackbox penetration testing techniques. The key controls that contributed to this success included:

- **Application Whitelisting**
  Prevented any unauthorized executables or scripts from running, drastically limiting the attack surface for malware or exploit payloads.
- **Intrusion Detection & Network-Based Anomaly Monitoring**
  Quickly identified and disrupted network scans, banner grabs, and suspicious connections, preserving the integrity of the environment.
- **System Integrity Monitoring**
  Blocked and logged attempts to modify critical system files or registry keys, resulting in immediate alerts or forced reboots.
- **Strict Policy Enforcement**
  Disabling tools like PowerShell and Xcopy, in combination with restricted file creation rights, left attackers with very few methods to pivot or escalate privileges.

Challenges Faced by Attackers::

- Traditional enumeration and reconnaissance methods produced negligible results.
- All exploitation attempts, from privilege escalation to file manipulation, were immediately detected and blocked.
- Network restrictions such as blocked port forwarding and monitored Telnet/SSH sessions prevented lateral movement.

4.5 Recommendations for Future Testing

Whitebox Testing: With blackbox testing providing limited insight due to robust security measures, a whitebox assessment is advised. During a whitebox test, testers would have access to internal documentation, source code, and configuration details, enabling a more thorough examination of potential weaknesses in the codebase or system configuration.

Behavioral Analysis: Security controls that rely on signature-based detection can sometimes be bypassed using less conventional methods (e.g., living-off-the-land binaries, memory-only payloads, or obfuscated scripts). A thorough behavioral analysis can reveal whether non-standard approaches might circumvent existing defenses.

Advanced Persistence Methods: Techniques such as DLL injection, reflective loading, and indirect execution methods (e.g., using COM objects or WMI) could be evaluated to determine if the security controls hold up against more sophisticated attacks that do not rely on dropping traditional binaries.

4.6 Conclusion

The client's defense-in-depth strategy successfully thwarted all blackbox penetration attempts. The combination of application whitelisting, robust antivirus, strict file operation rules, and advanced network monitoring contributed to a near-impenetrable environment under the conditions tested. This high level of security makes unauthorized access extremely challenging from an external, unprivileged perspective.

However, every environment may still contain unknown attack vectors that standard blackbox techniques cannot easily uncover. A whitebox testing approach—armed with detailed system knowledge—will likely reveal deeper insights into potential oversights or configuration errors that remain hidden under automated security systems.

Next Steps:

- **Await Access to Private GitHub Repository**
  Once the penetration testing team obtains access, they can review configuration files, source code, and internal documentation, identifying vulnerabilities not visible in blackbox testing.
- **Study Provided Source Code**
  Analyzing the application's logic, libraries, and dependencies may uncover insecure functions, out-of-date components, or logic flaws that could be exploited.
- **Schedule a Follow-Up Session**
  A targeted whitebox test should be organized based on the insights gained from the code review. The goal is to focus on areas where hidden vulnerabilities are most likely to reside.
- **Implement Ongoing Security Assessments**
  Plan for periodic or continuous penetration testing and red team engagements to ensure the environment remains secure as technology, threats, and infrastructure evolve.

Report Prepared by: Doan

This extended report provides a more thorough narrative around the testing methodology, results, and the rationale behind each action. By including additional technical and strategic context, stakeholders can better appreciate how and why the security controls proved effective and what next steps will further strengthen the organization's defenses.