SOFTWARE APPLICATION MONITORING AND MALWARE DETECTION (SAMM) FOR Z/OS

CONTENTS

Executive Summary

Introduction

History

Context and Mandates

Entropy

Malware

Ransomware

Persistent Threats

What's So Special About IBM Z and FIM?

Authorized Program Facility (APF), including LPA and potentially LLA

PARMLIB

JCL Libraries

CLIST, REXX, and other interpretive language program libraries

STEPLIBs and JOBLIBs

Source Libraries

SMP/E-Managed Libraries

Choosing Targets

Data Stores and Databases

Application and third-party non-SMP/E load libraries

FIM Strategies on IBM Z

Baselines and Integrity

Choosing Targets

Monitoring Mechanisms

What Customers Want

FIM and the Future

Future Regulations and Other Factors

Conclusion

Appendix A – FIM-Relevant Regulations and Frameworks

Executive Summary

In today's complex world of computing all organizations must deal with cybersecurity and regulatory compliance. One requirement spelled out in all cybersecurity frameworks and compliance baselines is what is called File Integrity Monitoring (FIM), a methodology that ensures the integrity of all software assets the computer system depends on. In the IBM Z mainframe world, one clear example of this is the Vanguard Integrity Professionals ("Vanguard") product known as SAMM (for Software Application Monitoring and Malware Detection).

FIM provides part of the solution, but its benefits are lost if there is no trusted source of truth to compare against. Software applications should exist on disk in the exact way that software vendors have written them in order to hold integrity. This must be true for all copies made of software applications across an enterprise with no exceptions.

When organizations rely on traditional maintenance operations and procedures they have often been dangerously complacent about monitoring and proving the integrity of critical programs and data. Finally, however, this critical security technology is available on the IBM Z mainframe.

FIM has become an essential aspect of monitoring and ensuring the integrity of data and systems of record. While other aspects of IBM Z have provided overlap coverage for this requirement, it's time for this practice to provide a greater insight into the status of system integrity.

Specific areas, implications, and approaches for FIM on IBM Z, which SAMM offers, are revealed and discussed.

Introduction

The spread of hacking and malware in non-mainframe platforms has led to the practice and technology for monitoring the integrity of critical programs and data files to catch and neutralize bad actors and their doings, such as ransomware. While the solidity of the IBM Z mainframe platform has allowed a sense of greater security, it is essential to get solutions that provide proof that security integrity is maintained.

As the system of record for the world economy, the mainframe is "where the money is", and therefore an ideal target for those who would modify critical programs and data on the mainframe for illicit gain. At the same time, modifying critical programs and data may also be a part of normal business operations. FIM is thus a necessary part of a complete security and integrity stack for this system of record, but it must be able to monitor all relevant changes and distinguish which are malicious.

"Talking about real control for your z/OS System Data sets, SAMM delivers that by intelligently tracking every critical change and reports on data sets, regardless of encryption status. If a modification doesn't match your verified SMP/E maintenance, SAMM steps in with an immediate alert which allows for users to check an LPAR/System and find out if PTFs applied to the execution libraries are being tracked. Plus, if you ever wished you could instantly check PTF [patch] application status across all your systems, no matter where your SMP/E data sets live, SAMM makes that a reality with its powerful search capability. It's about bringing true clarity and proactive security to your mainframe." – Brian Marshall, Chief Strategist

History

While much of computing security had its beginnings in the 1960's and 1970's, and was firmly in place and evolving on the IBM mainframe platform early on, certain threats didn't even become evident until the arrival of the internet era towards the end of the 20th century.

By 1993, the internet was already sufficiently widespread that academics were researching and responding to new threats from open access for both good and bad actors. One area where awareness had been sharpened was the vulnerability to unauthorized modification of critical programs and data. As a result, Gene H. Kim and Eugene H. Spafford wrote a Technical Report (CSD-TR-93-071) at Purdue University about the need and nature of File System Integrity Checking.

Over the following decades, this developed into a discipline known as File Integrity Monitoring, or FIM, as the battle lines between bad guys and protectors grew entrenched. Yet mainframe shops continued to mostly rely on other factors such as corporate firewalls and network security, traditional authentication and access management and logging on the mainframe, and sheer security by obscurity.

Such security is deceptive in a number of ways, the first being that you won't see what you're not looking for until it's far too late. Also, it ignores the fact that the mainframe has not been an isolated island ever since its first TCP/IP stack. On top of that are all the other security points of concern implicit in POSIX-compliant operating systems such as various UNIX and UNIX-like environments, including Unix System Services, originally introduced in 1993 as part of OpenEdition MVS which was first implemented in MVS/ESA SP 4.3.

Today we know, as IBM's Statement of Integrity makes clear, that while the mainframe may be the most securable platform, it is up to each organization to undertake the ongoing journey of maintaining and evolving that security in every relevant dimension. FIM is not only a necessary aspect of that, it is also available and in growing use, for example with SAMM.

+

Context and Mandates

FIM is about more than just pre-empting and detecting bad actors. Provable integrity has business value, especially to institutions responsible for financial, confidential, or personal data and processing. On top of that is an ever-growing list of business best practices, principles, rules, regulations, and laws in jurisdictions from local to global. Yet, in many ways, FIM began as something of an immunity-type after-the-fact response to a particular set of new threats.

If all critical data and programs could simply be hardened into ROM or absolute volumes, the security and integrity of data and processing would not be subject to such a range of exposures. But change is inevitable. Data must increase and be changed: that's what data does. Software must be patched to fix bugs and new security issues and to introduce new functionality. FIM solutions like SAMM are necessary.

Entropy

Call it what you will: Murphy's Law, human error, acts of God, fat fingers. Stuff happens – unplanned, unexpected, undesirable, unavoidable stuff. The most resilient of systems have availability measured in the nines, but even 99.999999% uptime is not 100%. And there are more dimensions than you can shake a measuring stick at. It makes no sense to assume things will always go as planned. It is essential to monitor and to identify when things do go sideways, and to have the ability to respond with resilience and restoration – i.e. FIM.

Malware

Viruses, Malicious Backdoors, Trojan Horses, Worms: Pandora's box of self-replicating malware is open, and there's no limit to what will come next. They all rely on the ability to bypass or modify security and then replicate themselves and travel to other environments. In so doing, they may "infect" system configurations and programs. They may also have more crafty payloads that lead to ransomware or other cyber attacks. Malicious attacks are now being carried out at very high levels of sophistication and span over long periods of time (e.g. XZ Utils backdoor from February 2024). Regardless, if critical programs, configuration, or other data are illicitly modified, catching and responding to that modification is essential – i.e. FIM.

Ransomware

Often following on from the above, or arising from similar attacks such as phishing, spear-phishing, social engineering, or any other hybrid or complex approaches, software can be infiltrated to encrypt or exfiltrate critical corporate data. Alerts must be triggered by the

modification of critical system configuration and software in order to activate the malware. Such alerts must invoke a response the moment the modifications are underway – i.e. FIM.

Persistent Threats

Malicious attacks on governments and organizations, that have been detected, are now being carried out gradually over a much longer time. A so-called *advanced persistent threat* (APT) involves an attacker spending significant time gaining trust and authorization to critical resources while remaining undetected. An APT attack would involve tampering with data or software assets in order to install malware and it is a serious concern. Often times, attackers will take additional time to camouflage their malware, involving many different pieces that individually do not appear malicious, but act together to create a backdoor. The need to prove software assets are authentic and have not been tampered with is critical for hardening the mainframe platform – i.e. FIM.

What's So Special About IBM Z and FIM?

