March 2022

Integrating Solutions on IBM Z with Secure Service Container

Table of contents

Abstract

Today's IT solutions most often consist of many structural elements of which some essential ones are deployed as hardware appliances. The IBM Secure Service Container for IBM Z® provides the means to integrate and consolidate them into a secure, reliable and easy-to-maintain software container instead of in hardware. An appliance, in this case, is defined as a collection of operating system, middleware and software components that work autonomously and provides core services and infrastructure. Appliances can serve as part of a data center infrastructure (e.g. firewall), or as an addition to a service or solution stack (e.g. database accelerator). Appliances integrated into IBM Z with Secure Service Container are encrypted and signed for security and inherit the core reliability, performance and overall platform characteristics of IBM Z. These appliances are tamper proof, and only the appliance administrator has access to the appliance software. Not even the system administrator can access the appliance software unless specifically authorized. Also, the administration of appliances hosted in a Secure Service Container is consistent for all appliances hosted in that infrastructure. In this paper, we discuss the design, capabilities and attributes of Secure Service Container as well as the IBM Z firmware that supports it.

Introduction

Introduced in April 1964 as System/360™ [1], the IBM Z mainframe series is processing the world's most sensitive data for customers including but not limited to banks and insurance agencies. The platform was developed in parallel to the evolving financial industry and therefore provides a perfect fit for this market, due to its design for zero downtime and massive parallel transaction processing capabilities. In this environment, the Secure Service Container [2] (first announced as z Appliance Container Infrastructure [3]) provides an environment that consists of a tamper protected software boot process and ensures confidentiality of data and code running within the appliance – both while data is processed within the appliance and when it is stored on appliance external media. By design, the Secure Service Container environment can therefore reduce the risk of so-called insider threats [4]. Aside the security aspects, the Secure Service Container provides a deployment mechanism that lets solution providers deploy their products on IBM Z as easy as installing an app on a mobile device.

The IBM Secure Service Container is based on IBM Z firmware technology, that was introduced to implement so-called Firmware Platform Containers (FPCs) [5] also known as firmware partitions. These FPCs are hosting firmware functions that are separated into specific workloads, running on a Linux®-based operating system with the necessary communication channels within a logical partition (LPAR). Already existing firmware partitions on IBM Z comprise the peripheral component interconnect (PCI) support partition (PSP) [5] that utilizes Linux device drivers to manage PCI adapters. The master control services (MCS) partition [6] that hosts the dynamic partition management based on the operating system for IBM Z (z/OS®) binary of the hardware configuration manager wrapped to run under a Linux environment. This partition provides the backend of the dynamic partition manager (DPM) [6]. The network bootloader is used to boot an LPAR from a Linux boot volume via LAN (local area network). Firmware IBM Z Advanced Workload Analysis Reporter (IBM zAware) [7] is a self-contained firmware IT analytics solution that helps systems and operations professionals rapidly identify problematic messages and unusual system behavior in near real time. Implemented as a firmware appliance, zAware took advantage of the closed and well-defined container delivery, the downside was that it was restricted due to the dependency to hardware and firmware updates to stay current. Delivering updates in a software model instead of firmware was inevitable for a better life-cycle management. This is one of the key aspects of software appliances based on Secure Service Container technology. The first IBM-internal exploiters of the development and deployment model are the z/VSE® network appliance (VNA) [8], IBM zAware as software appliance and the IBM Blockchain offering for high security business network (HSBN) [9].

In this paper, we first discuss the deployment mechanism of the Secure Service Container, followed by security aspects. Furthermore, we describe the administration capabilities of appliances based on Secure Service Container technology.

