Rocket® software

Why You Need A Mainframe Operating System Vulnerability Scanning Methodology

Contents

What is the mainframe operating system?

The IBM® mainframe operating system is **IBM® z/OS®**. It is engineered for environments where **availability, transaction reliability, data, and system integrity are non-negotiable.**

In simple terms:

z/OS is the **mission-critical operating system** that keeps an organization's most essential financial and operational systems running securely and reliably, with full data integrity. z/OS runs the business's most critical operations. It supports core revenue, payment, settlement, and regulatory systems. A system integrity failure can directly threaten business continuity, customer trust, and financial stability.

What is system integrity and why does it matter?

z/OS system integrity means that only explicitly authorized programs can execute with elevated privileges, ensuring the operating system and everything that runs on it cannot be silently bypassed, corrupted, or controlled without detection.

Why this matters to the CFO, CIO, CEO, and Board of Directors

Mainframe operating system integrity matters to company executives because it underpins business resilience, regulatory compliance, and the effectiveness of every other security control protecting the organization's most critical data and operations.

- **It runs the business's most critical operations.** The application programs running on it support core revenue, payment, settlement, and regulatory systems. A system integrity vulnerability can directly threaten business continuity, customer trust, and financial stability.
- **Integrity failures caused by poor or malicious software coding techniques can bypass traditional security controls.** Weaknesses in programs at the operating system level allow attackers to circumvent identity management, access controls, monitoring, and logging, undermining investments the Board has already approved in cybersecurity.
- **It creates systemic, enterprise-wide risk.** A single integrity flaw in a program can expose all applications and data on the mainframe, enabling silent compromise, data manipulation, or prolonged undetected access with material impact.
- **It is a Board-accountable regulatory issue.** Regulations such as PCI DSS, DORA, APRA, and NYDFS 23 NYCRR 500 require governance over platform security, operational resilience, and demonstrable control effectiveness. Operating system integrity is foundational to meeting those obligations.
- **It affects fiduciary duty and executive accountability.** Boards are increasingly expected to demonstrate informed oversight of technology risks that could materially affect the organization. Visibility into mainframe integrity directly supports that responsibility.

8.4+

CVSS score of critical ALTER-level vulnerabilities can lead to complete compromise of the z/OS system and all data.

5.0+

CVSS score of severe READ-level vulnerabilities can expose passwords, encryption keys, and sensitive memory.

1

Vulnerability is all it takes A single OS-layer integrity flaw can bypass security controls and expose the entire platform.

4+

Major regulations impacted PCI DSS, DORA, APRA, and NYDFS require governance and operational resilience.

In short:

Regulations such as PCI DSS, DORA, APRA, and NYDFS 23 NYCRR 500 require governance over data at the platform security level, along with operational resilience, and demonstrable control effectiveness. Mainframe operating system integrity is foundational to meeting those obligations.

Boards are expected to demonstrate informed oversight of technology risks that could materially affect the organization. Visibility into mainframe integrity directly supports that responsibility.

System integrity is your responsibility

NOTE: The IBM Statement of Integrity (originally the MVS System Integrity Statement) is IBM's long-standing guarantee that its enterprise operating systems (like z/OS and z/VM) are designed to block unauthorized programs from bypassing system security.

