IBM Storage Scale System (SSS) and Storage Scale Penetration Testing Report



Detailed report for the Network, Containers/PODs and CLI Penetration testing performed by PTC for IBM Storage Scale System (SSS) and Storage Scale

Date: Wed Jun 12 11:50:48 IST 2025

Client Contact: praygupt@in.ibm.com, Ratan.Swami@ibm.com, faryarag@in.ibm.com, sumitkuo@in.ibm.com, ravisure@in.ibm.com

Report Prepared By: K V, Praveen

**ASSESSOR TEAM**

K V, Praveen[OSCP,eWPT,eWPTX]

Penetration Tester/Ethical Hacker

praveenkv@in.ibm.com

Kaur, Ramandeep

Penetration Tester/Ethical Hacker

rakaur08@in.ibm.com

**REPORT QA**

S, Arun

Technical Lead

aruns014@in.ibm.com https://w3.ibm.com/w3publisher/ptc/security-services

**Table of Contents**

[**Disclaimer** 4](#_Toc200562818)

[**IBM Confidential** 4](#_Toc200562819)

[**Copyright** 4](#_Toc200562820)

[**TESTING METHODOLOGY** 6](#_Toc200562821)

[**EXECUTIVE SUMMARY** 8](#_Toc200562822)

[**Overview** 8](#_Toc200562823)

[**Timeline** 8](#_Toc200562824)

[**Summary of Findings by Severity** 8](#_Toc200562825)

[**Overall Security Vulnerability Snapshot** 8](#_Toc200562826)

[**Repeat Vulnerability Snapshot From Previous Test Cycle** 8](#_Toc200562827)

[**Summary of Open Vulnerabilities** 9](#_Toc200562828)

[**Scope** 11](#_Toc200562829)

[**Setup Details** 12](#_Toc200562830)

[**Build/Version** 12](#_Toc200562831)

[**Network and System Details** 12](#_Toc200562832)

[**Container/PODs Details** 12](#_Toc200562833)

[**CLI Details** 12](#_Toc200562834)

[**DETAILED PEN FINDINGS** 13](#_Toc200562835)

[**Network Security Vulnerability** 13](#_Toc200562836)

[Public Target Systems open ports identified during network mapping 13](#_Toc200562837)

[**194309: Portmapper UDP based amplification attack *(Low)*** 13](#_Toc200562838)

[**194311: Weak SSH Algorithms In Use *(Low)*** 15](#_Toc200562839)

[**194310: SSH Password-based Authentication Enabled *(Informational)*** 17](#_Toc200562840)

[**Container/PODs Security Vulnerability** 19](#_Toc200562841)

[**193926: Unrestricted Internet Access/Outbound Connections *(Medium)*** 19](#_Toc200562842)

[**193928: Containers Running with Root Privileges (Root Containers) *(Medium)*** 21](#_Toc200562843)

[**194312: Restrict Container From Acquiring Additional Privileges (No-New-Privileges) *(Informational)*** 23](#_Toc200562844)

[**194317: Missing Seccomp profile *(Informational)*** 24](#_Toc200562845)

[**194313: Container's Root Filesystem Has Read-Write Permission *(Informational)*** 25](#_Toc200562846)

[**194316: Containers With No CPU/Memory Limitations *(Informational)*** 26](#_Toc200562847)

[**194315: Running SSH Inside Containers *(Informational)*** 27](#_Toc200562848)

[**CLI Security Vulnerability** 29](#_Toc200562849)

[**195190: Container To Host Escape Via RCE And SSH Credentials   *(Critical)*** 29](#_Toc200562850)

[**195172: Hardcoded Credentials *(Medium)*** 36](#_Toc200562851)

[**195165: Unsigned Executable File *(Informational)*** 38](#_Toc200562852)

[**195171: Sensitive Information Passed In CLI Arguments *(Informational)*** 39](#_Toc200562853)

[**195199: Password Returned In Response *(Informational)*** 41](#_Toc200562854)

[**195164: No Binary Obfuscation *(Informational)*** 43](#_Toc200562855)

[**Appendix A - Detailed Testing Methodology** 45](#_Toc200562856)

[**Appendix B - Tools and Utilities** 50](#_Toc200562857)

[**Appendix C** 50](#_Toc200562858)

**Disclaimer**

**Penetration Testing Teams:**

Depending on the offering and scenario, IBM teams engage penetration testing teams from many different approved sources, including independent teams from within IBM such as IBM's Product-Security Technology Centre (PTC). The IBM PTC is an independent team within IBM, and is independent of product/offering teams, having no role in the design, architecture, development, testing and delivery of products and offerings. Following Separation of Duties (SOD) practices, the IBM PTC team will remain independent of the "being-tested" team, to ensure high quality, unbiased and independent penetration testing for offerings.

In this respect IBM follows the PCI Security Standards Council Guideline as described in Section 3, Information Supplement, Penetration Testing Guidance, September 2017: <https://www.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1_1.pdf>

**Penetration Testing Coverage:**

The penetration test results are related to the specific, performed, tests only. They indicate, but do not and cannot measure, the entire security posture (quality of protections) of an application and/or its systems and network, including as deployed in the context of a client’s environment for client-controlled products and offerings.

**IBM Confidential**

The information contained in this report is IBM Confidential. It is believed to be up-to-date and accurate at the time of its preparation. Any inaccuracies, changes or improvements will be documented in subsequent revision. Authorised readers are responsible for their interpretation and use of the information in this report. IBM Corporation will not be liable for any unauthorised use of this report.

**Copyright**

© Copyright IBM Corporation. All Rights Reserved. This is unpublished material and contains trade secrets and other confidential information and is subject to a confidentiality agreement. The unauthorised possession, use, reproduction, distribution, display, or disclosure of this material or the information contained herein is prohibited. The methodology used to test the computer systems is considered proprietary intellectual information of IBM and may not be disclosed without written permission from IBM. IBM gives permission to copy this report for the purposes of disseminating information within your organisation or any regulatory agency.

