



Securing the EDK II Image Loader

Marvin Häuser

*Technische Universität Kaiserslautern;
Ivannikov Institute for System Programming
of the Russian Academy of Sciences
mhaeuser@posteo.de*

Vitaly Cheptsov

*Ivannikov Institute for System Programming
of the Russian Academy of Sciences
cheptsov@ispras.ru*

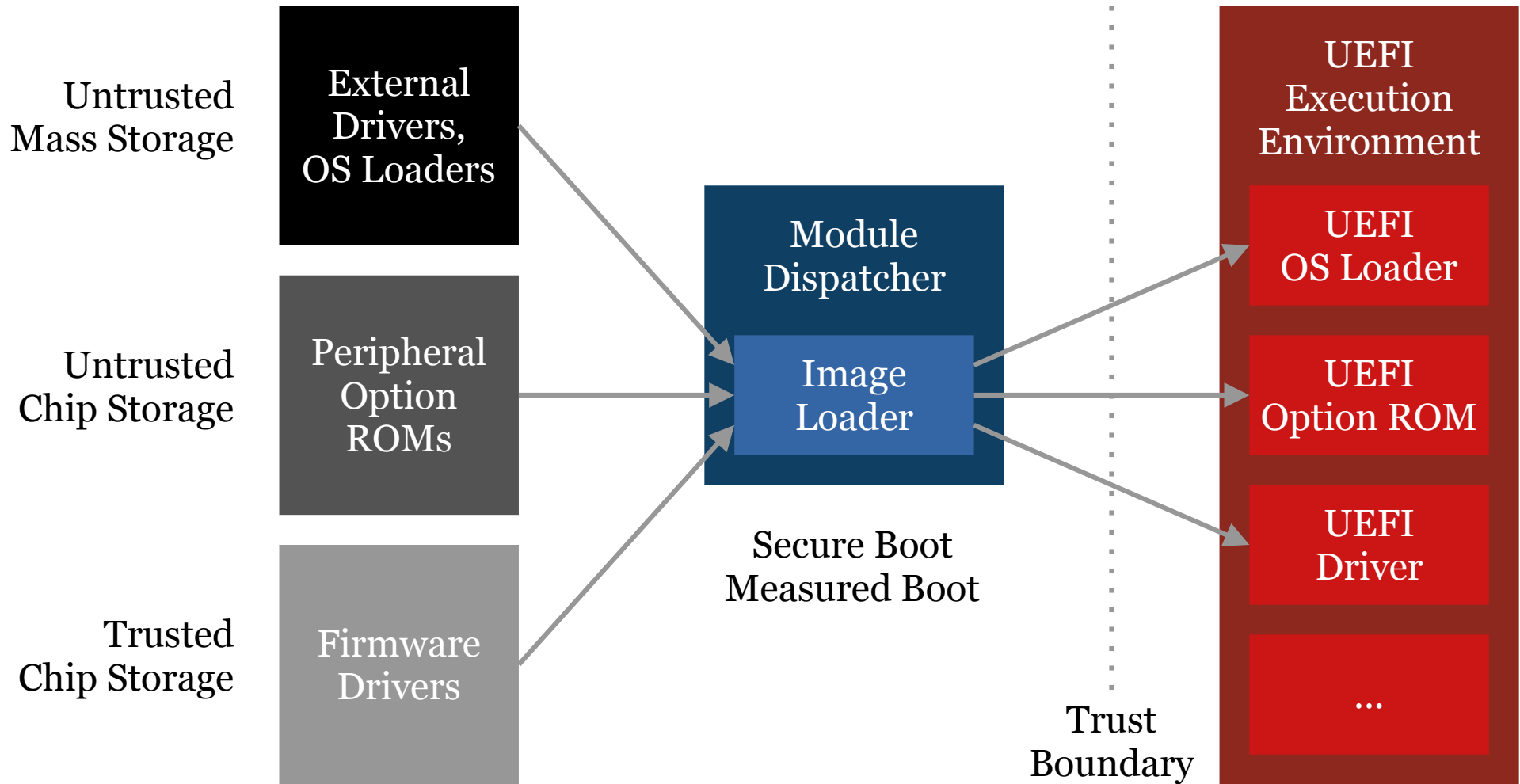


Image Loader requirements

- Verify raw file conformance to the key parts of the PE specification
- Authenticate raw file (e.g. signature)
- Map the raw file into memory space
- Ensure code safety in the process (e.g. alignment, wraparounds, termination, etc.)