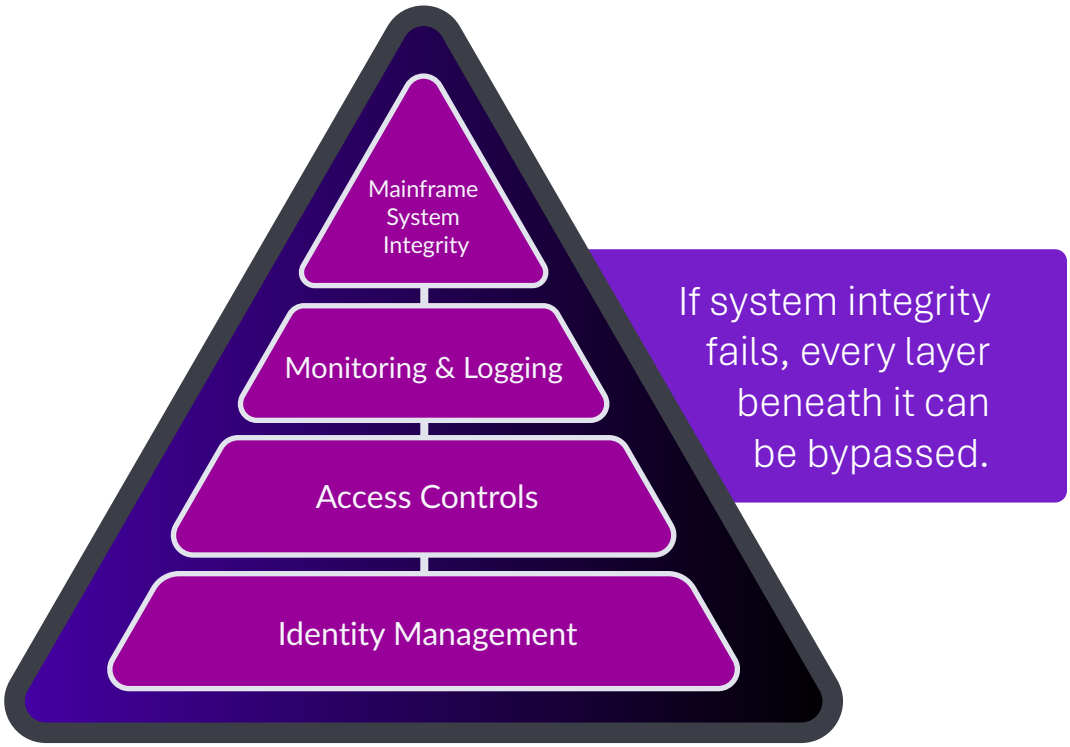
The IBM z/OS Statement of Integrity only applies to IBM software. It does not apply to any third-party vendor software or installation written code. As the z/OS system owner, you are responsible for continually verifying the integrity of any configuration-based modifications and code installed on your z/OS mainframe.

IBM's z/OS Authorized Assembler Services Guide makes this explicit. You, the customer, are responsible for ensuring that anything you install on each z/OS system you maintain meets the criteria of the IBM Statement of Integrity.

To ensure that system integrity remains in effect and that built-in controls are not compromised, you must take responsibility for:

- Physical environment of the computing system.
- Adoption of certain procedures (for example, the password protection of appropriate system data sets) that are a necessary complement to the integrity support within the operating system.
- That any modifications and additions (3rd party software, internally written exits) to the system do not introduce any integrity exposures.

All installation-supplied authorized code (for example, an installation SVC) must perform the same validity checking and control that the system uses to maintain its integrity.



Definition of integrity vulnerabilities

An integrity vulnerability is a weakness in the z/OS operating system layer caused by coding techniques that do not follow the IBM Statement of Integrity. The vulnerability could be in IBM software product-supplied programs, in independent software vendor (ISV)-supplied programs, or installation-added programs or exits.

Your exposure to integrity vulnerabilities

An integrity vulnerability occurs when an authorized program on your z/OS system violates the IBM Statement of Integrity. IBM's Statement of Integrity states, "IBM's commitment includes design and development practices intended to prevent unauthorized programs, subsystems, and users from bypassing z/OS security – that is, to prevent them from gaining access, circumventing, disabling, altering, or obtaining control of key z/OS system processes and resources unless allowed by the installation."

From definition to detection

Understanding these vulnerabilities is one thing. Identifying and prioritizing them across your environment is another. Without an automated and procedural approach, integrity vulnerabilities can go undetected — leaving your z/OS systems exposed to risks that are difficult to quantify and even harder to remediate. The good news is that knowing where to look is the first step toward meaningful protection, and that starts with having the right tool for the job.

What is z/Assure™ Vulnerability Analysis Program?

z/Assure Vulnerability Analysis Program (VAP) is a mainframe security and vulnerability scanning solution from Rocket Software. It is designed specifically for IBM z/OS mainframe environments to help organizations identify, analyze, and remediate integrity weaknesses before they can be exploited.

The z/Assure™ Vulnerability Analysis Program (VAP) categorizes integrity vulnerabilities as ALTER-level and READ-level. An ALTER-level integrity vulnerability is defined as a vulnerability that, when exploited, will completely compromise all data on your z/OS system, as well as the system itself. This is because the exploiter may change their authority to allow them to alter any security parameter and gain access to all data on the system. Integrity-based ALTER-level vulnerabilities will typically score at least 8.4 or higher using the CVSS V3 calculator.