**TESTING METHODOLOGY**

PTC penetration testing methodology is based on a structured approach applying industry best practices in testing.  
 **Reconnaissance** • Web Application Sitemap collection, API endpoints, ports and services running  
 • Gathering publicly available information  
 • Analysing and understanding architecture of application/network  
 • Listing technologies, third party services, open source libraries and frameworks used in the application  
 • Gathering threat modelling and previously scanned automated scanner reports

**Mapping the attack surface** • Planning the attacks based on the information gathered in the Reconnaissance phase  
 • Checking the misconfigurations, weak/default credentials against the used frameworks  
 • Determining all possible attacks scenarios by combining the mapped surfaces

**Automation Scanners** • Carefully selecting the best suited automated scanner based on the technologies and components used by the application  
 • Executing automated scan to identify possible vulnerabilities

**Manual analysis** • Analysing the application manually to understand the business logic of the application  
 • Crafting application specific test cases that requires manual intelligence and are out of reach to automated scanners   
 • Performing vulnerability analysis and developing proof of concepts for the identified vulnerabilities

**Exploitation** • Exploiting the identified vulnerabilities to mimic the impact of actual attack scenarios  
 • Map the system command executions, data exfiltration and possible damage an actual exploitation can make to the target system

**Reporting and Documentation** • Document the identified security vulnerabilities  
 • Assess the risk and impact and accordingly calculate CVSS score for each findings



**For more details please refer Appendix A.**

**EXECUTIVE SUMMARY**

**Overview**

Product-Security Technology Centre (PTC) performed Penetration Testing (Pen Test) on the IBM Storage Scale System (SSS) and Storage Scale service. This report documents the findings of the Pen Test conducted.

**Timeline**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Description** | **Product Version** | **Start Date** | **End Date** | **Remarks** |
| 1.0 | Initial Pentest | 5.2.3 | 19-May-2025 | 12-Jun-2025 |  |

**Summary of Findings by Severity**

**Overall Security Vulnerability Snapshot**

|  |  |  |  |
| --- | --- | --- | --- |
| **Severity** | **Network Count** | **Container Count** | **CLI Count** |
| Critical | 0 | 0 | 1 |
| High | 0 | 0 | 0 |
| Medium | 0 | 2 | 1 |
| Low | 2 | 0 | 0 |
| Informational | 1 | 5 | 4 |
| **Total** | **3** | **7** | **6** |

**Repeat Vulnerability Snapshot From Previous Test Cycle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Severity** | **Network Count** | **Container Count** | **CLI Count** |
| Critical | 0 | 0 | 0 |
| High | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 |

**Summary of Open Vulnerabilities**

**Network Security Vulnerability**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Type** | **Defect Id** | **Severity** | **CVSS Score** |
| Active Reconnaissance | Denial-of-Service | [194309](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194309) | Low | [3.5](https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:L) |
| Auditing | SSH Weak Algorithms Supported | [194311](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194311) | Low | [3.1](https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N) |
| Active Reconnaissance | Security Misconfiguration | [194310](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194310) | Informational |  |

**Container/PODs Security Vulnerability**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Type** | **Defect Id** | **Severity** | **CVSS Score** |
| Active Reconnaissance | Security Misconfiguration | [193926](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=193926) | Medium | [5.1](https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N) |
| Active Reconnaissance | Security Misconfiguration | [193928](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=193928) | Medium | [4.5](https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:L) |
| Active Reconnaissance | Security Misconfiguration | [194312](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194312) | Informational |  |
| Active Reconnaissance | Security Misconfiguration | [194313](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194313) | Informational |  |
| Active Reconnaissance | Security Misconfiguration | [194316](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194316) | Informational |  |
| Active Reconnaissance | Security Misconfiguration | [194315](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194315) | Informational |  |
| Auditing | Security Misconfiguration | [194317](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194317) | Informational |  |

**CLI Security Vulnerability**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Type** | **Defect Id** | **Severity** | **CVSS Score** |
| Input Validation | Remote Command Execution | [195190](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195190) | Critical | [9.9](https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H) |
| Active Reconnaissance | Security Misconfiguration | [195172](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195172) | Medium | [6.8](https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N) |
| Configuration and Deployment Management | Security Misconfiguration | [195165](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195165) | Informational |  |
| Active Reconnaissance | Security Misconfiguration | [195171](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195171) | Informational |  |
| Information Disclosure | Information Disclosure | [195199](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195199) | Informational |  |
| Reverse Engineering | Source Code Dump retrieved | [195164](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195164) | Informational |  |

Pen Testing performed on the service for the given scope shows that the service and infrastructure has "High" risk against attacks and the overall security assessment for the product as tested is classified as **"Weak"**.

**Scope**

The product team authorised the PTC to perform the Pen Test on their Network, Containers/PODs and CLI in the provided Test environment.

The Pen Test was performed to simulate how a hacker might attack the system to unearth the vulnerabilities and find any security weaknesses in the Network, Containers/PODs and CLI which could lead to exploiting the vulnerabilities to any undesired consequences.

The Pen Test was performed in an ethical hacking manner only, with the intention to find security vulnerabilities and report discovered vulnerabilities to the product owner. This will allow the appropriate remediation prior to the release of the offering, with the goal of minimizing the risk to client using this product.

For more details about the strategy used for Pen Test, refer to the section Testing Methodology.

**The following table outlines the overall scope of the penetration testing performed:**

|  |  |  |
| --- | --- | --- |
| **Component** | **In-Scope/ Out-Scope** | **Remarks** |
| Web Application | Out of Scope | No web application developed for the product |
| API | Out of Scope | No Rest APIs developed for the product |
| Network | In-Scope |  |
| Mobile | Out of Scope | No mobile application developed for the product |
| Thick Client | Out of Scope | Application is web based and no thick client involved. |
| Container | In-Scope |  |
| CLI | In-Scope |  |

**Setup Details**

The Pen Test was performed on the Network, Containers/PODs and CLI hosted on a Pre-Production environment.

