Xen for Embedded, IoT, Edge

Stefano Stabellini
Xen Maintainer, Principal System Software Engineer
2018/09
Lightning Xen Update
Very Embedded Requirements

> Real Time
  >> Low Deterministic IRQ Latency
  >> Static Partitioning
  >> Real Time Schedulers

> Short Boot Times

> Device Virtualization
  >> Device Assignment
  >> Device Sharing
  >> Driver Domains
  >> VM to VM communications

> Certifications
  >> Small Code Base
  >> Type-1
Static Partitioning

sched=null vwfi=native
Static Partitioning

sched=null vwfi=native

2.5 us
VM to VM communication mechanisms

- **Libvchan**
  - Linux Library
  - Direct VM to VM channel based on a ring on shared memory
  - `libxenvchan_send` and `libxenvchan_recv`

- **PVCalls**
  - Socket API virtualization
  - VM to VM communications mediated by the backend domain (dom0)
  - “lo” as an inter-VMs communication namespace

- **V4V**
  - Linux library and hypercall
  - VM to VM communication mediated by Xen
  - Trivial to implement in your kernel
  - Not fully upstream
Shared Memory

- Completely Configurable
  - Support any memory attributes including cacheable memory
- No need for Xen support to use it
- Can export the memory to Linux userspace and use OpenAMP

```plaintext
static_shm = ["id=ID1, begin=0x40000000, size=0x1000, role=master"]
static_shm = ["id=ID1, offset=0, begin=0x48000000, size=0x1000, role=slave"]
```
Reducing Code Size

```
make cloc

cloc --list-file=/tmp/tmp.1L2EdV9dL
   143 text files.
   143 unique files.
   0 files ignored.

http://cloc.sourceforge.net v 1.60  T=0.26 s (546.4 files/s, 262525.6 lines/s)

<table>
<thead>
<tr>
<th>Language</th>
<th>files</th>
<th>blank</th>
<th>comment</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>126</td>
<td>10527</td>
<td>10408</td>
<td>45144</td>
</tr>
<tr>
<td>Assembly</td>
<td>17</td>
<td>249</td>
<td>937</td>
<td>1439</td>
</tr>
</tbody>
</table>

| SUM:          | 143   | 10776  | 11345   | 46583 |

rm /tmp/tmp.1L2EdV9dL
```

© Copyright 2018 Xilinx
Certifications

Make xen.git certifiable:
- Reduce code size
- Fix compliance violations reported by PRQA

Ideas on how to do certifications in a Xen Project (Linux Foundation) context
Dom0-less

U-Boot

loads into memory

Xen

loads into memory

Dom0 / DomU

DomU 1

DomU 2

CPU0

CPU1

CPU2
**Dom0-less**

The diagram illustrates the boot process in a Dom0-less environment using Xen. The boot process starts with U-Boot, which then boots Xen. Xen, in turn, boots Dom0/DomU, which further boots DomU1 and DomU2, each being assigned to a CPU (CPU0, CPU1, and CPU2).
Secure Containers at the Edge
The Problem

- **Package applications for the target**
  - Contain all dependencies
  - Easy to update
  - Independent lifecycle

- **Run applications on the target**
  - Run in isolation
  - No interference between applications
The Problem

- **Package applications for the target**
  - Contain all dependencies
  - Easy to update
  - Independent lifecycle

- **Run applications on the target**
  - Run in isolation
  - No interference between applications
The Problem

- **Package applications for the target**
  - Contain all dependencies
  - Easy to update
  - Independent lifecycle

- **Run applications**
  - Run in isolation
  - No interference between
Packaging vs. Runtime

OCI Image Spec vs. OCI Runtime Spec
The problem with Linux namespaces
Malicious App

Cloud-native App

Cloud-native App

Large surface of attack

On average, 3 privilege escalation vulnerabilities per Linux release!
Large surface of attack

On average, 3 privilege escalation vulnerabilities per Linux release!
Large surface of attack

On average, 3 privilege escalation vulnerabilities per Linux release!
Security hardening techniques

From “Understanding and Hardening Linux Containers” by NCC Group:

• Run unprivileged containers (user namespaces, root capability, dropping)
• Apply a Mandatory Access Control system, such as SELinux
• Build a custom kernel binary with as few modules as possible
• Apply sysctl hardening
• Apply disk and storage limits
• Control device access and limit resource usage with cgroups
• Drop any capabilities which are not required for the application within the container

[...]
Security hardening techniques

[...] 
- Use custom mount options to increase defense in depth
- Apply GRSecurity and PAX patches to Linux
- Reduce Linux attack surface with Seccomp-bpf
- Isolate containers based on trust and exposure
- Logging, auditing and monitoring is important for container deployment
- Use hardware virtualization along application trust zones
Securing Linux Namespaces is possible but very difficult
- It requires specific knowledge of the cloud native app
- Auditing and monitoring should be performed everywhere

Using virtualization for isolation is still recommended
Linux Namespaces: very embedded problems

- **Mixed-criticality** is not supported
- **Limits** on resource utilization are hard to enforce
- **Real-Time** support is difficult
- **Certifications** are very difficult
The Solution: Xen as Container Runtime

Security, Isolation and Partitioning
  > Multi-Tenancy
  > Mixed-Criticality Workloads

Hardware Access to Applications

Real-Time Support

ViryaOS: a ready-to-use runtime environment for VMs and Secure Containers
The Problem #2

>Cross-building multiple VMs is difficult

>Assembling the output in a single runnable image is a manual process
Typically users know all the VMs they need beforehand

They still need to:
- Build them all, plus Xen and Dom0
- Install all images on target
- Partition the hardware using device assignment
  - Edit the Dom0 device tree
  - Generate appropriate device trees for DomUs with device nodes
- Plan for images upgrades and security fixes
It’s a lot of work!
You think this is bad enough...

...then you try disaggregation
Current Status

Everybody has their own scripts and handcrafted solutions
  - They are limited
  - Only target one use-case
  - Limited support for driver domains and service domains
  - Only support one hardware platform

We would all benefit from a unifying effort
ViryaOS

A proposal for a new Xen Project sub-project
> Secure Xen based runtime
> Containers supported natively
> a turnkey solution
> a Flexible build system
> support aarch64 and x86_64
> Targeted at embedded and IoT
ViryaOS

BUILD

ViryaOS Build

RUNTIME

App

SD
ViryaOS: Runtime

> Dynamically deploy VMs and Secure Containers

> Containers are run securely, transparently as Xen VMs
  >> 1 Kubernetes Pod per VM
  >> See KataContainers and stage1-xen

> Measure Boot

> System Software updates and Containers updates

> Uses Disaggregation, Service Domains, and Driver Domains
ViryaOS: Runtime

VIRYA OS

Hardware

Xen

Dom0

Secure Containers Runtime

Containers/ DomUs
Dom0

Domain Manager

Secure Containers Runtime

Internal API

DomUs

Xen

Hardware
Secure Containers Runtime

Internal API

Dom0

Domain Manager

TPM Manager

Network Manager

DomUs

Xen

Hardware

TPM

NIC

XILINX
ViryaOS: Build

- A multi-domain build system
  - Builds multiple domains in one go
- Create a runnable SD Card image from multiple domain builds
- Each domain build is independent and run in a Container
- Pre-configures device assignments to VMs
- Made for disaggregated architectures
ViryaOS: Build

- Dom0 - Alpine Linux
- Dom0 kernel - Yocto
- Dom1 DriverD - Yocto
- Dom2 DriverD - Alpine Linux
- Image Builder
- apks
- Image.gz
- Rootfs
- apks
- SD Card Image
ViryaOS: Build

- Everything builds in a Container
- Support cross-builds (aarch64 on x86) with qemu-user
- Support any build systems for domain builds
  - Enable mixed Alpine Linux / Yocto environments
  - Rootfs and kernel can be built independently
- Support multiple DomU build output formats
- The DomU build output is stored in a container
  - Intermediate artifacts can be pulled from the Docker Hub to speed up the build
Status

> Very early stage, experimental
> Interest, but no company backers yet, community driver
> Subscribe to the mailing list to learn more and participate!

> Initial implementation available for:
  >> SDK
  >> Containers-driven build
  >> Yocto kernel build
  >> Imagebuilder
Adaptable. Intelligent.

sstabellini@kernel.org
stefanos@xilinx.com