EDK II as a UEFI platform core

- Contains most platform-agnostic drivers
- Manual code review by the maintainers
- Gradual but slow improvements
- Dozens of “defects” reaching production
 - “Design defects”: Design that provokes bugs
 - “Functional defects”: Implementation bugs
 - Insufficient treatment of critical modules

Terse Executables

- Stripped PE executable to reduce size
- Poor specification makes tight models hard
- Only suitable internally due to lack of signing
- Not updated fields overcomplicate the Loader

The EDK II Image Loader

- It encourages TOC/TOU-vulnerable loading
- It accesses many unchecked offsets
- It does not consider data alignment at all
- It often does not report failures
- It misaligns TE Images in some cases
- It does not fully initialise the image destination

Deductive verification in practice

- C compiler (CompCert) with Coq by INRIA
- Microkernel (seL4) with Isabelle by OK Labs
- Cryptography (HACL *) with F* by Proj. Everest
- Hypervisor (Hyper-V) with VCC by Microsoft
- Avionics software with Frama-C by Airbus

AstraVer Toolset

- Based on established Frama-C technology
- Extended support for binary arithmetics
- Real-world usability showcased by verifying Astra Linux Security Module
- ACSL: First-Order Logic with C-like syntax
- Automatic and manual proving with Why3

Environment formal model

- Own definitions required to extend AV (Jessie)
- Memory model based on CompCert, Version 2,
Type model based on C constraints
 - Compatible with common compilers
 - Honours common culprits like Strict Aliasing

PE formal model

- Modelled in ACSL, file adherence proven
 - File Headers and Relocation Directory
 - Sane values (offsets, sizes, etc.)
- Derived from PE specification constraints
 - Adapted by us towards real-world binaries to increase performance, security, and compatibility
- Incomplete, but sufficient to prove the goals

Authenticode Signature

- Overcomplicated hashing algorithm
 - Sorts sections, bookkeeps number of hashed bytes
- Requires dynamic memory allocation
- Windows implementation does not follow specs
- Permits trailing data in the real world

Tightening the model

- Sane alignment rules for offsets
- Sane values (e.g. at least one section)
- Avoid uncommon, ambiguous constructs
 - Compound Base Relocations improperly defined
- Adjacent, sorted file sections
 - Simplifies Authenticode into almost a linear hash

Tightening the model (Cont'd)

X

Permissive PE
File Headers
Blob a
Section 2
Cert Directory
Section 1
Blob b

- Gaps and Blobs increase the risk for unhashed data
- Intersections allow for easy increase of hashing expense
- Unsorted sections require Authenticode to dynamically allocate memory and sort
- Non-trailing Cert Directory risks accidentally hashing it and covercomplicates the Authenticode algorithm

✓

Realistic PE
File Headers
Section 1
Section 2
Cert Directory

ACSL proving

- Predicates describe model properties
 - File bounds, alignment, disjunct sections, etc.
- Functions and statements are annotated with preconditions and postconditions
 - Two-way relation: model adherence \Leftrightarrow success

```
logic integer image_sect_correct_address (EFI_IMAGE_SECTION_HEADER *Sections, int  
SectIndex == 0 ?  
    image_loaded_hdr_virtual_size (Sections, SizeOfHeaders, SectionAlignment) :  
    align_up (image_sect_top (Sections + SectIndex - 1), SectionAlignment);
```

Culprits

- The AstraVer memory model is limited
 - Additional axioms required for extension
- The solvers' binary arithmetics are limited
 - Interactive proof work required
- Complex expressions are impractical
 - No FOL “counting”, no exponents, no HOL

Conventional validation

- Manual code review for common culprits
- Static analysis using SVACE and Coverity
- Dynamic tooling with libFuzzer, ASan/UBSan
 - 100 % coverage excluding defensive code
- Real-world test booting Microsoft Windows, Apple macOS, and GNU Linux bootloaders

Conclusion

- Re-implemented a fully-featured Image Loader
- EDK II implementation issues fully addressed
- 100% C safety proof coverage for core code
- Correctness proofs cover most core code
 - “Fully proven” execution possible on IA32 and X64
- So far no single bug found in proven code ☺

Future work

- Proving of image-embedded signature handler
- Allowing for tighter format constraints
 - Trailing data to header, sections or Cert Directory
- Integration with the EDK II core modules
- Submission of the project to TianoCore

Material available at: github.com/mhaeuser/ISPRASOpen-SecurePE

Thank you.
Questions?

ISP RAS: Project Amaranth

- Restricted UEFI specification subset
- Focused on security
- Open-source community collaboration
- Incorporates formal methods
- Targets virtual machines and real hardware
- In active development since 2020