**Build/Version**

The Pen Test was performed on the version 5.2.3.

**Network and System Details**

The Pen Test was performed on the below domain/IP(s)

**Public IP Details**

* 9.114.12.78
* 9.114.12.79

**Container/PODs Details**

* utilityBareMetal-api-official
* utilityBareMetal-rcl-official
* utilityBareMetal-deploygui-official

**CLI Details**

**/usr/lpp/mmfs/bin/essctl**

**DETAILED PEN FINDINGS**

**Network Security Vulnerability**

Public Target Systems open ports identified during network mapping

|  |  |  |
| --- | --- | --- |
| **IP** | **PORT** | **SERVICE** |
| 9.114.12.79 | 22 | ssh |
| 111 | rpcbind |
| 514 | shell |
| 1191 | gpfs |
| 4739 | ipfix |
| 9085 | ibm-rsyscon |
| 9.114.12.79 | 22 | ssh |
| 111 | rpcbind |
| 42007 | RPC |

[**194309: Portmapper UDP based amplification attack**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194309) ***(Low)***

|  |
| --- |
| **Description**  Portmapper (also referred to as rpcbind, portmap or RPC Portmapper) is a service, which always listens on tcp and udp 111, and is used to map other RPC services (such as nfs, nlockmgr, quotad, mountd, etc.) to their corresponding port number on the server. When a remote host makes an RPC call to that server, it first consults with portmap to determine where the RPC server is listening. Querying portmapper is a small request which generates a large response, which makes it a good candidate for DDoS attacks.  **References:**  https://ddos-guard.net/en/terms/attacks/port-111-rpcbind-vulnerability  https://www.securityweek.com/rpc-portmapper-abused-ddos-attack-reflection-amplification/  Below are the related screenshots:  workitemimagea  workitemimagea |
| **CVSS Score**  3.5 |
| **CVSS Vector**  CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:L |
| **Affected IP(s)**  9.114.12.79:111  9.114.12.78:111 |
| **Vulnerability Impact**  An attacker can use a malformed UDP request to the port-mapper service and reflected large size amplified response to be used to perform DOS on another service. |
| **Fix Recommendation / Mitigation**  Disable or restrict Portmapper: If your system does not require Portmapper, it is recommended to disable it altogether. If Portmapper is necessary for certain services, restrict its access to trusted networks or specific IP addresses.  Implement firewall rules: Configure your network firewall to block incoming traffic on UDP port 111 (the default Portmapper port) from external sources. |

[**194311: Weak SSH Algorithms In Use**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194311) ***(Low)***

|  |
| --- |
| **Description**  Weak SSH ciphers, Kex, and MAC algorithms are cryptographic algorithms used during the SSH communication process.  Ciphers are responsible for encrypting the data sent between two devices. Weak ciphers can be easily broken, allowing attackers to intercept and read sensitive information.  Key Exchange (Kex) algorithms are used to securely exchange encryption keys between devices. Weak Kex algorithms can lead to key compromise and may be exploited to launch Man-in-the-Middle (MitM) attacks.  MAC (Message Authentication Code) algorithms ensure data integrity by detecting any unauthorized modifications to the transmitted data. Weak MAC algorithms can be exploited to tamper with data and manipulate the communication process.  Identifying and addressing weak cryptographic algorithms is essential to maintaining a secure SSH environment.  **References:**  https://community.progress.com/s/article/SSH-Weak-Key-Exchanges-Ciphers-HMAC-Sunset-on-3-17-2019  https://wiki.archlinux.org/title/OpenSSH#Security\_best\_practices  Detected the following weak MACs:  hmac-sha1-etm@openssh.com hmac-sha1  Detected the following weak HostKey algorithms:  rsa-sha2-512 ecdsa-sha2-nistp256  rsa-sha2-256  Please refer below screenshot:  workitemimagea |
| **CVSS Score**  3.1 |
| **CVSS Vector**  CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Affected IP(s)**  9.114.12.79:22  9.114.12.78:22 |
| **Vulnerability Impact**  The use of weak SSH ciphers, Kex, and MAC algorithms can lead to several security risks, including:  "Confidentiality and Integrity Breach: Attackers can intercept and read sensitive data transmitted through the weak SSH connection, leading to a breach of confidentiality. They can also modify the data being transmitted, leading to a breach of integrity.  "Man-in-the-Middle (MitM) Attacks: Weak Kex algorithms can be exploited to launch MitM attacks, where attackers can intercept and modify the SSH communication, leading to data interception, modification, and theft.  "Tampering Attacks: Weak MAC algorithms can be exploited to launch tampering attacks, where attackers can manipulate the transmitted data, leading to data tampering, modification, and theft. |
| **Fix Recommendation / Mitigation**  To fix weak SSH ciphers, Kex, and MAC algorithms, the following steps should be taken:  "Upgrade to the latest version of SSH, which supports strong ciphers, Kex, and MAC algorithms.  "Disable weak SSH ciphers, Kex, and MAC algorithms. To do this, the SSH configuration file should be modified to remove any weak algorithms and enable only the strong ones.  "Monitor SSH traffic for any suspicious activity, such as unauthorized logins, unusual login patterns, and excessive failed login attempts.  "Implement two-factor authentication to enhance the security of the SSH connection. |