The IBM Z mainframe has been the system of record for key data and processing in the global economy since it was first available, a role that appears to remain entrenched for the foreseeable future. While its long-standing presence offers some explanation for its stability, this platform also boasts unique security features. These features have successfully prevented many of the common exposures and incursions seen on other systems.

The key basis of these security features is the IBM z/OS® System Integrity Statement, first announced in 1973 for the MVS operating system. This guarantee of integrity is backed up with a range of features from authentication, authorization, and logging through memory isolation and supervisor state protection. But it only guarantees that the platform is securable, and leaves it up to individual customers to choose how they wish to properly take advantage of that securability.

Even though the mainframe is inherently secure, it can still have weak spots. These often come from how precisely and currently it's configured, secured, and updated.

Over time, new security issues that arise from imperfect configuration are commonplace, and need to be proactively dealt with, beginning with using a FIM solution such as SAMM. To clarify further, the following are some of the key data and processing resources on the mainframe which, even if protected using traditional security controls, still need to be actively monitored to avoid corruption that has processing, regulatory, and security implications.

Authorized Program Facility (APF), including LPA and potentially LLA

Ironically, one of the most important potential exposures on the mainframe is one of its key security dimensions: the control over the execution of privileged instructions using the Authorized Program Facility (APF).

Application programs and other user functions run in what is known as "problem state" and do not have access to operating system-level functions that could circumvent security measures. Only trusted operating systems and utility software should be allowed to run in the "supervisor state" which enables advanced functionality. The measures that limit access to this advanced state include only allowing programs (or "load modules") that are located in trusted data sets ("load libraries") to perform operating system type authorized behaviors. This is enforced using secure lists of trusted load libraries.

The definitive list of this nature is kept in the system parameter library, (often named "SYS1.PARMLIB") and can be changed real-time using system performance management software and console commands. It is also the case that load modules which are kept in memory at all times in what is known as the "Link Pack Area" (LPA) are authorized. There are additional load libraries which are defined to the system for fast retrieval of load modules using the Linked List (LLA), which can be set to have its programs authorized as well, though enabling this feature is not a security Best Practice.

In all of these cases, the contents of the parameter library must be tightly secured and monitored, as must the contents of each of the load libraries that can contain authorized load modules. Changes to any of these are significant security exposures if they are not approved and from a trusted source.

FIM (such as SAMM) monitors and alerts on all such changes.

What the industry says:

A prominent international bank, facing the complexities of regulations like DORA, GDPR, and PCI DSS, recognized a critical need - ensuring every PTF and vital file was installed correctly on their systems. Instead of hoping for the best, they proactively embraced Vanguard's technology designed to validate every single change to critical system areas. This strategic move not only solidified their compliance across global standards but also gave them confidence that their systems were always correctly configured, secure, and monitored.

PARMLIB

In addition to the above-mentioned lists of authorized load libraries, the system parameter library (again, often named "SYS1.PARMLIB") contains much of the other configuration information about how the z/OS platform operates. Along with other libraries that contain configuration information for sensitive systems, such information must be tightly controlled and

monitored as illicit or erroneous modification can result in significant losses and exposures. These text library data sets often can be updated by multiple user IDs, so tracking the occasions and nature of any updates must be undertaken externally to any one responsible individual.

This is exactly what a FIM does.

JCL Libraries

JCL, or Job Control Language, is the usual scripting and definition method for most of the tasks that run directly under the control of the mainframe operating system. Text library members that constitute JCL jobs and PROCs (i.e. procedures) are often stored in PROCLIBs (procedure libraries) which may be defined directly to the Job Entry Subsystem (JES) or other production or workload automation systems, or located using a PROCnn DD statement in a JCL job.

These JCL data set members are used to run systems and production programs which often modify sensitive data, and are consequently sensitive themselves. Any unauthorized modification to them could be a serious security incident. Therefore, it is important to monitor and alert on any such modifications.

In other words, FIM.