A READ-level integrity vulnerability is defined as a vulnerability that, when exploited, will completely compromise some or all memory on your system. It is common practice to place sensitive data into fetch-protected memory. Data placed into fetch-protected memory includes clear-text passwords, encryption keys, and other similar sensitive data. This sensitive data could also include installation-defined sensitive data. Severe security code-based READ-level vulnerabilities will typically score at least 5.0 or higher using the CVSS V4 calculator.

An integrity code-based exploit is a set of instructions placed in an offending program by a bad actor that will enable an unauthorized z/OS user to bypass your installation's External Security Manager controls. It only takes one integrity code-based vulnerability in a z/OS layer program to give a bad actor the ability to bypass controls considered essential best practices for securing an application and the source data associated with the application. These vulnerabilities, when exploited, allow the exploiter full access to any data (including SMF records) and any data residing on that system.

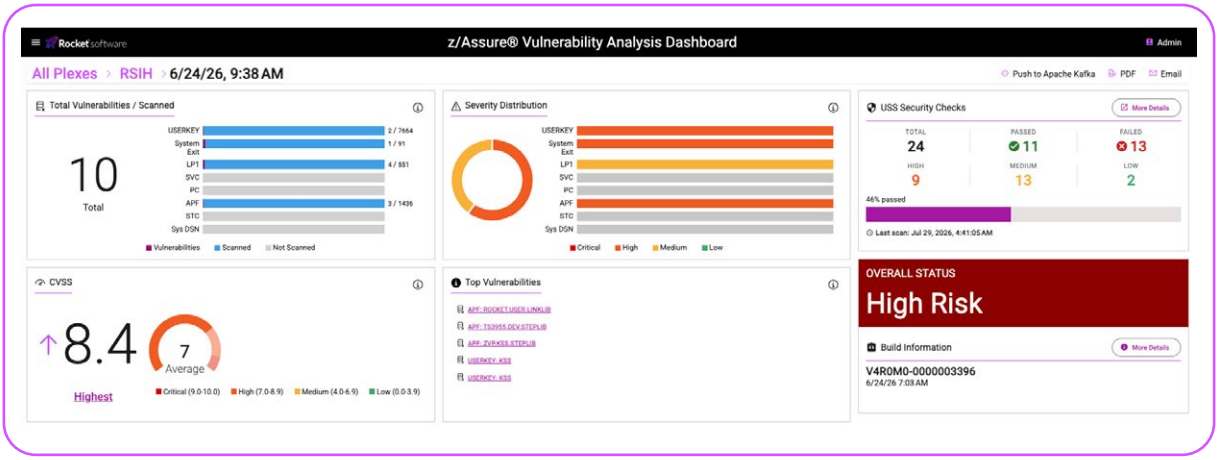
Note that External Security Managers (RACF®, CA ACF2™, and CA Top Secret™) are not part of the solution, nor are any application security testing tools or run-time application self-protection (RASP) tools. No current AI-driven threat detection software, ESM, or application security testing tool will identify or remediate these vulnerabilities. Ensuring system integrity is outside the scope of the current External Security Managers. The ESMs are not designed to enforce a security policy when a hacker (external or internal) uses an OS layer vulnerability to circumvent z/OS system integrity by escalating their security authority in memory to gain unauthorized access.

z/Assure® VAP currently scans the following:

- Supervisor call routines (SVCs) are used to define services in z/OS. Although the vast majority of these are supplied by IBM and Independent Software Vendors (ISVs), organizations can add SVCs to perform services for their own applications.
- Program call routines (PCs) are also used to define services in z/OS. Although the vast majority of these are supplied by IBM, and ISVs, organizations can add PC routines to perform services for their own applications.
- APF programs — link-edited as AC(1) and residing in an APF-authorized library — are extensions of the operating system.
- User key common storage in z/OS is an area of memory that is addressable to all address spaces. Each 4K memory address has a storage key associated with it. Storage keys 0-7 are considered system keys. Keys 8 and above are considered user keys.
- Identify programs (LP1) that are link-edited APF authorized and should not be.
- STC/storage-recovery vulnerability coverage. Above-the-bar storage allocation and recovery.

z/Assure VAP reports:

- Vulnerability Detail Reports (VDRs) document integrity code-based vulnerabilities that, if exploited, could be used to bypass the controls that z/OS and your External Security Manager have put in place. These reports are generated when VAP finds a vulnerability. There is one VDR per vulnerability.
- These reports should go directly to the vendor for mitigation, as they include the offset of the vulnerability in the listing. The vendor is responsible for providing a patch to fix the offending vulnerability. Once the patch is applied, VAP should be rerun against the offending program to verify mitigation has occurred.
- VAP does not abend or terminate an offending program. It is the responsibility of the customer to decide if the offending software should be removed or stopped. In many cases, the software is critical to the operation of the business. The CVSS score and categorization of a vulnerability provide the information required to decide on the next steps.
- Executive-level reporting (VARs) are PDFs that provide a consolidated view of the number of vulnerabilities found on the system being scanned. The VAR detail report includes the z/OS areas analyzed, CVSS score, and recognized categories such as trap door, storage alteration.
- A browser-based dashboard provides historical information for each scan and its results, including VDR summaries and drill-down information for each vulnerability found on each LPAR scanned.



Sample of categories of vulnerabilities

z/Assure VAP recognizes categories of mainframe integrity vulnerabilities. Common to all the listed categories is that vulnerabilities can be triggered or exploited by a non-authorized user with no special privileges.

Trap Door

Trap Door vulnerabilities provide a non-authorized user the ability to call a user provided routine in an authorized state. (Either system key (0-7) or supervisor state.)

This enables the user to directly make any changes to the active environment, including:

- Direct invocation of any authorized z/OS interface.
- Changing user's state (authorized).
- Making changes to the operating system.
- Making changes to system or application data.
- Impersonating other users.
- Disabling logging auditing (SMF).
- Disruption of z/OS operations (crash or failure).
- Trap Door vulnerabilities represent a serious compromise of the operating systems integrity, reliability, and the installation's Security Policies.
- Exploitation of a Trap Door vulnerability requires passing the address of a non-authorized routine to the vulnerable code. The vulnerable code then invokes the routine in an authorized state. Trap Door exploitation is the most severe vulnerability to z/OS integrity.

Storage Alteration

Storage Alteration vulnerabilities provide a non-authorized user with the ability to alter memory. While this is not as direct as a Trap Door vulnerability, it allows the manipulation of almost all virtual memory. By making changes to their environment, the non-authorized user can:

- Change user's state to be authorized (PSW supervisor state).
- Make changes to the operating system.
- Make changes to system or application data.
- Impersonate other users.
- Disable log auditing (SMF).
- Disrupt z/OS operations (crash or failure).
- Storage Alteration vulnerabilities represent a serious compromise of the operating system's integrity, reliability, and the installation's security policies.
- Storage Alteration vulnerability exploitation is easier to implement than a Trap Door vulnerability, as it does not directly grant total control. However, because any storage may be altered, the exploiter can use storage alteration to obtain total control, as well as corrupt data or cause an outage.

System Instability

System instability occurs when an authorized program is invoked in an unintended way and does not protect itself against improper invocation. This can cause z/OS service issues, including crashing the system.

Storage Reference

Storage Reference vulnerabilities allow a non-authorized user the ability to pass fetch1- protected storage to an authorized service (SVC or PC). This enables the non-authorized user to access and use data or parameters for which they are not authorized.

Identity Spoofing

Identity Spoofing vulnerabilities allow a non-authorized user to create alternate security credentials.

Security Probing

Security Probing vulnerabilities allow a non-authorized user to deconstruct security parameters without logging or tracking.

Least Privilege

Least Privilege vulnerabilities occur when a privilege or authority is assigned where a lessor privilege is appropriate.

About Rocket Software

Rocket Software is a global technology leader in modernization and a partner of choice that empowers the world's leading businesses on their modernization journeys, spanning core systems to the cloud. Trusted by over 12,500 customers and 750 partners, and with more than 3,200 global employees, Rocket Software enables customers to maximize their data, applications, and infrastructure to deliver critical services that power our modern world. Rocket Software is a privately held U.S. corporation headquartered in the Boston area with centers of excellence strategically located throughout North America, Europe, Asia and Australia. Rocket Software is a portfolio company of Bain Capital Private Equity.

Rocket® software

Modernization. **Without Disruption.**™

Visit RocketSoftware.com

MAR-18413_WP_VulnerabilityScanningMethodology_V2