[**194310: SSH Password-based Authentication Enabled**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194310) ***(Informational)***

|  |
| --- |
| **Description**  SSH (Secure Shell) is a protocol used for secure remote access to servers and other network devices. SSH password-based authentication involves logging in to a remote system by providing a username and password. While this method of authentication is simple and easy to use, it is also vulnerable to various security risks.  **References:**  https://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html  https://www.servers.com/support/knowledge/linux-administration/how-to-replace-password-based-ssh-authentication-with-key-based  workitemimagea |
| **Affected IP(s)**  9.114.12.79  9.114.12.78 |
| **Vulnerability Impact**  Using SSH password-based authentication presents several security risks, including:  "Brute force attacks: Attackers can use automated tools to guess or crack weak passwords, leading to unauthorized access to the system.  "Password reuse: If users reuse the same password across multiple accounts, a compromise of one account can lead to the compromise of others.  "Password sniffing: Passwords transmitted in plain text can be intercepted and stolen by attackers.  "Social engineering attacks: Attackers can trick users into revealing their passwords through phishing emails or other forms of social engineering.  "Insider threats: Malicious insiders with authorized access to the system can abuse their privileges to gain unauthorized access to other systems. |
| **Fix Recommendation / Mitigation**  To mitigate the risks associated with SSH password-based authentication, it is recommended to:  "Use SSH key-based authentication: SSH key-based authentication involves generating a public-private key pair on the client system, and then copying the public key to the remote system. This allows the client system to log in to the remote system without the need for a password. The private key should be kept secure on the client system and should not be shared with anyone.  "Disable password authentication: Disable SSH password-based authentication altogether and only allow SSH key-based authentication. This prevents attackers from attempting to guess or crack passwords.  "Disable root login: Disable the ability to log in directly as the root user. This can help prevent attackers from gaining root access to your system if they are able to guess or crack a password.  "Use a firewall: Configure a firewall to restrict access to the SSH service only to trusted IP addresses. This helps to prevent attackers from accessing the SSH service from unauthorized locations. |

**Container/PODs Security Vulnerability**

[**193926: Unrestricted Internet Access/Outbound Connections**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=193926) ***(Medium)***

|  |
| --- |
| **Description**  Containers running critical or confidential applications and components should have a restricted external communication. Upon compromise of these container attacker would be able to exfiltrate/transfer files to (public external networks) and from container without any restrictions. Proper policies should be configured in either container or HOST level.  **References:**  https://docs.docker.com/network/iptables/  It was observed that there are no restriction on outbound connections in containers, in other words untrusted network/internet access should be blocked or preferably whitelisting should be used.  Note: We were able to download a file to a public server due to unrestricted communication from the container.  Please refer below screen shot:  workitemimagea  workitemimagea workitemimagea workitemimagea |
| **CVSS Score**  5.1 |
| **CVSS Vector**  CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N |
| **Affected Container/Pod(s)**  utilityBareMetal-api-official  utilityBareMetal-gui-official  utilityBareMetal-rcl-official |
| **Vulnerability Impact**  If an attacker is in position to initiate connections from a container to the external/untrusted network, exploitation could be trivial. |
| **Fix Recommendation / Mitigation**  Configure and restrict communications from each running containers using iptables. |

[**193928: Containers Running with Root Privileges (Root Containers)**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=193928) ***(Medium)***

|  |
| --- |
| **Description**  Container running as root user introduces security vulnerabilities due to the elevated privileges associated with the root user. Running containers as root increases the risk of potential compromise of the system hosting the container.  It was observed that the containers mentioned below are running with UID 0.  **Note:** If running containers as root is a specific product requirement or necessity due to dependencies or functionality, the severity of the defect related to running containers as root may be lowered. Please share your justification on this.  **Reference:**  https://medium.com/better-programming/running-a-container-with-a-non-root-user-e35830d1f42a  workitemimagea |
| **CVSS Score**  4.5 |
| **CVSS Vector**  CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:L |
| **Affected Container/Pod(s)**  utilityBareMetal-api-official  utilityBareMetal-gui-official  utilityBareMetal-rcl-official |
| **Vulnerability Impact**  Root access can lead to unauthorized access to sensitive data, modifications to the container and potentially disrupt the availability of service. |
| **Fix Recommendation / Mitigation**  Modify the container configurations to run as non-root users, following the principle of least privilege. This reduces the impact of potential security breaches |

[**194312: Restrict Container From Acquiring Additional Privileges (No-New-Privileges)**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194312) ***(Informational)***

|  |
| --- |
| **Description**  It was observed that containers were run without 'no-new-privileges' security option.  **References:**  https://raesene.github.io/blog/2019/06/01/docker-capabilities-and-no-new-privs/  https://docs.datadoghq.com/security\_platform/default\_rules/cis-docker-1.2.0-5.25/  workitemimagea |
| **Affected Container/Pod(s)**  utilityBareMetal-api-official  utilityBareMetal-gui-official  utilityBareMetal-rcl-official |
| **Vulnerability Impact**  Possible privilege escalation using setuid binaries , sudo and other vulnerable components. |
| **Fix Recommendation / Mitigation**  Run docker container with --security-opt=no-new-privileges option.  **Reference:**  https://docs.datadoghq.com/security\_platform/default\_rules/cis-docker-1.2.0-5.25/ |

[**194317: Missing Seccomp profile**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194317) ***(Informational)***

|  |
| --- |
| **Description**  Seccomp (secure computing mode) is used to restrict the set of system calls applications can make, allowing cluster administrators greater control over the security of workloads.  It was observed that seccomp profile was not set inside containers or disabled.  **Reference:**  https://blog.aquasec.com/new-docker-security-features-and-what-they-mean-seccomp-profiles  Below is the related screenshot:  workitemimagea |
| **Affected Container/Pod(s)**  utilityBareMetal-api-official |
| **Vulnerability Impact**  Ignoring seccomp would allow an attacker to make system calls that might compromise the security of the containers. |
| **Fix Recommendation / Mitigation**  Seccomp support is achieved via two annotations in the pod configuration:  - seccomp.security.alpha.kubernetes.io/pod: profile applies to all containers in the pod that do not override  - container.seccomp.security.alpha.kubernetes.io/<container\_name>: container-specific profile override  **Reference:**  https://access.redhat.com/documentation/en-us/openshift\_container\_platform/3.9/html/cluster\_administration/admin-guide-seccomp  https://itnext.io/hardening-docker-and-kubernetes-with-seccomp-a88b1b4e2111 |

[**194313: Container's Root Filesystem Has Read-Write Permission**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194313) ***(Informational)***

|  |
| --- |
| **Description**  It was observed that containers root filesystem has R/W permissions. This increases attack surface since the root filesystem can be tampered.  As a best practice, it is advised to mount containers filesystem as read-only.  **Reference:**  https://docs.datadoghq.com/security\_platform/default\_rules/cis-docker-1.2.0-5.12/  http://www.projectatomic.io/blog/2015/12/making-docker-images-write-only-in-production/  Below is the related screenshot:  workitemimagea |
| **Affected Container/Pod(s)**  All in scope containers |
| **Vulnerability Impact**  An attacker can write exploit scripts or tamper the existing files in containers root filesystem. |
| **Fix Recommendation / Mitigation**  Use --read-only option. This prevents any writes to the containers root filesystem at container runtime and enforces the principle of immutable infrastructure |

[**194316: Containers With No CPU/Memory Limitations**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194316) ***(Informational)***

|  |
| --- |
| **Description**  By default, a container has no resource constraints and can use as much of a given resource as the hosts kernel scheduler allows. Docker provides ways to control how much memory, or CPU a container can use, setting runtime configuration flags of the docker run command.  It is important not to allow a running container to consume too much of the host machines memory. On Linux hosts, if the kernel detects that there is not enough memory to perform important system functions, then it starts killing processes to free up memory. Any process is subject to killing, including Docker and other important applications. This can effectively bring the entire system down if the wrong process is killed.  By default, each containers access to the host machines CPU cycles is unlimited. You can set various constraints to limit a given containers access to the host machines CPU cycles.    **Reference:**  https://docs.docker.com/config/containers/resource\_constraints/  Below are the related screenshots:  workitemimagea |
| **Affected Container/Pod(s)**  All in scope containers |
| **Vulnerability Impact**  By default, Docker containers have access to the full RAM and CPU resources of the host. Leaving them to run with these default settings may lead to performance bottlenecks.  If you don’t limit Dockers memory and CPU usage, Docker can use all the systems resources. |
| **Fix Recommendation / Mitigation**  It is advised to set memory and CPU limits for all the containers.  **Reference:**  https://docs.docker.com/config/containers/resource\_constraints/ |

[**194315: Running SSH Inside Containers**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=194315) ***(Informational)***

|  |
| --- |
| **Description**  During reconnaissance, it was observed that all the containers in scope is running SSH service on port 22. As a best practice don’t run SSH service in container and don't disclose sensitive data like SSH keys inside containers. Instead, run SSH on the host and use docker exec or docker attach to interact with other containers.  **References:**  https://cloud.redhat.com/blog/hardening-docker-containers-images-and-host-security-toolkit  <https://www.tenable.com/audits/items/CIS_Docker_Community_Edition_L1_Docker_v1.1.0.audit:b6e68c652c1bc572fa9d41212ae84895>  https://blog.aquasec.com/container-security-best-practices-for-conscientious-devops  Please refer to the below screenshot(s):  workitemimagea |
| **Affected Container/Pod(s)**  All in scope containers |
| **Vulnerability Impact**  Running SSH service within containers makes managing SSH keys/access policies difficult. This should be avoided if possible. Also, SSH server that is running inside a container may be used by attackers. If attackers gain valid credentials to a container, whether by brute force attempts or by other methods (such as phishing), they can use it to get remote access to the container by SSH. |
| **Fix Recommendation / Mitigation**  As a best practice it is recommended not to run SSH service in a pod and do not disclose sensitive data like SSH keys inside containers. Instead, run SSH on the host and use docker exec or docker attach to interact with the container. |

**CLI Security Vulnerability**

[**195190: Container To Host Escape Via RCE And SSH Credentials**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195190) ***(Critical)***