CLIST, REXX, and other interpretive language program libraries

Another text-based way that actions are performed on the IBM Z mainframe is using interpretive programming languages such as CLIST and REXX and a growing list of other such languages like Python.

Changing such programs is as simple as modifying the text library member that contains that program. A rogue modification of such a PDS member could have significant and immediate impacts with a wide range of functional areas, including system automation.

It is therefore essential to be notified of any possibly illicit modifications to such programs immediately, as the consequences are equally immediate.

FIM does this.

STEPLIBs and JOBLIBs

While application and user programs that only run in “problem state” aren’t able to invoke operating system functions, they do often have access to critical production data including financial information, personally identifiable information, corporate confidential information, and the whole range of sensitive business information.

The covert modification or replacement of such load modules can have immediate and devastating impacts on an organization's entire business. So it is essential to be notified immediately of any suspect changes to such programs in the libraries where they reside, often referred to in the JCL STEPLIB and JOBLIB statements.

This is an important use of FIM. A FIM solution such as SAMM will provide organizations with an automated method to monitor that the intended runtime libraries are in use and report/alert on any changes.

Source Libraries

COBOL isn't the only compiled programming language in use on the IBM Z mainframe, and there are other platforms where it runs as well. However, the pairing of this platform and language is a powerhouse of the world economy. Adding in the other compiled languages on the mainframe, from PL/I to C to numerous proprietary vendor languages, and all the text-based source versions of these programs amount to an essential resource. Without having a definite version of the source code for each compiled load module in use, it becomes dangerous to try to make any changes and recompile a program.

While source code managers are a normal solution to ensuring that the connection is tracked, the libraries of such programs (and their add-in portions such as copybooks with standardized lists of variables) make up a significant source of vulnerability to unauthorized modification, managed or not. And given the large number of user IDs that typically have access to them, tracking the source of a rogue modification can become a nightmare when something goes wrong with the compiled load module that should map to that source module.

A FIM like SAMM is the missing link to monitoring and tracking such changes and alerting immediately if something inappropriate may have occurred.

SMP/E-Managed Libraries

Today the distributed jargon term "patch management" is often applied to mainframe maintenance practices, though it is not always appreciated by experienced mainframers. Whether or not the term is accepted, when fixes are distributed to a system, it is important to detect immediately if bad actors have targeted the systems being maintained, whether they are IBM or Independent Software Vendor (ISV) products. Also bear in mind that Security/Integrity fixes are first received into SMP/E (System Modification Program/Extended) where they are then used to modify target and eventually distribution libraries. It is the organization's responsibility to ensure that these fixed versions are placed into all production libraries throughout the enterprise. This makes SMP/E libraries assets that must have integrity since copies of them are likely to be made.

Failure to monitor, track, alert on, and report on changes would significantly dislodge your environment from the safety zone. Authentic maintenance produced by SMP/E must be used by all production libraries. It is therefore imperative that any maintenance be recognized, but also that false positives be avoided if the contents of a given load module are in a different order when maintenance is applied compared to the baseline version but there are no additions, deletions, or changes.

As a proper mainframe FIM solution SAMM is able to distinguish between genuine and unwelcome modifications across all production and SMP/E libraries independent of a mere reordering of constituent sub-modules. It also can independently verify that members in target libraries match exactly what is produced by the applied PTF or APAR.

Data Stores and Databases

Why would someone commit a ransomware attack on a production environment such as the IBM Z mainframe? Because that's where the data is. Critical mainframe data sources such as proprietary tables are essential to the applications that use them.

When a normally static reference data source is unexpectedly modified, this must be immediately detected and then raise an alert if the data is in any way critical. And that's what a FIM such as SAMM can do.

Application and third-party non-SMP/E load libraries

Not every application or product in use in a mainframe environment uses standard installation and maintenance methods such as SMP/E. Regardless, if programs, load libraries, or configurations are in use, which could be modified by someone who is misusing authorization, the implications for your organization's integrity demand a FIM solution such as SAMM to respond immediately.

