
# The Tragedy of the Commons in Platform Security

**John Loucaides**

**Platform Security Summit 2019**

# My Problem. Your Problem. Everyone's problem.

- **How big is this industry again?**
  There must be conflicting goals.
- **Do you know your dependencies?**
  There must be something changing about them.
- **Can we keep up with all this and do our jobs?**

# Current State: Vulnerabilities, Advisories and Updates

```
[*] running module: chipsec.modules.common.bios_wp
[x][ ================================================================================
[x][ Module: BIOS Region Write Protection
[x][ ================================================================================
[*] BC = 0x     288 << BIOS Control (b:d.f 00:31.5 + 0xDC)
   [00] BIOSWE           = 0 << BIOS Write Enable
   [01] BLE              = 0 << BIOS Lock Enable
   [02] SRC              = 2 << SPI Read Configuration
   [04] TSS              = 0 << Top Swap Status
   [05] SMM_BWP          = 0 << SMM BIOS Write Protection
   [06] BBS              = 0 << Boot BIOS Strap
   [07] BILD             = 1 << BIOS Interface Lock Down
[-] BIOS region write protection is disabled!

[*] BIOS Region: Base = 0x00340000, Limit = 0x007FFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (84)     | 83EF03B0 | 003B0000 | 003EFFFF | 1   | 0
PR1 (88)     | 862F03F0 | 003F0000 | 0062FFFF | 1   | 0
PR2 (8C)     | 866F0630 | 00630000 | 0066FFFF | 1   | 0
PR3 (90)     | 87EF06F0 | 006F0000 | 007EFFFF | 1   | 0
PR4 (94)     | 87FF07F0 | 007F0000 | 007FFFFF | 1   | 0

[!] SPI protected ranges write-protect parts of BIOS region (other parts of BIOS can be
modified)

[!] BIOS should enable all available SMM based write protection mechanisms or configure SPI
protected ranges to protect the entire BIOS region
[-] FAILED: BIOS is NOT protected completely
```

# Current State: Bypassing Security with Drivers

Signed driver allows direct access to arbitrary hardware/firmware:

- Arbitrary ring 0 memcpy
- Arbitrary physical memory write
- Physical address lookup from virtual address
- Arbitrary MSR read/write
- Arbitrary CR read/write
- Arbitrary IO Port read/write
- Arbitrary PCI config read/write

Can either be already on system or carried with malware


**https://github.com/eclypsium/Screwed-Drivers**

# New Approach: Disproportionally help defenders

**DevOps/DevSecOps**

- Real-time, automated checks
- Update deployment rings
- Quick test & response

**Focus Areas**

- Identification
- Configuration

**Working Together**

- Enable additive security—“plugins” to enforce policies
- Enable integrity checks—“plugins” to observe/measure
- Enable trusted sources—how do we know what is “official”?

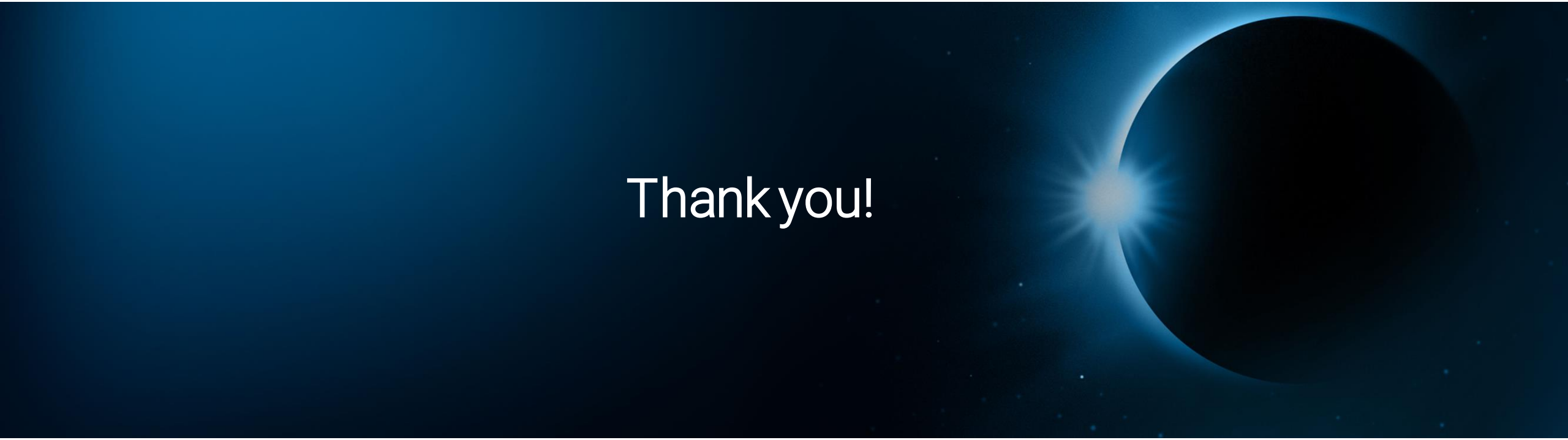
# Maybe We're on the Right Track...

**Open Communities & Sharing**

- Code (Tianocore, Coreboot, etc)
- Measurements & Updates (LVFS)
- Security Advisories
- Testing (CHIPSEC, HBFA, etc)

**Next? Making it easier...**

- Published Measurements
  - Hashes
  - Behaviors
  - Status / config
- Integrity Interfaces
  - Read the measurements
- Policy plugins


Thank you!