|  |
| --- |
| **Description**  OS Command Injection occurs when an application unsafely passes user input into a system shell or OS command. If input is not properly sanitized, an attacker can inject arbitrary commands and gain control over the system. The CLI API endpoints constructs all or part of an operating system (OS) command using externally-influenced input from an upstream component. However, it fails to properly neutralize or incorrectly neutralizes special elements that could alter the intended OS command when sent to a downstream component.  It was observed that the CLI commands and API endpoints are vulnerable to Remote Code Execution (RCE), allowing us to inject OS commands and retrieve their output in the response. By leveraging this, we successfully obtained a reverse shell on the containerized target host and ultimately compromised the host with root-level access. This was possible due to the fact that the container in question had ssh keys of the host. An attacker who access the container via any API vulnerability would use these keys to breakout of the isolated container leading to complete compromise of the cluster. Keeping these cases in mind, as per security best practices, it is always important to avoid hosting SSH deamon to be part of the runtime containers.  Reference: https://www.tenable.com/audits/items/CIS\_Docker\_1.11.0\_v1.0.0\_L1.audit:b690a3daa8524fea816e18868ccedb80  **Steps to Reproduce:**  1. There are 2 ways to perform this attack. One with CLI command by executing the malicious payload in the username or password field. In this case:  /usr/lpp/mmfs/bin/essctl essmgmt copy-ssh-id --hostname utilityBareMetal --password "cluster|echo test hello||a #' |id||a #|\" |echo dlqdofdsdq 8gorcmjfyy||a #" --username root --url utilityBareMetal --insecure-skip-tls-verify which reveals users id details.  2. The other way is from API endpoint. Access the [Essmgmt Service](https://9.114.12.78:46443/sss/v3/copy_ssh_id) endpoint which is a POST request.  2. As shown in the screenshot, we injected an OS command payload using multiple concatenations within the username and password parameters, both of which were found to be vulnerable.  3. It was observed that the crafted command with the concatenation is being executed and its output is reflected in the response.  4. By injecting system commands such as id and whoami etc, we were able to view their output in the application's response.  5. Therefore, we injected a reverse shell to obtain an interactive session. Upon successful execution, we gained a reverse shell within the container, as shown in the screenshot.  6. The obtained shell user is a non-root user, therefore we did further reconnaissance on the container machine for privilege escalation.  7. We ran the getcap -r / 2>> /dev/null command to discover binaries with special privileges that could potentially lead to privilege escalation. While some interesting capabilities were found, they could not be exploited due to implemented restrictions.  8. Upon further enumeration of important directories, we identified SSH key pairs located in the /var/mmfs/scaleadm directory.  9. Using the SSH private key, we were able to log in as the root user on the host machine, ultimately leading to full compromise of the host with root-level access.  workitemimagea  workitemimagea  workitemimagea workitemimagea workitemimagea  workitemimagea workitemimagea  workitemimagea workitemimagea  workitemimagea |
| **Parameter**  Most of the CLI arguments and API's parameters are vulnerable to this attack. |
| **Affected CLI/API(s)**  **CLI:** /usr/lpp/mmfs/bin/essctl  **API Endpoints:**  https://9.114.12.78:46443/sss/v3/copy\_ssh\_id  https://9.114.12.78:46443/sss/v3/set\_new\_password\_to\_emsvm23e  https://9.114.12.78:46443/sss/v3/pull\_new\_sss\_code  https://9.114.12.78:46443/sss/v3/pull\_new\_emsvm\_script |
| **Vector**  /usr/lpp/mmfs/bin/essctl essmgmt copy-ssh-id --hostname utilityBareMetal --password "cluster|echo test hello||a #' |id||a #|\" |echo dlqdofdsdq 8gorcmjfyy||a #" --username root --url utilityBareMetal --insecure-skip-tls-verify |
| **Vulnerability Impact**  Successful exploitation of OS command injection vulnerabilities can result in arbitrary command execution with the privileges of the vulnerable application or user. This can lead to unauthorized access to sensitive data, remote shell access, privilege escalation, and potential full system compromise. In containerized environments, such exploitation may also lead to breakout attempts or further compromise of the host infrastructure, depending on configuration and privileges. |
| **Fix Recommendation / Mitigation**  If possible, CLI/APIs should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks that cannot be manipulated to execute unintended commands.  If incorporating user-supplied data into operating system commands is unavoidable, the following two layers of defense should be implemented to prevent attacks:   * User data should be rigorously validated. Ideally, a whitelist of specific accepted values should be used. * Alternatively, only short alphanumeric strings should be accepted. * Input containing any other data, including conceivable shell metacharacters or whitespace, should be rejected. * Ensure all API inputs are strictly validated using allowlists. Reject or escape any special characters that could be used to inject OS commands. * Refrain from using unsafe functions (e.g., system(), exec()) with user input. If necessary, use safe execution methods like parameterized subprocess calls. * Run containers and applications with the minimum privileges required. Avoid running containers as root and drop unnecessary Linux capabilities. * Do not store SSH private keys or sensitive credentials inside containers. Use appropriate permissions and mount secrets securely using a secrets manager. * Use minimal, hardened container images. Disable interactive shells inside containers and isolate them from the host system using tools like AppArmor or SELinux. * Implement logging and alerting for suspicious API behavior, such as command-like input patterns or abnormal output responses. |

[**195172: Hardcoded Credentials**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195172) ***(Medium)***

|  |
| --- |
| **Description**  Hardcoded credentials refer to sensitive authentication details such as usernames, passwords, API keys, or tokens that are embedded directly within application code or binaries. This practice poses a significant security risk, as attackers can extract these credentials through static analysis techniques, enabling unauthorized access to systems or data.  During the course of pentest, static inspection using the strings utility revealed embedded plaintext credentials on /usr/lpp/mmfs/bin/essctl binary. This indicates the application stores authentication details directly within the compiled executable, making them accessible to any user with read access to the file and exposing the application to potential credential theft and misuse.  workitemimagea  workitemimagea |
| **CVSS Score**  6.8 |
| **CVSS Vector**  CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N |
| **Affected CLI/API(s)**  /usr/lpp/mmfs/bin/essctl |
| **Vulnerability Impact**  Hardcoded credentials can be easily extracted by attackers with local access, leading to unauthorized access to internal systems or services. This can result in data breaches, privilege escalation, or lateral movement within the network. If reused elsewhere, compromised credentials could expose other critical environments as well. |
| **Fix Recommendation / Mitigation**  Remove hardcoded credentials and use environment variables, secure configuration files with proper permissions, or a secrets management solution. |

[**195165: Unsigned Executable File**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195165) ***(Informational)***

|  |
| --- |
| **Description**  File on the system are not digitally signed, therefore the integrity of the software cannot be guaranteed. This issue can be exploited to activate malicious program functionality by circumventing system and application integrity routines.  Reference :  https://en.wikipedia.org/wiki/Code\_signing  workitemimagea |
| **Affected URL/API(s)**  /usr/lpp/mmfs/bin/essctl |
| **Vulnerability Impact**  The system allows an attacker to modify executables related to system functionality. The program can be modified to subvert authentication checks or to add malicious functionality. |
| **Fix Recommendation / Mitigation**  1. Digitally sign the software to make it more difficult for an attacker to modify. As a best practice, sign both the installer and the main executable of the application.  2. If using digital signatures is not an option, use an MD5 signature and have the user check that the signature of the software they are about to load matches the signature expected. The expected signature should be computed by the release team after the new release is built.  3. The private key used to sign development releases, and the private key used to sign production releases should be completely separate and access to the production release private key should be highly restricted. |