FIM Strategies on IBM Z

Clearly, there are many aspects of the IBM Z mainframe that need to be monitored by an advanced FIM solution such as SAMM – even some that are specifically about securing the integrity of that environment. There are also many factors and strategic approaches in establishing a fully-functional FIM solution in this environment, as follows.

Baselines and Integrity

Two central aspects of a FIM solution are, first, having a clear and independent, authoritative baseline picture of what the relevant data and programs in an uncorrupted system look like; and second, an ability to reliably detect what has been changed and when it has been changed in order to compare the result to the baseline.

There are some practical considerations here. It is necessary to do frequent and efficient compares of changes being made soon after changes are made to avoid serious damage. And it is necessary to minimize false positives by recognizing when a change is not significant, even if it reorders the sub-modules in a load module.

Based on these requirements, especially if the number of critical data sets and their total size are very large, an efficient and fully reliable formula is preferable over an exact mirror of the data in question. A method such as a signature or checksum using a recognized hashing algorithm can save on both baseline size and comparison time, as a rapid hashing of the changed data and comparison of signatures is much faster than a linear byte-by-byte comparison.

When considering the maintenance of executable load modules, it is essential that it propagates out to all libraries. It is necessary for a FIM solution to implement a reliable hashing method such that it can still recognize a load module is unmodified, even if constituent elements have been re-ordered. An efficient FIM solution that can do this would be capable of searching where a PTF or APAR presence is detected and where it is absent across all libraries. This would equip organizations with a tool that can reliably track the progress of deploying maintenance to production systems. Especially when the maintenance involves Security/Integrity PTFs and APARs.

For these reasons, an efficient formula approach that explicitly takes these matters into account is a much better method than a simple brute force comparison, and therefore the approach taken by SAMM.

Choosing Targets

When IBM introduced pervasive encryption to their Z mainframe, allowing any data to be stored encrypted and dynamically decrypted only when needed for use, it quickly became clear that many objects were not worth the resource costs of constant encryption and decryption. System files such as the programs that run the platform, for example, were not an efficient choice.

Not every load module or data source on the platform is sufficiently critical to receive the attention that a FIM solution can bring. But the criteria are somewhat different, because the implications of modifying critical programs and data are different from keeping them

confidential – and, in fact, for some critical system programs, it is a much preferable option to find out if they've been modified than to constantly decrypt them.

In both scenarios, a proper inventory of critical load libraries and data sources should be made, and recommendations and decisions should be carefully carried out. SMP/E and similar maintenance environments, especially for products that touch consequential data or employ APF-authorized programs, are examples of targets worth monitoring.

Monitoring Mechanisms

IBM's Z mainframe platform offers a wealth of ways to monitor changes, and the monarch of these would be the System Management Facilities, or SMF. Should you choose to enable the creation of SMF records every time a data set is changed, a strong FIM product can examine each event in real-time to determine if an alert is necessary. However, there is a certain amount of overhead involved in such a volume of real-time data monitoring.

Some sites avoid the creation of large volumes of SMF data, preferring to instead do regular checks on critical data sets to see if anything has changed, and investigate further when a modification has been uncovered. In this case, there is a trade-off between high frequency verification which uses more resources but can more rapidly detect and respond to illicit changes, versus lower frequency which is gentler on resource usage but may leave the window for bad actors to get a foothold open longer.

There is also the issue with Direct Access Storage Devices (DASD) shared across systems. A FIM product on the mainframe must be able to detect changes to load modules and text data set from other systems without relying on the SMF data on the system being monitored.

On top of platform-local considerations, there is also the matter of sending alerts and heartbeats to an enterprise SIEM for security operations awareness, monitoring, and sophisticated correlation and alerting. Hence a FIM such as SAMM must offer these features.

