Confidential Computing with Secure Execution (IBM Z)

Jakob Naucke
Cloud Development for Linux and OpenShift on IBM Z & LinuxONE
# IBM Secure Execution for Linux

| IBM Z & LinuxONE/s390x/“mainframe” used for Red Hat OpenShift workloads | Hardware confidential computing support since z15 (September 2019) & LinuxONE III | Necessarily based on Linux KVM virtualization | Other virtualized confidential computing technologies include IBM Power’s PEF, AMD SEV, and Intel TDX |
How do you know your workload runs in a secured context?

This can be achieved through attestation and smaller encrypted key containers.

Secure Execution (like PEF) relies on a fully encrypted boot image that can house anything.

The asymmetric key is tied to the machine and can be verified through a certificate authority.

But how can the machine retrieve the private key for decryption? If the hypervisor could simply read it, you haven’t gained anything.

$ ssh my-secure-domain
Hi, this is the motd from your cloud provider! I am totally running this inside Secure Execution!
Enter the Ultravisor

- QEMU
- Guest Memory
- KVM
- Linux
- Secure Guest State
- UV
- HW/FW
- Other Linux Processes
The ultravisor is implemented in hardware and firmware. It communicates with the hypervisor, but the hypervisor cannot read registers or memory.

Only the ultravisor can decrypt the encrypted boot image. When booting a Secure Execution guest, the guest is immediately rebooted with a call to the ultravisor.

Pages are encrypted by the ultravisor when being swapped out, and checked for integrity when being swapped in. I/O and scheduling remains handled by the hypervisor.

Some things not (yet) supported: Memory encryption rather than protection, hypervisor initialized dump, migration/saving, VFIO, huge pages, ballooning.
“Classical” Secure Execution

...but what if you want containers?

Linux rootfs
traditional LUKS
on trusted s390x hardware

Linux rootfs
Secure Execution, possibly untrusted hardware
Enter Kata Containers
“The speed of containers, the security of VMs”
How do you achieve confidential computing with Kata Containers?

Utilize hardware. Lock the agent.

As a first, basic solution, we can put anything we might want to use into a custom, encrypted image.

This image is pulled upon creating a container.

Where is the key to decrypt it?
Integrating the current Secure Execution workflow with the Attestation Agent

“Bake-in” approach
Integrate the keys to decrypt image layers

Simple, but inflexible

“Key fetch” approach
Classical authentication

(Somewhat) more flexible
TLS as substitute for runtime attestation
## Development

<table>
<thead>
<tr>
<th>Hardware enablement</th>
<th>Bake-in approach: Sharing work with SEV</th>
<th>Key fetch approach: Integrating with Attestation Agent, using gRPC over TLS</th>
<th>Finding ways to avoid building full guest images on trusted IBM Z hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>(done)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Thank you

Jakob Naucke
Cloud Development for Linux and OpenShift on IBM Z & LinuxONE
—
jakob.naucke@ibm.com
github.com/Jakob-Naucke
ibm.com