[**195171: Sensitive Information Passed In CLI Arguments**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195171) ***(Informational)***

|  |
| --- |
| **Description**  Passing sensitive information as command line arguments poses a security issue as these information can be accessed by any user either through the bash history or by reading "/proc/<pid>/cmdline".  It is observed that sensitive information like "Username" and "Password" can be passed as command line arguments via essctl commands  Below are the related screenshot:  workitemimagea  workitemimagea |
| **Affected CLI/API(s)**   * /usr/lpp/mmfs/bin/essctl essmgmt copy-ssh-id --hostname utilityBareMetal --password cluster --username root --url utilityBareMetal --insecure-skip-tls-verify * /usr/lpp/mmfs/bin/essctl essmgmt copy-ssh-id --hostname emsvm --password cluster --username root --url utilityBareMetal --insecure-skip-tls-verify * /usr/lpp/mmfs/bin/essctl essrun config-load -n c145f15ess6k01a.gpfs.test -p cluster -b admin123 --url utilityBareMetal --insecure-skip-tls-verify * /usr/lpp/mmfs/bin/essctl essrun config-load -n c145f15ess6k01a.gpfs.test -p cluster -b admin123 --url utilityBareMetal --insecure-skip-tls-verify |
| **Vulnerability Impact**  A user can obtain sensitive information like username, password especially if the same command is reflected in the logs |
| **Fix Recommendation / Mitigation**  Disable/Remove passing sensitive information via CLI arguments.  Store credentials in environment variables and read them inside the application.  Use Configuration Files with Restricted Permissions  Interactive Prompt for sensitive content. |

[**195199: Password Returned In Response**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195199) ***(Informational)***

|  |
| --- |
| **Description**  APIs return passwords submitted in cleartext form in the responses. This behavior increases the risk of user's passwords captured by an attacker. Many types of vulnerabilities such as weaknesses in session handling, broken access controls, and cross-site scripting could enable an attacker to leverage this behavior to retrieve the passwords of other CLI users. This possibility typically exacerbates the impact of those other vulnerabilities, and in some situations can enable an attacker to quickly compromise the entire CLI/API. It is observed that the application sends passwords in the response.  workitemimagea |
| **Affected CLI/API(s)**  https://9.114.12.78:46443/scalemgmt/v3/operations  https://9.114.12.78:46443/scalemgmt/v3/operations/{id}  https://9.114.12.78:46443/sss/v3/config\_load  https://9.114.12.78:46443/sss/v3/configure\_gui  https://9.114.12.78:46443/sss/v3/copy\_ssh\_id  https://9.114.12.78:46443/sss/v3/run\_ess\_ssr\_setup  https://9.114.12.78:46443/sss/v3/set\_new\_password\_to\_emsvm23e |
| **Vulnerability Impact**  Vulnerabilities that result in the disclosure of users passwords can result in compromises that are extremely difficult to investigate due to obscured audit trails. Even if the CLI/API itself only handles non-sensitive information, exposing passwords puts users who have re-used their password elsewhere at risk. |
| **Fix Recommendation / Mitigation**  There is generally no valid reason for a CLI or API to return user passwords in its responses. If user impersonation is a business requirement, it should be implemented through a custom function with proper access controls and logging. Credentials must be redacted from all responses.  Reference: https://cwe.mitre.org/data/definitions/204.html |

[**195164: No Binary Obfuscation**](https://isljazzrtc.in.ibm.com:9552/ccm/web/projects/PTSS%20Projects#action=com.ibm.team.workitem.viewWorkItem&id=195164) ***(Informational)***

|  |
| --- |
| **Description**  Due to lack of Binary Protection/Obfuscation, application can be decompiled to gain knowledge about the programming technique, the interface and class definitions used, the open source algorithms and functions used etc. In this finding, all the details that can be obtained by decompiling the current application being tested.  workitemimagea |
| **Affected CLI/API(s)**  /usr/lpp/mmfs/bin/essctl |
| **Vulnerability Impact**  Binary inspection gives the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, business logic, cryptographic constants and ciphers, and intellectual property. This might aid in further attacks using dynamic instrumentation and reverse engineering. |
| **Fix Recommendation / Mitigation**  Obfuscate application source code.  Reference: <https://www.appsealing.com/code-obfuscation/> |

**Appendix A - Detailed Testing Methodology**

PTC penetration testing methodology is based on a structured approach applying industry best practices in testing. The below mentioned areas require different approaches and techniques and an overview of the general methodology is provided.