What Customers Want

Peace of mind and proof of integrity is the likely most honest answer a customer of a FIM solution would give when asked why such a thing matters to them. The moment it becomes clear that they don't have the resources to ensure that all of their critical programs and data are secured from any type of intentional or accidental corruption, it becomes clear that there is a need for an additional dimension of awareness to catch any violations of such integrity.

Of course, a solution that already exists, is easy to install and configure and use, and runs with low resource usage and high reliability is essential.

It is fortunate that FIM solutions such as SAMM now exist for the IBM Z mainframe platform, as they are no longer optional in order to meet the ever-increasing compliance requirements that have well-defined penalties.

FIM and the Future

FIM is about preserving the integrity on systems of record, very much including the IBM Z mainframe platform. The history of business computing to this point in history has made it clear that we often cannot predict future threats until we've experienced them, much like the human body's immune system response.

However, whatever future attacks look like, they will either modify critical data or processing, or take advantage of emergent weaknesses that must consequently be patched. In all of these cases, and in all the established matters of maintaining integrity for the systems of record, a FIM such as SAMM is now a permanent necessity. The future of FIM solutions must involve increased reliability in recognizing load modules, integration with maintenance operations and procedures, and efficient use of resources.

Future Regulations and Other Factors

On top of emergent security exposures, there will be unlimited additional regulations around the world, all of them in some way relevant to the sensitivity of data and processing. FIM solutions will continue to be critical as they evolve to respond to the future requirements that are still developing today.

What does the market say?

Facing strict compliance demands like 23 NYCRR 500, a major New York regional bank sought an aggressive solution. They found their answer in Vanguard's SAMM, a technology they now highly recommend for its effectiveness in meeting complex regulatory requirements for both internal and external audit requirements.

Conclusion

FIM is here for the IBM Z mainframe, and not a second too early. In the three-plus decades since FIM was envisioned for internet-connected computers, the IBM Z platform has become extremely integrated with the internet in many ways, and relying exclusively on traditional security is not enough.

As part of the ongoing journey of maintaining the integrity of the system of record for the world economy, a FIM like SAMM is now a required aspect of a full security program, ensuring that the

critical data and processing are not only protected but provably have their integrity proactively guarded and monitored.

Appendix A – FIM-Relevant Regulations and Frameworks

California Consumer Privacy Act (CCPA)

Center for Internet Security (CIS) benchmarks

Defense Information Systems Agency Security Technical Implementation Guides (DISA STIGs)

Digital Operational Resilience Act (DORA)

Federal Information Security Act (FISMA)

General Data Protection Regulation (GDPR)

> The GDPR protects the rights and freedom of data subjects, which includes defining the process/steps data holders must take to protect data. File Integrity Monitoring can be used to help become compliant with these GDPR-required Articles:
>
> Article 25: Data Protection by Design and Default
>
> Article 32: Security of Processing
>
> Article 39: Tasks of the Data Protection Officer (DPO)
>
> Article 57: Tasks
>
> Article 59: Activity Reports

NIST SP 800-66 / HIPAA (Healthcare Insurance Portability and Accountability Act)

> A File integrity monitoring tool allows businesses/organizations to not only achieve but also maintain compliance with HIPAA best practices, including the continuous evaluation of access controls and data security

New York State Department of Financial Services (NYDFS) Cybersecurity Regulation (NYC500 aka 23 NYCRR Part 500)

PCI DSS Version 4.0

> Payment Card Industry Data Security Standard
>
> 11.5.2: Deploy a change-detection monitoring (such as file integrity monitoring) to alert personnel to unauthorized modification of critical system files, configuration files, or content files, and configure the software to perform critical file comparisons at least once per week.

SOX / COBIT Framework

SOX (Sarbanes-Oxley Act) does not explicitly state the types of controls or methods organizations/businesses should use for compliance.

Due to this, the COBIT framework was established for compliance.

Standard of COBIT aided with the use of file integrity monitoring include:

- Acquisition and implementation
- Delivery and support
- Monitoring