Deployment

The foundation of the Secure Service Container infrastructure is a new LPAR type named SSC (Secure Service Container) [10]. Within the activation profile, LPARs can be configured to run in SSC mode. The main distinction with other LPAR modes are a special boot loader and an installer for Secure Service Container appliances. Both components are integrated and delivered with the IBM Z firmware. These capabilities enable the administrator of a Secure Service Container appliance to perform installation, deployment and life-cycle management of the appliance with almost no support from the administrator for IBM Z. In general, interaction with the system administrator for IBM Z is only needed for the first-time setup of an LPAR in SSC mode, its first activation and for disaster recovery. First-time setup defines the appliance administrator credentials, disk input/output (I/O) configuration and network configuration.

The network settings to connect to the browser-based graphical user interface (GUI) and the initial credentials to log on to this GUI are configured in the Image Activation Profile with the first-time setup of the LPAR in SSC mode. When activated for the very first time, the LPAR will come up with the Secure Service Container Installer started. Once logged on to the GUI, the designated appliance administrator can install an appliance in this LPAR. The installation process is designed for 'ease of use'. It does not require specific Secure Service Container knowledge nor IBM Z skills, operating system skills or middleware knowledge. The minimum inputs to install a Secure Service Container appliance are an appliance image and a media (volume assigned to LPAR) to install it on. Currently supported media for the installation are fibre connection direct-access storage devices (FICON® DASD) and fibre channel protocol (FCP) attached small computer system interface (SCSI) devices. The only limitation for the installation media is, that it must be big enough to hold the complete appliance image.

Secure Service Container appliances are provided as images, allowing for a fast installation. For example, with high network bandwidth, an installation of a 500-MB-sized Secure Service Container appliance image on a 3390 model 9 DASD is performed in less than 1 minute.

Through the Secure Service Container REST API (representational state transfer application programming interface), the whole installation process can be easily automated. Directly after the installation, the newly installed Secure Service Container appliance can be started. The running appliance hosts a web interface for infrastructure configuration purposes.

While the Secure Service Container appliance is up and running, the self-contained life-cycle management of the appliance is established through functions 'Reboot the appliance' and 'Boot to the Secure Service Container Installer' exposed via REST APIs and in the GUI. With a reboot to the Secure Service Container Installer, a new Secure Service Container appliance can be installed or a Secure Service Container appliance already installed on a preloaded FICON DASD or FCP volume can be migrated to this LPAR.

For more information about the life-cycle management in case the appliance encounters a problem, please refer to the 'Quality' section of this paper.

Security

The design of the IBM Secure Service Container targets superior security for all assets, that define an appliance instance. The central goal is to provide protection against disclosure and manipulation of customer data by anyone, but the respective owner of an asset. The Secure Service Container does this by leveraging a security focused design, utilizing state of the art security mechanisms and algorithms and backing them up with the extended hardware security of the IBM Z, including LinuxONE [11] offerings.

Security Roles

Security in the IBM Secure Service Container is built upon a clear distinction of roles and access rights/and methods to code and data. The respective roles are:

- IBM as provider of the Secure Service Container and the host machine.
- The vendor of the Service running inside the Secure Service Container.
- The administrator of an appliance instance [appadmin].
- The user of the service provided by the appliance.

Additionally, there are other roles, that are necessary for the operation of an appliance or might interact with it:

- The administrator of the IBM Z server (LinuxONE/host system) [sysadmin].
- Administrators of other infrastructure elements (e.g. storage or network).
- The rest of the world.

While the ultimate responsibility to secure an appliance in general and in particular in respect to the abovementioned rules is in the hands of the vendor, that builds the appliance, the Secure Service Container framework provides functionality and toolsets to make this task as easy as possible. As example, the most important isolation mechanisms and capabilities of a Secure Service Container appliance are described below very briefly.

Isolation
First, there is the isolation of the LPAR from the host system and other LPARs. This isolation can be compared to the isolation of different virtual machines and their host system. But in contrast to other widely used technologies, the LPAR isolation on IBM Z (LinuxONE) leverages separation mechanisms on hardware level and thus is certified with an evaluation assurance level (EAL)[12] of 5+.

Second, as it is possible in most other environments, IBM Z also offer multiple crash reconnaissance tools, that enable the sysadmin to get detailed information about the states and memory contents of a crashed virtual machine. Since this can be a problem in scenarios where the appadmin and the sysadmin are different people or even different organizations (e.g. in cloud/multi-tenant environments), IBM Z will refuse to disclose any information about the contents of an LPAR running a Secure Service Container, regardless of the privilege level of the requesting user. Instead an error inside of an appliance triggers the first failure data capture (FFDC) routines of the Secure Service Container and generates an encrypted report for the vendor, which only includes data relevant for debugging a problem, never touching any customer data.

Third, the Secure Service Container is isolated against different privileged and non-privileged individuals in the local network through various layers of encryption and signatures, that ensure, that no bit of data leaves the appliances memory without being encrypted. Furthermore, the appliances boot process up to the operating system (OS) kernel stage is made tamper proof by signature verification.

Finally, the appliance is isolated against the rest of the world by providing only well-defined REST interfaces over hypertext transfer protocol secure (HTTPS) via network port 443 for administration and application access and blocking any other communication attempts. This includes any operating system access, e.g. secure shell (SSH). To be operational with different kinds of services, appliance vendors can open ports, specifically needed for their appliances functionality. Vendors are responsible for securing these ports.

Administration
In a typical data center, many appliances, sourced from a variety of vendors, can be found integrated into the network. Each of these appliances typically has a different platform on which it is based with different hardware, operating system, system administration middleware and user interfaces (command line or GUI). For managing components which tend to be common across appliances such as storage, user management, and life-cycle management, the differences between these heterogeneous appliances adds to the cost of ownership. In this environment, each appliance requires unique administrative skills and processes.

Secure Service Container solves this problem in a variety of ways. First Secure Service Container is based on IBM Z LPAR technology. Within an LPAR, configuration of components such as memory, processors, and I/O adapters and devices, is managed in a common way through the hardware management console (HMC). In addition, I/O configuration can be done by the classical mainframe hardware configuration definition (HCD) and hardware configuration manager (HCM) tools or by the DPM.

Second Secure Service Container applications are already built with middleware pre-configured by the solution provider at the factory. This reduces the amount of configuration which is needed to deploy an appliance.

In addition, Secure Service Container provides administration mechanisms for components within the appliance infrastructure that are typically common across appliances. These include but are not limited to: Appliance status and management, user management and security, storage management, appliance configuration export and import (for life-cycle management), logging and serviceability, and network management. In each case, the administrative mechanisms are comprised of the corresponding component management middleware, a set of REST APIs to make the component functionality available from outside the appliance itself and a GUI for direct user interaction. The GUI also utilizes the REST APIs for communicating with the component. By providing these common administrative mechanisms, end user administrators of appliances that are based on Secure Service Container can leverage their administrative skills and processes across different appliances. This significantly lowers the cost of ownership of such appliances. Note also that by using Secure Service Container administration mechanisms, the end user administrator does not need to have any operating system specific knowledge. This makes administration easier to use and requires less training. Again, a savings in cost of ownership.

By providing a REST API for each administrative interface, Secure Service Container administration lends itself to data center automation. When administrative REST APIs are documented, IT professionals can programmatically utilize these REST APIs from the data center management infrastructure of their choice to automatically manage Secure Service Container appliances on IBM Z alongside the other systems in their data center.

Some appliances may choose to have a unique way of managing a component that fits better with their specific appliance functionality. The base Secure Service Container administrative mechanisms are offered as configurable building blocks that can be overridden by the appliance developer. For example, a network filtering appliance may choose their own style of managing networks and override the base Secure Service Container administrative mechanisms. It is the choice of the appliance developer which pieces of network management (GUI, REST API and underlying component management middleware) they choose to reuse or override. This flexibility allows for differentiation, where it is a value add to be different, and common mechanisms where it is not a value add to be different.