|  |  |
| --- | --- |
| **Area** | **Methodology Overview** |
| Web Application / API | PTC follows a customized pen test strategy based on OWASP Application Security Verification Standard ► Information Gathering  Gather scope via questionnaire, application demo by product team, review of threat Model, source scan and open source scan reports, CVE info etc. ► Reconnaissance  Fingerprinting server using banner grabbing techniques, HTTP request/response, analysis of cookies, HTML source code etc. ► Scanning and vulnerability assessment Automated scans using Burp Suite, OWASP ZAP etc., false positive analysis, identifying attack surface ► Manual testing and exploitation Extensive manual tests are performed in the following areas with additional focus on newly found vulnerabilities in the market  • Authentication, Authorization and Session Management  • Input Validation  • Error Handling  • Security configuration   • Client-side tests  • Business Logic ► Reporting Easy to consume executive summary and detailed findings report prepared by documenting the findings based on CVSS scoring along with risk assessment and fix recommendation, including steps that can be taken to mitigate each vulnerability. |
| Network | PTC follows a customized pen test strategy based on PTES and PCI Penetration Testing Guide  ► Information Gathering   • Gather scope of the target network and its infrastructure, IP ranges, network topology, and active hosts, systems and devices to be tested ► Reconnaissance   • Passive: ExploitDB / NVD search, DNS Fingerprinting etc  • Active: Banner grabbing using services such as FTP, Telnet, HTTP etc.   Firewall / IDS / IPS detection ► Scanning and enumeration  • Automated scans using tools such as nmap, Nessus etc., to identify open  ports, misconfigured services/OS fingerprinting  • Vulnerability assessments using Nmap scripts / Nessus plugins  ► Manual testing and exploitation  • Conduct manual testing to identify vulnerabilities that automated scanners may not detect  • Exploiting the vulnerabilities using Metasploit and manual techniques ► Reporting Easy to consume executive summary and detailed findings report prepared by documenting the findings based on CVSS scoring along with risk assessment and fix recommendation, including steps that can be taken to mitigate each vulnerability.  In addition to network tests, Segmentation and Container testing may also be required involving other specific tests.  Segmentation Testing  - Identify the segments that need to be tested. PCI classification of Network segments include CDE, Non-CDE and Out-of-scope segments.  - Perform UDP, TCP and ICMP scans from each segment to test for reachability  - Perform Segment bypass by crafting custom packets.  - Identify the non-compliant segments and report it.  Container Testing  - Identify the target container infrastructure, including the container runtime, orchestration tool, and underlying infrastructure.   - Container Image analysis: Analyze the container images to identify any vulnerabilities, including outdated software versions, unpatched vulnerabilities and other security issues.  - Container Runtime testing: memory leakages, namespace conflicts, container breakout scenarios etc.  - Container Orchestration testing: Test the container orchestration tool to identify any vulnerabilities, including misconfigurations, privilege escalation, and other security issues.   - Network tests: Test the container network configuration to identify any vulnerabilities, including misconfigurations, insecure communication channels, and other security issues.  - Reporting as mentioned earlier. |
| Mobile Application | PTC follows a customized pen test strategy based on OWASP Mobile Application Security Verification Standard ► Information Gathering  Gather scope via questionnaire, mobile application demo by product team, review of threat Model, source scan reports, CVE info etc.  ► Scanning and vulnerability assessment  Scan the application binary/installer using appropriate security scanning tools depending on feasibility, false positive analysis, identifying attack surface ► Manual testing and exploitation  Extensive manual tests are performed in the following areas with additional focus on newly found vulnerabilities in the market  • Authentication, Authorization and Session Management  • Input Validation   • Binary Analysis – insecure data Storage and Client code quality  • Security configuration   • Business Logic  • Testing SSL pinning - bypass SSL pinning and certificate checks  • Reverse-Engineering   - Code Obfuscation  - Decompiling and/or Disassembly  - Rebuilding with modified/malicious code  ► Reporting Easy to consume executive summary and detailed findings report prepared by documenting the findings based on CVSS scoring along with risk assessment and fix recommendation, including steps that can be taken to mitigate each vulnerability. |
| Thick Client | PTC follows a customized pen test strategy based on OWASP Thick Client Security Testing Guide ► Information Gathering Gather scope via questionnaire, thick client demo by product team, review of threat Model, source scan reports, CVE info etc. ► Scanning and vulnerability assessment Automated scans using Burp Suite, Zap etc., false positive analysis, identifying attack surface ► Manual testing and exploitation Extensive manual tests are performed in the following areas   • Authentication, Authorization and Session Management  • Input Validation  • Error Handling  • Binary Analysis – insecure data Storage, memory analysis   • Security configuration  • Business Logic ► Reporting Easy to consume executive summary and detailed findings report prepared by documenting the findings based on CVSS scoring along with risk assessment and fix recommendation, including steps that can be taken to mitigate each vulnerability. |
| Firmware | PTC follows a customized pen test strategy based on OWASP FSTM & OWASP ISVS methodology ► Information Gathering and reconnaissance Gather scope via questionnaire, acquire related technical and documentation details pertaining to the target device's firmware. ► Obtaining and Analyzing firmware Obtain firmware from vendor portal or other preferred ways and explore aspects of the file to identify its characteristics such as potential root filesystem metadata, and gain additional understanding of the platform it's compiled for.  ► Extracting and Analyzing filesystem contents  • This analysis typically involves reverse engineering the firmware to extract the file system, configuration files, and other critical components.  • Statically analyze extracted filesystem configuration files and binaries for  vulnerabilities. ► Emulating firmware  Emulating firmware files and components to verify potential vulnerabilities. ► Dynamic analysis Performing dynamic security testing against firmware and application interfaces in its normal or emulated environment. ► Runtime analysis Analyze compiled binaries during device runtime. Runtime analysis involves attaching to a running process or binary while a device is running in its normal or emulated environment. ► Binary Exploitation Exploit identified vulnerabilities discovered in previous stages to attain root and/or code execution to compromised firmware/devices.  ► Reporting Easy to consume executive summary and detailed findings report prepared by documenting the findings based on CVSS scoring along with risk assessment and fix recommendation, including steps that can be taken to mitigate each vulnerability. |

**Appendix B - Tools and Utilities**

Some of the tools used for pen test are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Web Application/API** | **Network** | **Mobile Application** | **Thick Client** | **Firmware** |
| ▪ BurpSuite   Professional  ▪ OWASP ZAP  ▪ Insomnia  ▪ SOAP UI  ▪ Curl  ▪ Sqlmap  ▪ Web developer  tools and browser extensions | ▪ nmap  ▪ Nessus  ▪ Metasploit  ▪ Wireshark  ▪ Kubehunter  ▪ Custom Scripts | ▪ Genymotion  ▪ APK Tool  ▪ JDGUI  ▪ MobSF  ▪ Frida | ▪ Procmon  ▪ SysInternal  ▪ Fiddler | ▪ Binwalk  ▪ Emba/Embark  ▪ Firmadyne  ▪ Ghidra  ▪ FactCore  ▪ Firmwalker  ▪ qemu  ▪ ByteSweep  ▪ Radare2  ▪ Custom Scripts |

**Appendix C**

Risk assessment evaluated by PTC is based on the number of Critical/High/Mediums identified along with the overall size of the product tested.

Risk Calculation Matrix:

