OpenXT Platform

Ross Philipson
Assured Information Security
OpenXT Summit, June 7th 2016

Brief History

- I have been involved in this project since its inception in the winter of 2008.
- First there was Citrix XenClient - a general purpose client virtualization product.
- Then there was Citrix XenClient XT - an offshoot of XenClient focusing on high security use cases.
- In June of 2014 we open source XenClient XT as OpenXT.
- Today OpenXT is the only project that is still maintained.
- But OpenXT is still fundamentally the XenClient XT product.

The OpenXT “Product”

- Underlying platform buried in OpenXT that contains:
  - Core components: TBOOT, Xen hypervisor, dom0 kernel, device domains, installers, etc.
  - Core security features: measured launch, XSM, SELinux, RO fs, device model isolation, etc.
  - Operational components: toolstack, QEMU, para-virtual device drivers, guest tools, v4v, etc.
  - Build system: OpenEmbedded, custom recipes and scripts, etc.
- High level management components:
  - Xen Manager for overall management and business logic of the product.
  - UIVM user interface VM for control and configuration of the product.
- Graphical display and user input components:
  - Surfman display surface management and switching.
  - Input Server to handle user input routing to guests.
- Synchronizer backend server and client VM.

An OpenXT Platform

- This is a proposal for creating an OpenXT platform.
- What we want OpenXT to be is an extensible, stable and highly secure client virtualization platform.
- We believe the way forward is to first define a minimum base platform layer that provides the features and security attributes that are essential.
- Higher layers can then be built on top of the base platform layer, either as part of the OpenXT project or as independent projects.
- What would be in the base platform?

OpenXT Platform Base Layer

- Start with all the underlying platform bits from slide 3:
  - Core components: TBOOT, Xen hypervisor, dom0 kernel, device domains, installers, etc.
  - Core security features: measured launch, XSM, SELinux, RO fs, device model isolation, etc.
  - Operational components: toolstack, QEMU, para-virtual device drivers, guest tools, v4v, etc.
  - Build system: OpenEmbedded, custom recipes and scripts, etc.
- Construct a stable and extensible API to configure and control the base layer.
- Define a tightly controlled inter-domain RPC protocol.
- Headless operation:
  - Configuration and control done use command line in local terminal or remote shell.
  - Scripts to simplify configuration and control.

Not In The Base Platform Layer

- There is no graphics control stack or input handling.
- The Synchronizer components would be moved out.
- No high level management component or business logic (Xen Manager).
- No graphical UI configuration and control (UIVM).
- Note that “not in the base” does not mean “not in OpenXT”.
- Higher level layers...

OpenXT Platform Optional Layers

- The best approach to extend the OpenXT base is with OpenEmbedded layers.
- OpenXT today is itself a layer on OpenEmbedded as will be the base layer.
- Layers can override portions of layers below them as well as extend them.
- As an example, given what exists today, OpenXT could be reorganized in layers:
  - OpenXT base layer as described earlier.
  - OpenXT graphical layer on top of the base layer (with Xen Manager, UIVM, surfman, input handling).
- Or consider the Synchronizer as a layer:
  - OpenXT layer to contain the client side portions that interact with OpenXT directly.
  - Move the server portions to another separate but related project.

Optional Platform Layers (cont.)

- Layers that depend on major components in other layers should be built on top of those layers.
- Consider an example where each layer is built on the layer below: