Universal Secure Loader

Daniel P. Smith

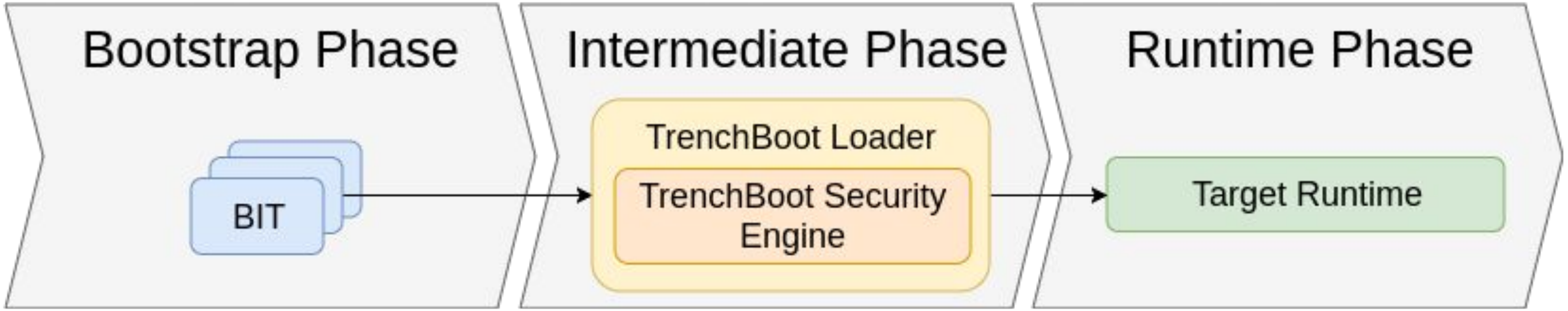
Background

- Apertus Solutions Owner/President/Chief Technologist
- OpenXT Maintainer and Contributor
  - Developed the first public “forward-seal” automated upgrade for a Measured Boot solution
- Xen Security Modules (XSM) Maintainer
- Founded TrenchBoot project in 2018
  - Purpose was to extract and generalize OpenXT’s Measured Boot
- Co-architect for Xen’s Hyperlaunch capability
  - TrenchBoot and Hyperlaunch were devised together as part of a larger security architecture

https://xen2020.sched.com/event/baXt/design-session-talk-reliable-platform-security-xen-and-the-fidelis-platform-for-hardened-access-terminals-hat

TrenchBoot Architecture

Every platform has a bootstrap environment used to prepare the hardware for runtime software to take control of the platform. This bootstrap often provides one or more Boot Integrity Technologies (BITs) that enable the construction of trust assertions about the platform and the software running on it.TrenchBoot provides a general purpose solution/approach to construct trust statements about the platform through the introduction of an Intermediate Phase.



Architecture: Possible Design Patterns

On initial review of the architecture, one might be lead that the realization of the TrenchBoot architecture results in a physically independent Intermediate Loader implementation. This is not the case, the architecture can be realized through multiple implementation models, aka Design Patterns. The two common patterns that are currently being implemented are:

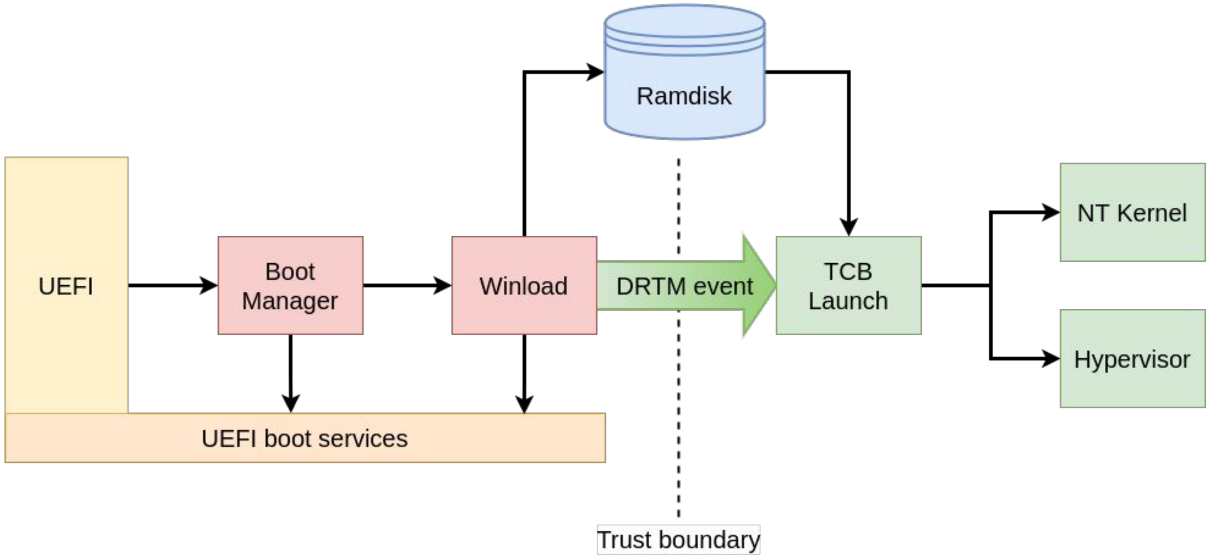
- An OS/Runtime independent system loader, aka intermediate loader
- A function of an OS/Runtime startup code

The Universal Secure Loader is an implementation of the intermediate loader pattern.

Intermediate Loader Design Pattern

An exemplar is the MS Windows approach to Dynamic Launch that uses the intermediate loader pattern. The TCB Launch is a discrete binary whose responsibility is to load, assess, and start either the HyperV kernel or the NT kernel. After starting the kernel, it is freed from memory.

The assessment by TCB Launch is utilized by Window's Defender System Guard capability to protect the integrity of the system.



https://www.microsoft.com/security/blog/2020/09/01/force-firmware-code-to-be-measured-and-attested-by-secure-launch-on-windows-10/

Intermediate Loader: Intel Trusted Boot Comparison

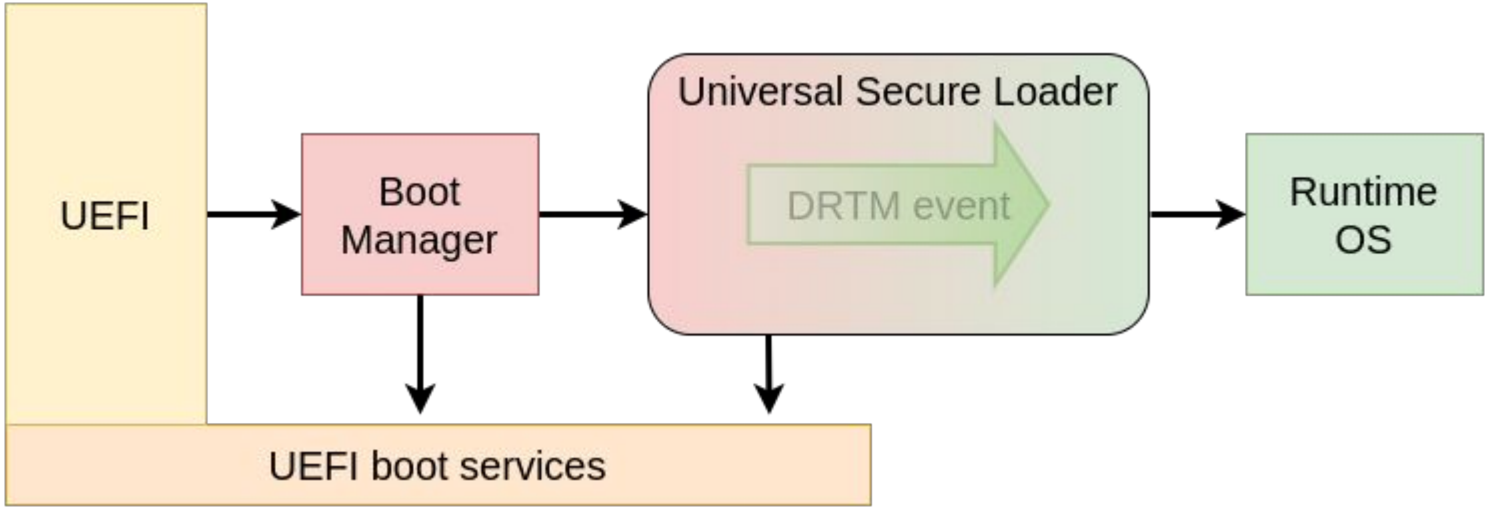
A common misconception is that Trusted Boot (tboot) is an intermediate loader, but this only partly true. This is partly due to tboot’s own description,

“*Trusted Boot (tboot) is an open source, pre-kernel/VMM module that uses Intel(R) Trusted Execution Technology (Intel(R) TXT) to perform a measured and verified launch of an OS kernel/VMM*”

Tboot is in fact an Exokernel that persists in memory and holds responsibility over managing the system power state management (ACPI). An OS kernel must be enlightened to call into tboot when wanting to switch ACPI power states.

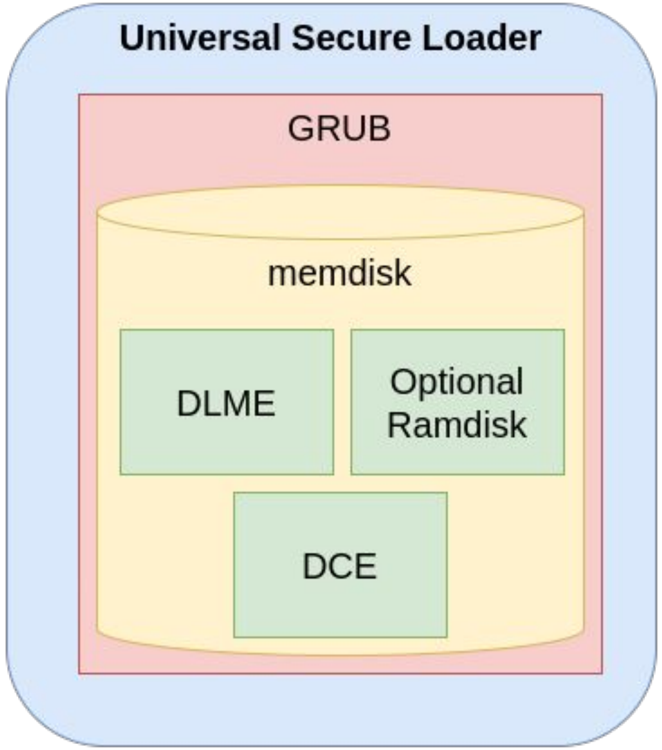
Universal Secure Loader

Universal Secure Loader (USL, pronounced uṣūl) provides a single binary that can perform a Dynamic Launch of OS kernels using either the Linux or both Multiboot protocols.



USL Internal Design

- The internal design to USL is to leverage the GRUB and its memdisk capability to create a single bootable binary.
  - Though any bootloader that implements the TrenchBoot Secure Launch Protocol can be used.
- GRUB functions as the D-RTM Configuration Environment (DCE) Preamble
- The memdisk is packed with the:
  - DCE
  - Dynamically Launch Measured Environment (DLME)
  - Optional Ramdisk for the DLME
- The DCE is platform specific:
  - Intel SINIT ACM (gen specific)
  - AMD Secure Loader Block
- The packaged DLME must:
  - Implement the TrenchBoot Secure Launch Protocol
  - Capable of loading and booting Runtime OS



USL Standard Components

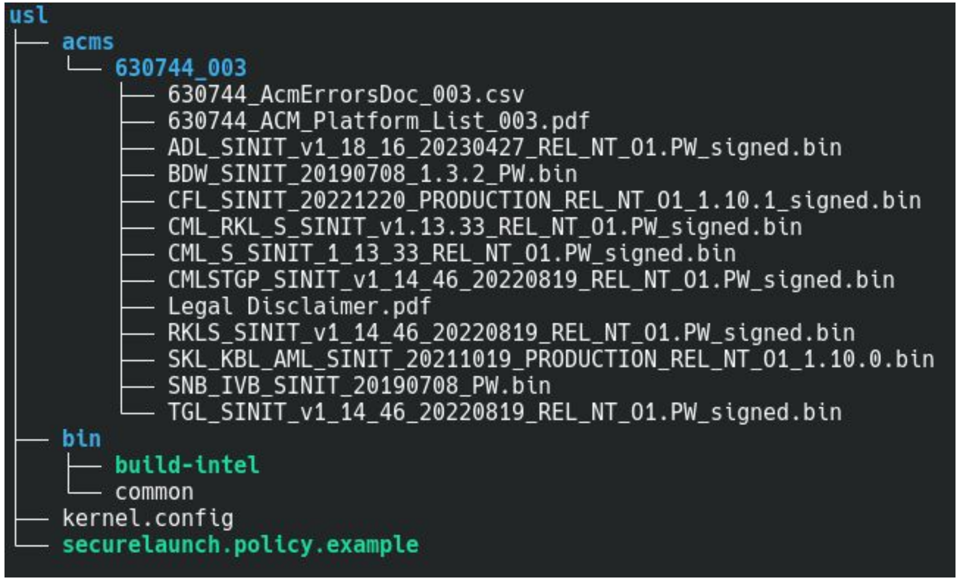
For binaries distributed by the TrenchBoot project, the following standard components will be used.

- DCE Preamble
  - GRUB
- DCE
  - Intel: Appropriate SINIT ACM
  - AMD: TrenchBoot Secure Kernel Loader (SKL) (future release)
- DLME
  - Minimally configured Linux Secure Launch Kernel
  - u-root sluinit

USL Build System

A build system that will compile the necessary components and package them into a single UEFI bootloader.

- Configured through environment variables
- For Intel, generates a binary per CPU generation
- Provides for customization
  - Linux kernel config
  - Custom initramfs image
- Incorporate alternative components as they become available



```
usl
├── acms
│   └── 630744_003
│       ├── 630744_AcmErrorsDoc_003.csv
│       ├── 630744_ACM_Platform_List_003.pdf
│       ├── ADL_SINIT_v1_18_16_20230427_REL_NT_O1.PW_signed.bin
│       ├── BDW_SINIT_20190708_1.3.2_PW.bin
│       ├── CFL_SINIT_20221220_PRODUCTION_REL_NT_O1_1.10.1_signed.bin
│       ├── CML_RKL_S_SINIT_v1.13.33_REL_NT_O1.PW_signed.bin
│       ├── CML_S_SINIT_1_13_33_REL_NT_O1.PW_signed.bin
│       ├── CMLSTGP_SINIT_v1_14_46_20220819_REL_NT_O1.PW_signed.bin
│       ├── Legal Disclaimer.pdf
│       ├── RKLS_SINIT_v1_14_46_20220819_REL_NT_O1.PW_signed.bin
│       ├── SKL_KBL_AML_SINIT_20211019_PRODUCTION_REL_NT_O1_1.10.0.bin
│       ├── SNB_IVB_SINIT_20190708_PW.bin
│       └── TGL_SINIT_v1_14_46_20220819_REL_NT_O1.PW_signed.bin
├── bin
│   ├── build-intel
│   └── common
├── kernel.config
└── securelaunch.policy.example
```

Deploying and Configuring USL

Oracle and Apertus collaborated in the development of the Secure Launch uinit (sluinit) for u-root's uinit system. The u-root project was selected for its SystemBoot capability designed for LinuxBoot. The benefits to sluinit are:

- Rich environment to build custom integrity measurement collection and attestation agents.
- Designed to consume a flexible JSON policy file.
- Written in a memory safe language.

```
{
    "default_action": "",
    "collectors": [
      {
        "type": "dmi",
        "events": [
          {
            "label": "BIOS",
            "fields": []
          },
          {
            "label": "System",
            "fields": []
          }
        ]
      },
      {
         "type": "storage",
         "paths": [ "sda3" ]
      }
    ],
    "launcher": {
        "type": "kexec",
        "params": {
            "initrd":"sda2:/initramfs.img",
            "cmdline":"BOOT_IMAGE=/vmlinuz root=/dev/sda3 ro intel_iommu=on
pci=realloc iommu=pt",
            "kernel":"sda2:/vmlinuz"
         }
    },
    "eventlog": {
        "type": "file",
        "location": "sda2:/evtlog"
    }
}
```

https://github.com/u-root/u-root/blob/main/docs/securelaunch/README.md

USL Project State

Status:

- Build system is working with CI binary generation in progress.
- Currently builds images for Intel TXT/vPro platforms.

Next Steps:

- Incorporate Mulitboot support in sluinit.
- Track changes as the TrenchBoot Secure Launch Protocol stabilizes.
- Seek volunteers to test CI images on a variety of Intel generations.
- Collect bug reports, establishing an HCL for TrenchBoot.

GitHub Project: https://github.com/TrenchBoot/usl

USL Roadmap

Goal:

- To continue refining USL and its build system to allow the easy adoption of Dynamic Launch, maximizing the ability to establish integrity about a system.

Target Users:

- Home Users
- Enterprise
- Vendors/Supply Chain

Challenges:

- UEFI, the proverbial gift that keeps on giving

Exemplar: Secure Upgrades with DRTM

USL can be used to build a “Management MLE”:

1. A single binary solution.
2. Launched by kexec, thus no reboot is required.
3. Uses a kernel without network drivers and IOMMU blocking all DMA
4. Unlocks TPM Platform Owner to allow forward seal to upgraded environment.

Details can be seen in the FOSDEM 2021 presentation by the same name.

https://archive.fosdem.org/2021/schedule/event/firmware_suwd/

Exemplar: Supply Chain Integrity

Extend the USL runtime, u-root, with ability to work with Supply Chain blockchains. An implementer could take two approaches:

- USL constructs a blockchain of installed components to be presented to an attestation service
- USL, through some appropriate mechanism[1], is provided a vendor's Supply Chain blockchain which is used to evaluate the system for compliance before booting.

The former scenario might be used to allow a client device access to a cloud provider's service. The latter might be used by an enterprise to ensure clients and servers are in a good state before allowing them to boot on the enterprise network.

[1] *determined by customer requirements and security posture*

Closing Remarks

- USL utility is the build system.
- GitHub CI/CD will run the build system to produce a set of general purpose USL images to allow the average user to install and use.
- USL brings the security benefits of dynamic secure launch to a variety of unmodified operating system kernels. For advanced security architectures like Xen Hyperlaunch, additional security properties are gained by the upcoming TrenchBoot enlightenment of Xen.

Q&A

References

TrenchBoot Website: https://trenchboot.org

Past Events: https://trenchboot.org/events/

Latest Linux Secure Launch Series: https://lore.kernel.org/lkml/20240214221847.2066632-1-ross.philipson@oracle.com/

Xen Hyperlaunch: https://xenbits.xen.org/docs/unstable/designs/launch/hyperlaunch.html