Quality

The Secure Service Container infrastructure was developed by the IBM Z firmware development team. The basic idea was to enhance the capabilities of a IBM Z mainframe through pluggable code components aka Secure Service Container appliances. For the implementation of a Secure Service Container appliance, no special knowledge about IBM Z is needed. But nevertheless, a Secure Service Container appliance will leverage the reliability, availability and serviceability (RAS), FFDC and security features, the IBM Z platform is known for.

One of the key RAS features of an IBM Z machine is that it tries to recover from problems encountered as much as possible and provide error reports for all problems encountered. This has to happen without any operator intervention and with no need for a reproduction of the error scenario to get more debug data. Being a closed system, Secure Service Container appliances do follow the same principle. Once an appliance recovers from a problem, the encrypted error data will be available for download on the web frontend of the Secure Service Container appliance. Several other systems may have the same capabilities. Secure Service Container appliances, however, do ensure the collection of debug data even in case of hang conditions or a complete crash of the appliance. This is accomplished with the help of a firmware watchdog, at which Secure Service Container appliances do register. If the appliance is not able to send a heartbeat signal to this watchdog in a certain time frame, the watchdog will disruptively dump and reboot the appliance. The encrypted dump data will be available for download over the web frontend respectively the REST APIs of the rebooted appliance. If the appliance is not able to boot, there is a fallback mechanism to retrieve the dump data via direct storage volume access.

With this approach, a Secure Service Container software appliance becomes a stand-alone system within an IBM Z machine with almost no intervention of the system administrator needed once it is configured and started for the very first time.

This also applies to the case of an appliance update. For Secure Service Container appliances, updates are delivered as complete appliance images. As a result, an appliance update always means to re-install the appliance. Compared to incremental updates via a packaging system, this approach offers some advantages for a closed Secure Service Container appliance. First of all, the appliance code installed is always consistent, approved and tested as-is. There is no open door for malicious code to get into the appliance via the update path. Also, this update mechanism allows to install the updated appliance onto a different disk. This has two advantages. First, the outage of the appliance is limited to the time it takes to reboot the appliance. Second, in case of problems encountered with the updated appliance, the old appliance is still available as backup. To have the updated appliance continue in the same state as the old appliance, the Export/Import feature of the Secure Service Container infrastructure can be used. While running the old version of the appliance code, the appliance configuration can be exported. After the installation of the new appliance version, the preserved appliance configuration can be imported to restore the appliance state.

All Secure Service Container appliances are built based on the same framework. This framework does provide a common code base. With the framework certified by IBM and also being tested by all vendors and customers together with their appliances, the test coverage and thus the quality of the appliance's code does fulfill highest standards.

Building an Appliance

As described above the Secure Service Container framework is a base environment for software solutions like zAware V3.1, VNA or IBM Blockchain HSBN to provide a secure appliance for IBM Z. In addition to the previously described values the framework adds to these software solutions, a build capability is coming with that framework. Although the framework is today only available for IBM-internal software solutions, IBM is working towards an enablement of independent software vendors (ISVs) to use this framework to build own solutions. Please find information about releases here: [13].

The Secure Service Container framework provides a build and packaging infrastructure to the solution provider. This infrastructure enables the solution provider to plug the software solution into the Secure Service Container framework and create the appliance image. The framework is providing the base operating system. Additionally, the Secure Service Container framework provides a build environment which coordinates the build process. This is done with the help of scripts and configuration files. The result of the build process is an appliance image which can be deployed. Before building the appliance image, the solution provider needs to add the software code he wants to be part of the appliance. In addition, he needs to connect the application to the interfaces and modules provided by the Secure Service Container. Some of them are FFDC, network and user management as described previously.

The source for the software solution can be plugged into that framework in 3 different ways. The solution provider can provide source code. The Secure Service Container supports most of the standard programming and Script languages as C/C++, Python, bash scripts, Dojo toolkit etc. With help of pre-defined build scripts, the added source code is integrated into a binary format which than is plugged into the base operating system. A second possibility for the solution provider is to already provide binary packages of his solution. The binary packages will be integrated into the framework with help of configuration scripts and so be part of the final appliance throughout the build process. The third possibility for a solution provider is to plug in a Docker container. The software solution is already pre-packaged as a Docker container and plugged into the Secure Service Container framework. The solution provider needs to provide additional management scripts to coordinate the start, stop and management of the Docker container by them self.

After the Source code is added to the framework, the solution providers can customize the layout of their appliance. The solution provider chooses the management capabilities for the intended usage scenarios. The solution provider can add own REST API/UI capabilities for special purposes and can register the corresponding components to interfaces like FFDC and boot process. When the final appliance build is triggered, the Secure Service Container framework packages together the whole solution to a binary format which can be deployed to an SSC type LPAR.

Conclusion

In the last few years, the world has become increasingly digitized in terms of the growing amount of data being generated, stored, and communicated. While in the beginning there was a focus on increasing performance and capacity with decreasing size of information technology (IT) solutions today, the security becomes increasingly into focus. Customers of IT products care about their data and care about privacy. While in the past, attacks from outside of a company were of paramount concern, today insider attacks continue to grow in importance. With increasing the usage of IT technologies in our daily lives we need to be prepared to guarantee protection against such threads.

The IBM Secure Service Container is a solution focusing on security combined with an easy-to-use and maintain solution enablement. To achieve this, Secure Service Container combines different technologies and makes use of them in a specific way. IBMs IBM Z provides a strong virtualization model combined with a certified isolation level (EAL5+), which IBM Secure Service Container utilizes. In addition, solutions like blockchain based on Secure Service Container can benefit from Federal Information Processing Standard

(FIPS) 140-2 certified cryptographic hardware, provided by IBM Z. In addition to this hardware exploitation, Secure Service Container also combines standardized packaging and virtualization solutions like kernel-based virtual machine (KVM) and Docker to provide an option space for software solutions to fit into that environment. To change the focus to be more on the solution then on the management, complex operating system activities are abstracted. While the solutions are inside a protected and restricted runtime environment, it is very important that they benefit from the decades of experience of our technologists with IBM Z first failure data capture.

References

1. G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, "Architecture of the IBM System/360", *IBM Journal of Research and Development*, vol. 8, issue 2, pp. 87–101, 1964.2. IBM Knowledge Center - IBM Secure Service Container (Secure Service Container). [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/HW11R/com.ibm.hwmca.kc_se.doc/introductiontotheconsole/wn2131zaci.html
2. IBM Knowledge Center -IBM Secure Service Container (Secure Service Container). [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/HW11R/com.ibm.hwmca.kc_se.doc/introductiontotheconsole/wn2131zaci.html
3. Expanding the IBM Systems' portfolio with additions to IBM Z and IBM LinuxONE; IBM Europe, Middle East, and Africa Hardware Announcement ZG16-0002. [Online]. Available: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&supplier=877&letternum=ENUSZG16-0002
4. 2017 IBM X-Force Threat Intelligence Index. [Online]. Available: https://www.ibm.com/security/data-breach/threat-intelligence-index.html
5. C. Axnix, G. Bayer, H. Böhm, J. von Buttlar, M. S. Farrell, L. Cranton Heller, J. P. Kubala, S. E. Lederer, R. Mansell, A. Nuñez Mencias and S. Usenbinz, "IBM z13 firmware innovations for simultaneous multithreading and I/O virtualization", *IBM Journal of Research and Development*, vol. 59, issue 4/5, pp. 11:2-11:6, 2015

6. O. Lascu, B. Sannerud, C. A. De Leon, E. Hoogerbrug, E. Palacio, F. Pinto, J. J. Yang, J. P. Troy, M. Soellig, *IBM z13s Technical Guide*, pp. 503-504. IBM Redbooks®, 2016.
7. IBM zAware – IBM Offering Information. [Online]. Available: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=ZSL03184USEN
8. IBM: z/VSE – Products and components – VSE components – VSE Connectors. [Online]. Available: https://www-03.ibm.com/systems/z/os/zvse/products/connectors.html#vna
9. IBM Blockchain – Blockchain Solutions. [Online]. Available: https://www.ibm.com/blockchain/offerings.html
10. z Systems Secure Service Container User's Guide. [Online]. Available: http://www-01.ibm.com/support/docview.wss?uid=isg2bb79df265313634d85258088005188e3
11. IBM LinuxONE. [Online]. Available: https://www-03.ibm.com/systems/linuxone/
12. Common Criteria Ensurance Levels - NSCS Site. [Online]. Available: https://www.ncsc.gov.uk/articles/common-criteria-assurance-levels

Bios

Volker Bönisch *IBM Systems, IBM Deutschland Research and Development GmbH, 71032 Böblingen, Germany (volker.boenisch@de.ibm.com).* Mr. Bönisch is a user interface/user experience (UI/UX) designer in the Enterprise Platform Development department at IBM Research and Development. He received a Diploma in Computer Science from the Baden-Württemberg Corporate State University Stuttgart in 2001. He subsequently joined IBM where he has worked as software developer in the financial business sector in Software Group. In 2015, he changed to IBM Systems as UI/UX designer. He is author or co-author of 5 patents and 1 technical paper.

John Lee *IBM Silicon Valley Lab, 555 Bailey Ave, San Jose, CA 95141-1003, United States (john.lee@us.ibm.com)*. Mr. Lee is a long-time Unix/Linux expert from the Silicon Valley area. After spending time at Sun Microsystems as a director of Solaris OS Filesystems and Networking, Mr. Lee was Vice President at Platform Solutions Inc. before its acquisition by IBM. At IBM Mr. Lee has been working on I/O configuration and I/O virtualization and has recently been the verification lead for IBM Secure Service Container. Mr. Lee's focus is in bringing new workloads to IBM Z.

Stefan Roscher *IBM Systems, IBM Germany Research and Development GmbH, 71032 Boeblingen, Germany (stefan.roscher@de.ibm.com).*

Mr. Roscher is a UNIX/Linux expert from IBM Germany Development Lab. He received a Diploma in Applied Computer Science from the Baden-Württemberg Corporate State University Stuttgart in 2006. He subsequently joined IBM Z and was involved in the development of Linux device drivers for Ethernet and high-performance computing (HPC) adapters (Infiniband®). After some years of fiber channel over Ethernet (FCoE) development for switch technologies and fibre connection direct-access storage devices (FICON DASD) development for simulation environments he started to work on the IBM Secure Service Container.

As architectural lead, he is focusing on developing the framework and to bring new workloads to the IBM Z environment. He is author or co-author of 7 patents.

Jochen Schweflinghaus *IBM Systems, IBM Deutschland Research and Development GmbH, 71032 Böblingen, Germany (schwefel@de.ibm.com)*. Mr. Schweflinghaus did graduate at university 'Fridericiana' (TH) in Karlsruhe and joined IBM in 1997. Starting as a test engineer for IBM Z architecture and firmware Mr Schweflinghaus became an expert for boot loaders for IBM Z. As an early member of the Secure Service Container team, he signs responsible as author or co-author for 6 patents and 1 paper.

Dennis Zeisberg *IBM Systems, IBM Deutschland Research and Development GmbH, 71032 Böblingen, Germany (dennis.zeisberg@de.ibm.com).* Mr. Zeisberg is a Security Engineer in the Enterprise Platform Development department at IBM Research and Development, working on the Security Architecture and Secure Engineering topics within the Secure Service Container Team. He finished his Apprenticeship as Systems Integration Specialist in 2013 and received his B.E. degree in Electrical Engineering in 2014 from South Westphalia University of Applied Sciences in a dual model. He subsequently joined IBM as intern and master's degree candidate and graduated with an M.E. in Information and Communication Systems with Management from South Westphalia University of Applied Sciences in 2017.