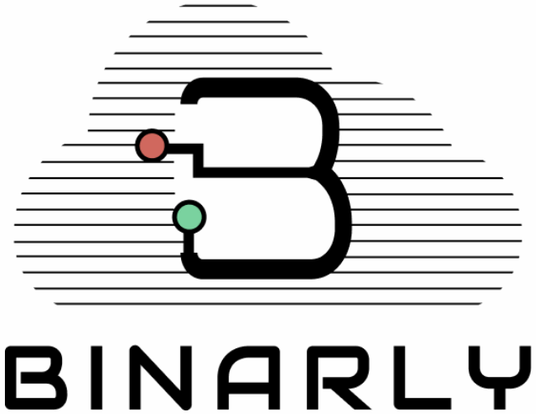
presented by



Tackling Security Through the Supply Chain

UEFI 2022 Virtual Summit

May 17, 2022

Presented by Alex Matrosov, CEO, Binarly

and Tim Lewis, CTO, Insyde Software

Meet the Presenters

Tim Lewis

CTO, Insyde Software

Alex Matrosov

CEO, Binarly

Agenda

- Firmware Supply Chain and Firmware Security
- Case Studies In Supply Chain Failures
- Bridging The Gaps In The Supply Chain
- Next Steps

Overview

"Firmware presents a large and ever-expanding attack surface, as the population of electronic devices grows. Securing the firmware layer is often overlooked, but it is a single point of failure in devices and is one of the stealthiest methods in which an attacker can compromise devices at scale. Over the past few years, hackers have increasingly targeted firmware to launch devastating attacks."

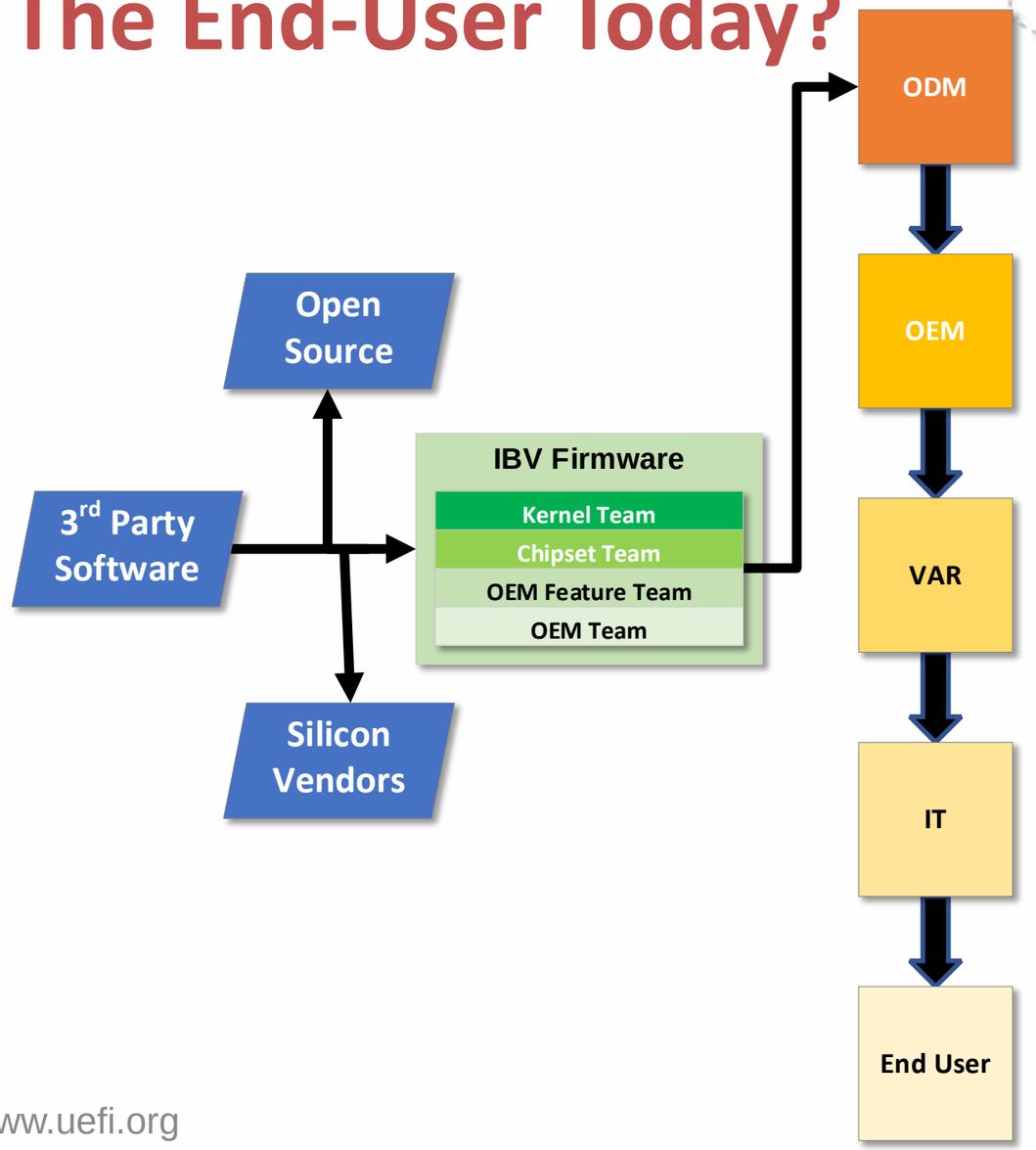
– U.S. Dept. of Commerce and Homeland Security, 24 Feb 2022

Firmware Security Supply Chain

How Does Security Get To The End-User Today?

- How does each stage know that what they have received has not been tampered with?

- How does each stage know if a security disclosure applies to what they have received?

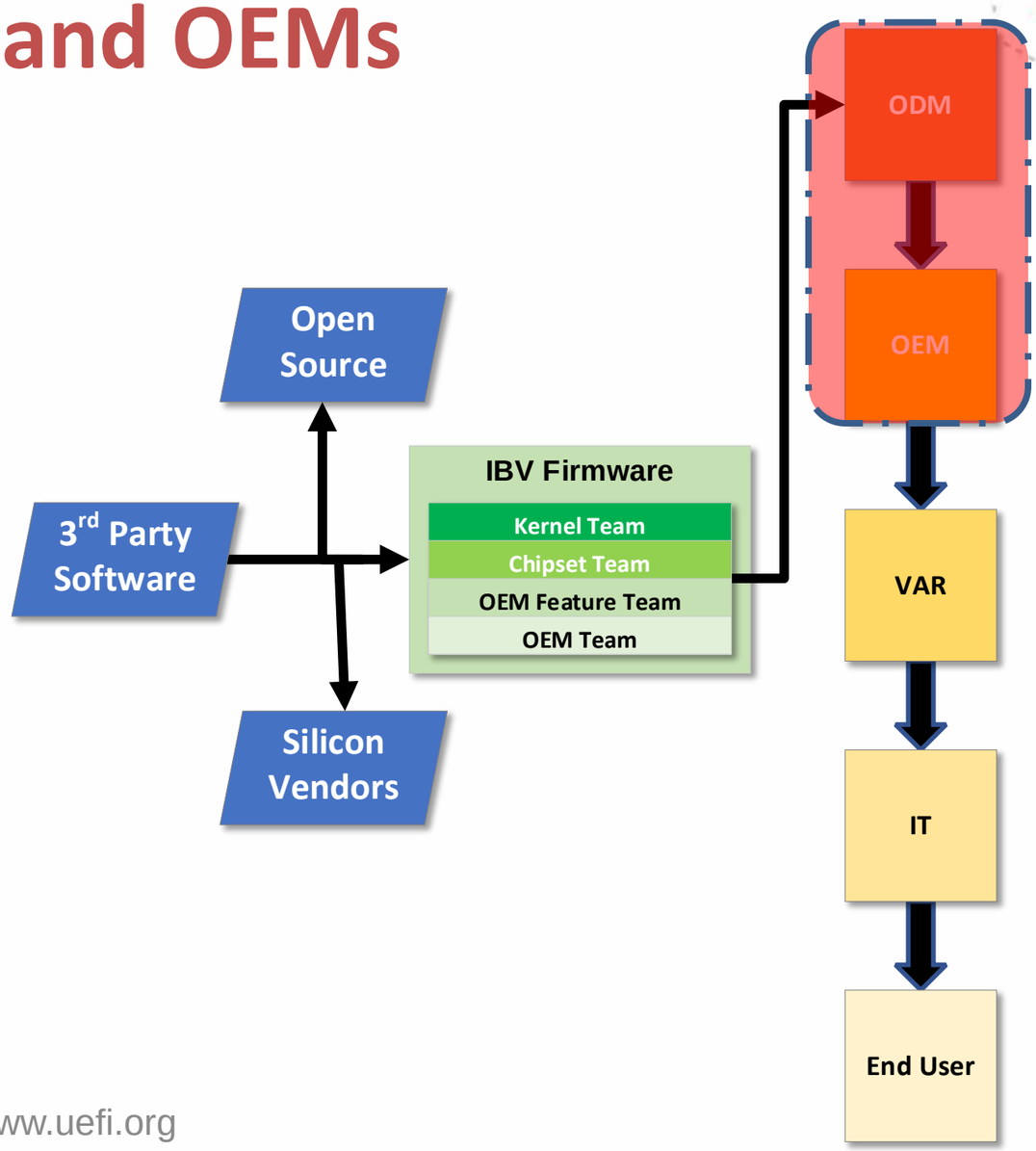


Firmware Security Supply Chain

The Critical Role of ODMs and OEMs

- ODMs/OEMs create production firmware binaries and distribute them.

- Last stage to see firmware ingredients.

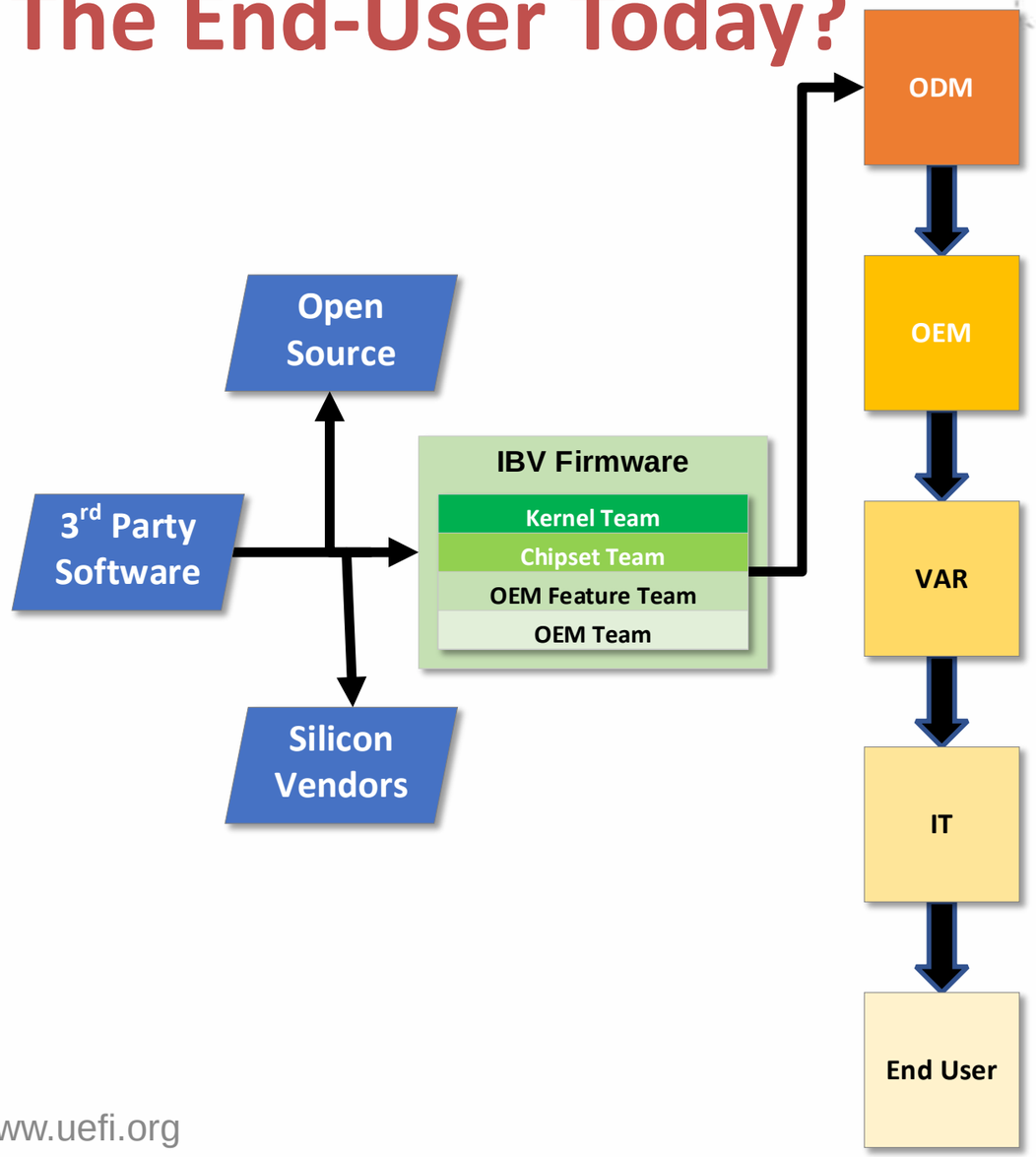


Firmware Security Supply Chain

How Does Security Get To The End-User Today?

- Production firmware consists of many ingredient providers through several stages.

- Firmware ingredients are received from the previous stage, possibly modified and combined, and then passed to the next stage.

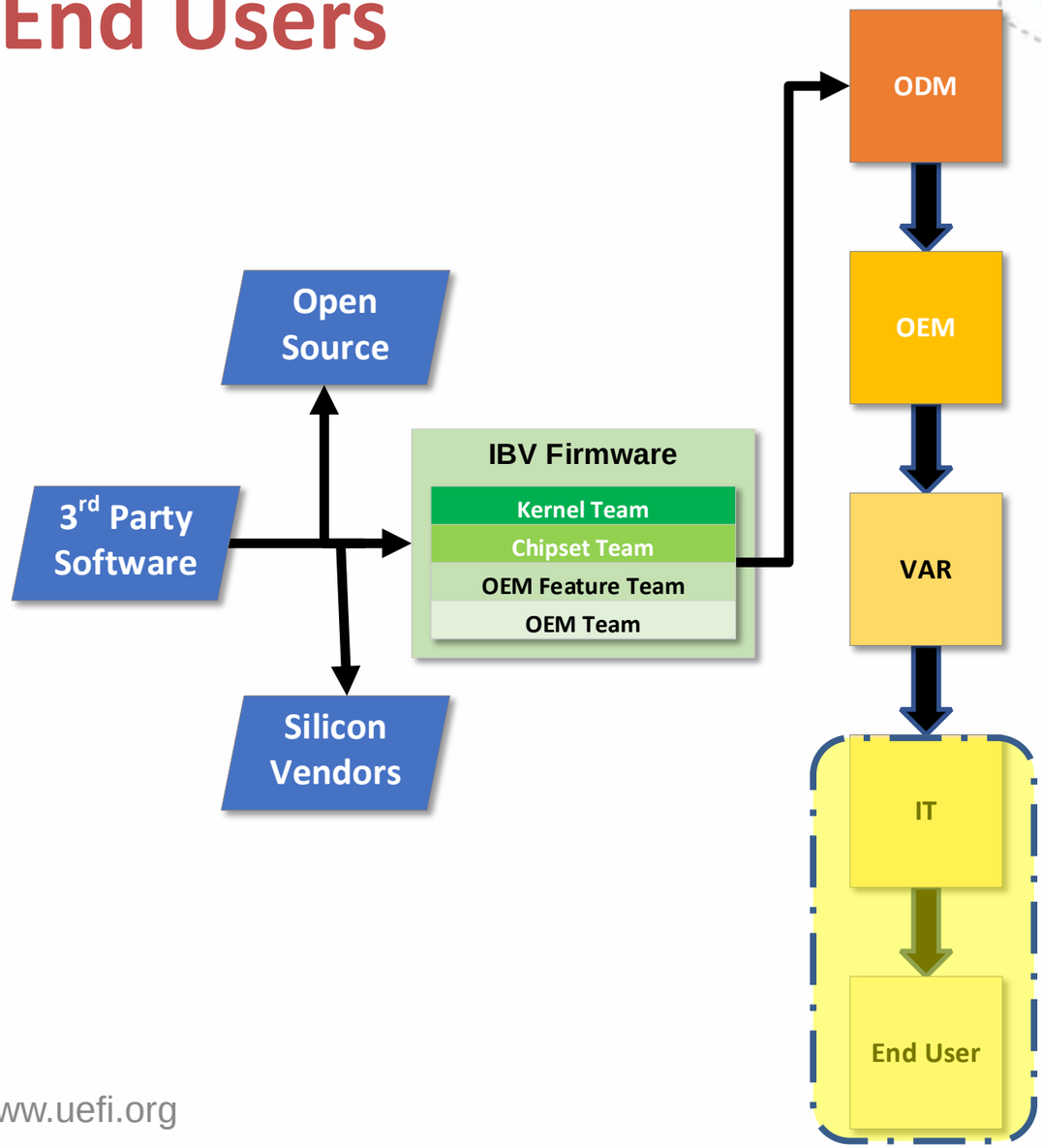


Firmware Security Supply Chain

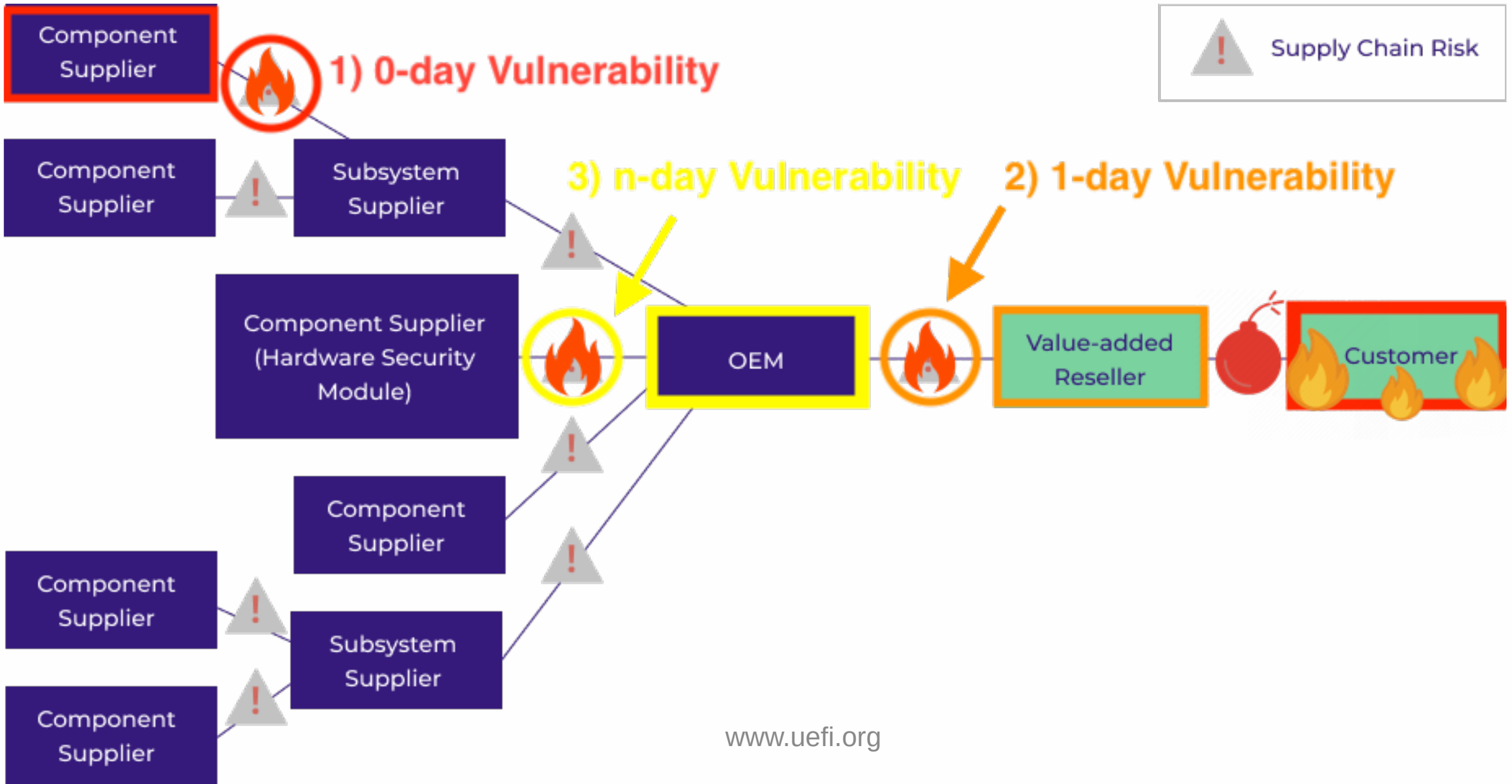
The Critical Role of IT and End Users

- IT/End User responsible for updating firmware.

- End users are the least likely to read published reports and have the least visibility into firmware ingredients.



Supply Chain Complexity is Growing



Binarly Vulnerability Disclosures Statistics

Vulnerability Category	Count	Average Impact
SMM Privilege Escalation	15	CVSS: 8.2
SMM Memory Corruption	22	CVSS: 8.2
DXE Memory Corruption	5	CVSS: 7.7

* Based on Binarly disclosures: https://www.binarly.io/advisories

LEADER BOARD

ROUND ONE

Join us in congratulating...

First to Submit:
ZwinK

Highest Single Payout:
Mickey (@HackingThings)

Most Eligible Reports:
Alex Matrosov

Widest Impact:
Alex Matrosov

BINARLY

Helping Others:
Mickey (@HackingThings)

Just in Time:
Dan Lutas

ROUND TWO

Bounty recipients named mid-April

ROUND THREE

Bounty recipients named mid-May

* **Intel Project Circuit Breaker**: https://www.projectcircuitbreaker.com/

Where Firmware Supply Chain Has Failed

Vulnerability	Category
Intel BSSA DFT	Silicon Vendor Reference Code
AMD CVE-2021-39298	Silicon Vendor Reference Code
Insyde IdeBusDxe	IBV Reference Code
AMI UsbRt	IBV Reference Code
HP CVE-2022-23932	ODM/OEM Firmware Code
Dell CVE-2022-24419	ODM/OEM Firmware Code
Lenovo CVE-2021-3971	ODM/OEM Firmware Code

* Based on Binarly disclosures: https://www.binarly.io/advisories

Intel BSSA DFT (INTEL-SA-00525)

```
char isPhysicalPresenceEstablished()
{
  return 1;
}
```

```
(*PeiServices)->LocatePpi(PeiServices, &gReadOnlyVariable2Guid, 0, 0, &ReadOnlyPpi);
ZeroMem(syscg_stack, 2048);
ReadOnlyPpi->GetVariable(ReadOnlyPpi, L"syscg", &gSsaBiosVariablesGuid, 0, &DataSize, syscg_stack);
syscg = AllocatePool(DataSize);
memcpy_0(syscg, syscg_stack, DataSize);
TotalConfigs = *(syscg + 0x10);
EvLoadTool(host, syscg, &ConfigIndex, &ImageBase);
```

https://www.binarly.io/posts/Attacking_(pre)EFI_Ecosystem
https://www.binarly.io/posts/Detecting_Firmware_vulnerabilities_at_scale_Intel_BSSA_DFT_case_study

IdeBusDxe (VU#796611)

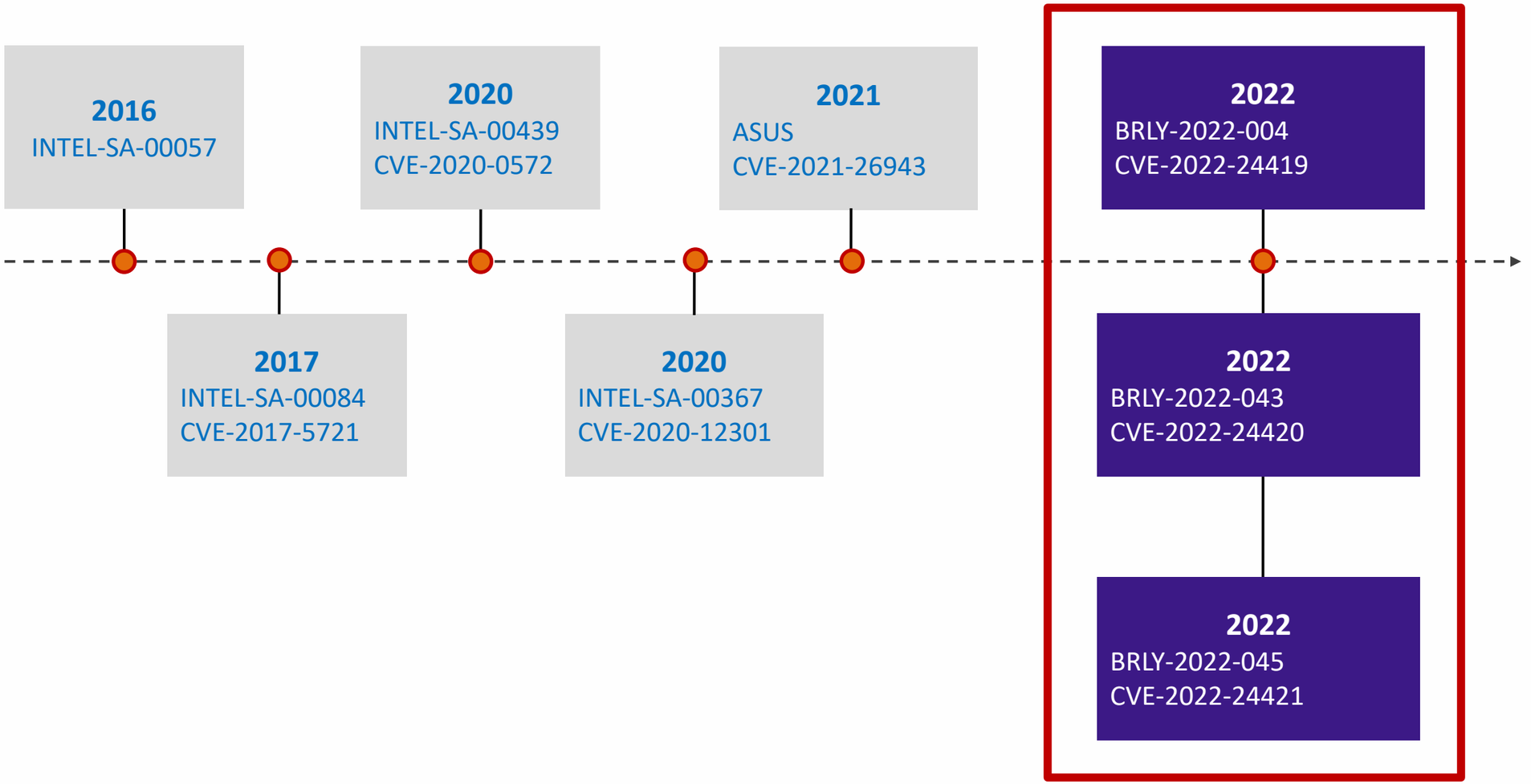
```
EFI_STATUS __fastcall ChildSwSmiHandler(
        EFI_HANDLE DispatchHandle,
        const void *Context,
        char *CommBuffer,
        UINTN *CommBufferSize)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( !CommBuffer || !CommBufferSize )
    return 0i64;
  if ( *CommBuffer == 1 )
  {
    Buffer = 0i64;
    if ( gBS->LocateHandleBuffer(ByProtocol, &EFI_ATA_PASS_THRU_PROTOCOL_GUID, 0i64, &NoHandles, &Buffer) )
    {
      v5 = EFI_NOT_FOUND;
    }
    else
    {
      gBS->FreePool(Buffer);
      Buffer = 0i64;
      BufferSize = 0i64;
```

```
_WriteStatus:
  // Input Communication Buffer is not validated to be outside of SMRAM
  // since the Communication Buffer size (`*CommBufferSize`)
  // is not checked to be valid for relying on the validation implemented
  // in PiSmmCommunicationSmm module (based on a Communication Header)
  *(_QWORD *)(CommBuffer + 4) = Status;
  return 0i64;
}
```

* Based on Binarly disclosures: https://www.binarly.io/advisories

UsbRt - Intel & Binarly Disclosures



* Based on Binarly disclosures: https://www.binarly.io/advisories

AgesaSmmSaveMemoryConfig

```
__int64 __fastcall SwSmiHandler(__int64 a1)
{
  __int64 result; // rax
  UINT32 v3; // [rsp+30h] [rbp-48h] BYREF
  UINTN v4; // [rsp+38h] [rbp-40h] BYREF
  EFI_SMM_SW_DISPATCH2_PROTOCOL *EfiSmmSwDispatch2Protocol; // [rsp+40h] [rbp-38h] BYREF
  __int64 v6[6]; // [rsp+48h] [rbp-30h] BYREF

  v3 = 6;
  v4 = 32i64;

  if ( (gRT->GetVariable(L"AmdMemContextData", &VendorGuid, &v3, &v4, v6) & 0x8000000000000000ui64) == 0i64 )//
                                                //
                                                // overwrite GetVariable service address in the EFI_RUNTIME_SERVICES table with the shellcode address
                                                //
    sub_15FC(v6);
  result = gSmst->SmmLocateProtocol(&EFI_SMM_SW_DISPATCH2_PROTOCOL_GUID, 0i64, &EfiSmmSwDispatch2Protocol);
  if ( result >= 0 )
    return EfiSmmSwDispatch2Protocol->UnRegister(EfiSmmSwDispatch2Protocol, a1);
  return result;
}
```

https://www.binarly.io/advisories/BRLY-2021-004

Static Code Analysis Tools Limitations

```
// BRLY-2021-040 (CVE-2022-23932)
// HP coordinated fix 03/08/2022

if ( CommBuffer->Sig == 'GFCU' )
        {
          if ( CommBuffer->Case == 0x10 )
          {
            if ( !gBufferPtr )
            {
              BufferPtr1 = GetCopy(0x78, &CommBuffer->BufferPtr);
              BufferSize = CommBuffer->BufferSize;
              BufferPtr2 = CommBuffer->BufferPtr;
              gBufferPtr = BufferPtr1;
              sub_2288(BufferPtr2, BufferSize);

              // Vulnerability present below
              PcdProtocol = BsLocatePcdProtocol();
              if ( (PcdProtocol->Get8)(0x2C4) == 1 )
                HandlerUnregister();
            }
          }
          ...
        }
```

```
// BRLY-2021-047 (CVE-2022-XXXXX)
// HP        fix 03/15/2022

if ( CommBuffer->Sig == 'GFCU' )
        {
          switch ( CommBuffer->Case )
          {
            case 0x10:
              if ( !gBufferPtr )
              {
                BufferPtr1 = GetCopy(0x78, &CommBuffer->BufferPtr1);
                BufferSize = CommBuffer->BufferSize;
                BufferPtr2 = CommBuffer->BufferPtr;
                gBufferPtr = BufferPtr1;
                sub_261C(BufferPtr2, BufferSize);

                // Vulnerability present below
                PcdProtocol = BsLocatePcdProtocol();
                if ( (PcdProtocol->Get8)(0x23B) == 1 )
                    HandlerUnregister();
              }
            }
            ...
        }
```

https://www.binarly.io/advisories/BRLY-2021-040

Dependency Graph Limitations

TrustedDeviceSetupApp - 658D56F0-4364-4721-B70E-732DDC8A2771

BRLY-2021-044 - not exploitable on Intel M15

```
DataSize = GetDataSize(Data);
Buffer = gBuffer;
Size = DataSize;
while ( Buffer != &gBuffer )
{
  if ( !CompareMemWrapper(Buffer + 49, Data, Size) )
  {
    CopyMemWrapper((Buffer + 2), a2, 32);
    return 0;
  }
  Buffer = *Buffer;
}
Mem = AllocateZeroPool(Size + 0x31);          // Callout here (gBS->AllocatePool)
```

- **Call-out vulnerability in SMI handler registered in UEFI Application**
- **Code removed from EDKII in 2018**
- **The pattern discovered in 2022 firmware, linked from another library in SecurityPkg by mistake**

Compilers-Generated Artifacts

SmmIsBufferOutsideSmmValid() - SMM input pointer validation routine

1 - normal version

2 - compiler-optimized version (hardcoded size)

1
```
char __fastcall SmmIsBufferOutsideSmmValid(unsigned __int64 ptr, unsigned __int64 size)
{
```
2
```
char __fastcall SmmIsBufferOutsideSmmValid(unsigned __int64 ptr)
{
```
...

1
```
if ( size <= gTopMemoryAddress && ptr <= gTopMemoryAddress )
```
2
```
if ( (unsigned __int64)gTopMemoryAddress >= 0x20 && ptr <= gTopMemoryAddress && ptr <= gTopMemoryAddress - 0x1F )
```
...

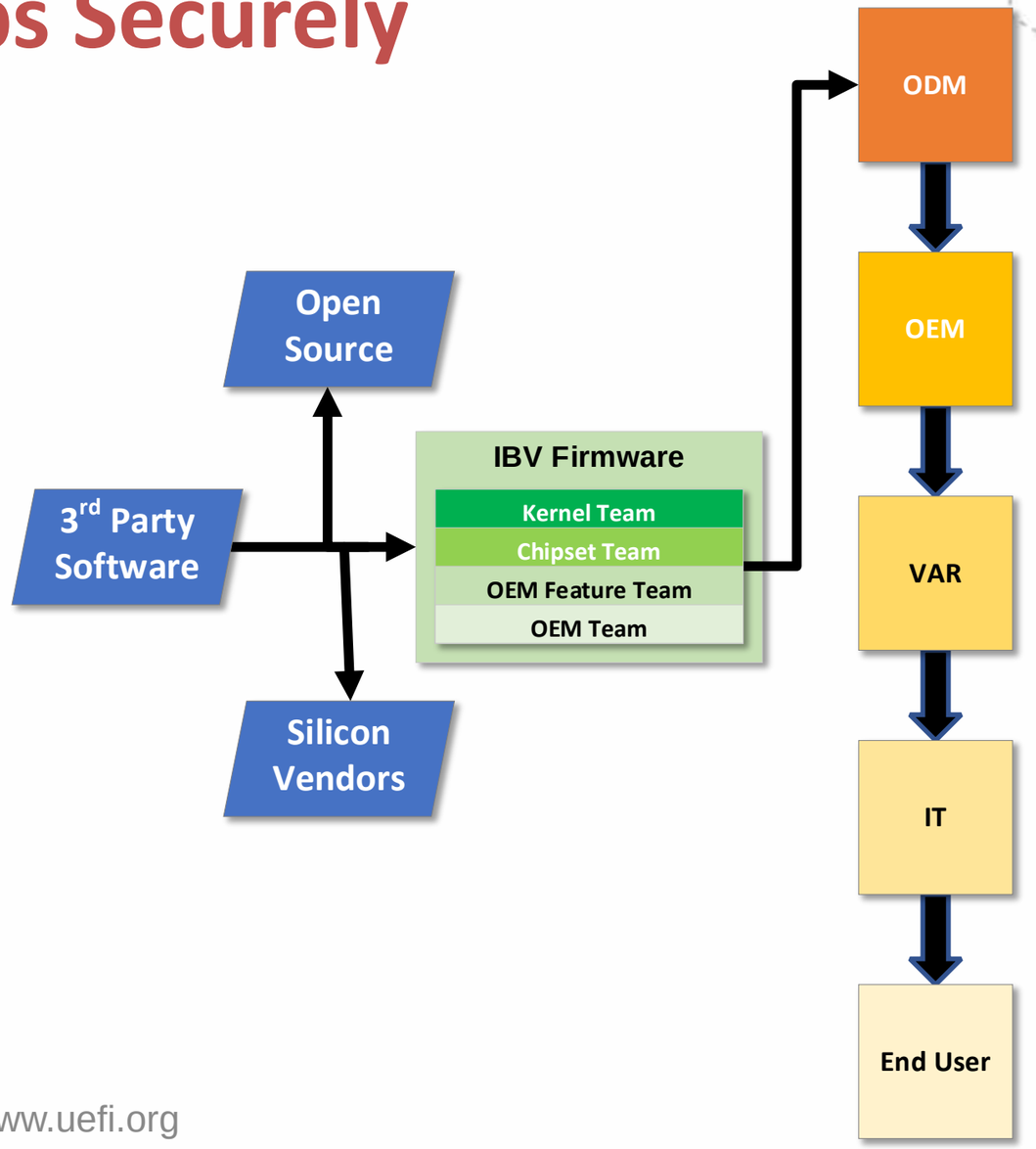
1
```
if ( v6 < ptr + size )
  return 0;
```
2
```
if ( v5 < ptr + 0x20 )
  return 0;
```
...

1
```
if ( ptr >= *(_QWORD *)(v9 + 8) && ptr + size <= *(_QWORD *)(v9 + 8) + (*(_QWORD *)(v9 + 0x18) << 12) )
```
2
```
if ( ptr >= *(_QWORD *)(v8 + 8) && ptr + 0x20 <= *(_QWORD *)(v8 + 8) + (*(_QWORD *)(v8 + 0x18) << 12) )
```
...

Firmware Security Supply Chain

Bridging Supply Chain Gaps Securely

- What firmware ingredients does each product have?
- Which firmware ingredients have a known vulnerability?
- Have firmware ingredients been tampered with?
- Do users know to update their product?



What Firmware Security Tools Do We Have?

Delivery	Supply Chain	Verify pedigree and tamper status of source code/pre-built binaries	SPDX, CycloneDX, SWID
Build Time	Static Analysis	Analyze the source code and binaries for common security mistakes	Klocwork, Coverity, Visual Studio code analysis, FwHunt, binary analysis
	Code Review	Review the source code based on threat model in high-risk technology areas	EDK2 security review guidelines, MISRA C
Testing	Security Testing	Use traditional testing and fuzzing for data that crosses trust boundaries	Defensics, CHIPSEC, FWTS
Runtime	Tamper Protection	Measure firmware code/data, compare measurements, log measurements	Intel BootGuard, TCG event log, golden measurements
	Compiler Runtime Protections	Use the compiler to inject checking code for common security failures	Integer overflow, Uninitialized variables, Local stack corruption
	Access Control	Protect against module accessing resources that are not permitted	Intel System Resource Defense (ISRD), Heap Guard, Stack Guard, NULL Pointer, PE/COFF
	Kernel Protections	Detect unsafe usage patterns and data corruption in BIOS kernel	Heap and pool corruption, TPL inversion, critical data structure checks, ASLR, panic
	IT Intervention	Alert users/IT, provide response options (lock, shutdown, reflash, etc.)	HIRS ACA, BMC, firmware security monitoring

Firmware Security Tools We Have SPDX SBOMs

- SPDX SBOMs focus on transmission of source ingredients before creating the production firmware binaries.

- SPDX SBOMs help OEMs/ODMs to:
  - Track the origin and licenses of ingredient source code and binaries.
  - Identify whether ingredients have been modified from stage to stage.
  - Know if products contain ingredients with reported known vulnerabilities.

- Support:
  - Tianocore tags all files with “SPDX-License-Identifier” to help automation.
  - Tools readily available: https://github.com/spdx/tools-python

```
PackageName: FatPkg
SPDXID: SPDXRef-Package-FatPkg
PackageVersion: 1.0
PackageDownloadLocation: http://svn.insyde.com/
PackageSupplier: Organization: Insyde Software Corp.
(https://www.insyde.com/)
PackageVerificationCode: df94ca698b3e3ffa862b3cd5ac3d9568dfda10cd
PackageLicenseDeclared: BSD-2-Clause
PackageLicenseConcluded: BSD-2-Clause
PackageLicenseInfoFromFiles: BSD-2-Clause
PackageCopyrightText: <text>Copyright (c) 2012 - 2022, Insyde Software
Corp. All Rights Reserved.</text>

# File

FileName: \EDK2\FatPkg\EnhancedFatDxe\Data.c
SPDXID: SPDXRef-Data-src
FileType: SOURCE
FileChecksum: SHA1: d0cc1c226b572507bb3aafb5b4ef363fa0579404
LicenseConcluded: BSD-2-Clause
LicenseInfoInFile: BSD-2-Clause
FileCopyrightText: <text>Copyright (c) 2005 - 2013, Intel Corporation.
All rights reserved.</text>
```

Firmware Security Tools We Have SWID SBOMs

- SWID SBOMs focus on identity of production firmware binaries.

- SWID SBOMs can help IT/End-Users to
  - Inventory firmware on the platform (executables and blobs).
  - Check for security disclosures reported to affect that firmware.

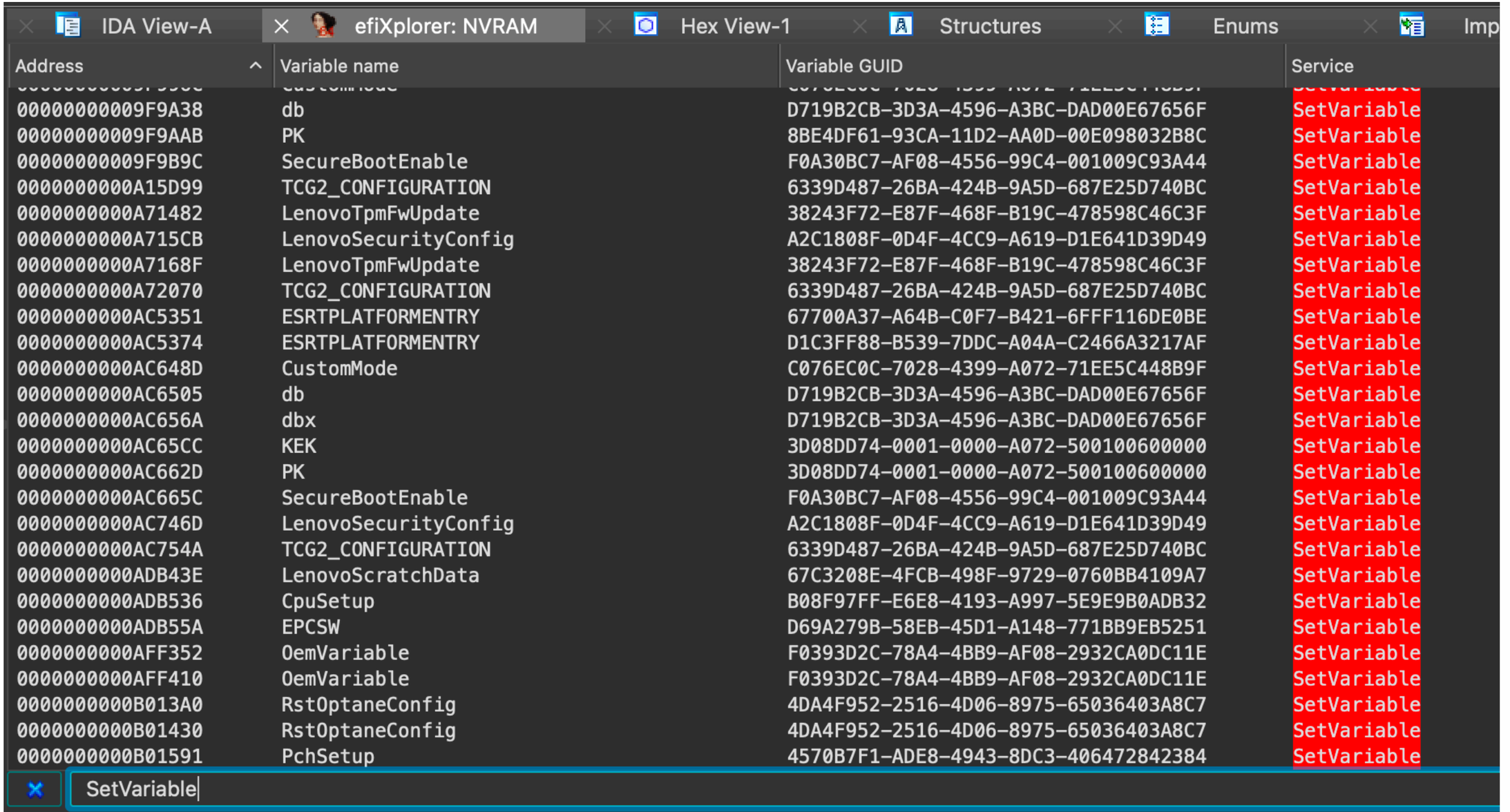
- UEFI firmware can:
  - Retrieve attached firmware information using DMTF's SPDM.
  - Validate firmware measurements against golden values.
  - Record measurements and identifiers in the TCG event log.

- See Traceable Firmware Bill of Materials Overview – UEFI 2021 Virtual Plugfest, python-uswid and LVFS.

Firmware Security Tools We Have

- **Intel's CHIPSEC (github.com/chipsec)**
  - Checks running configuration for vulnerable settings.
- **Binarly's efiXplorer (github.com/binarly-io/efiXplorer)**
  - Help investigate vulnerabilities in BIOS binaries.
- **Binarly's FwHunt (github.com/binarly-io/FwHunt)**
  - Checks BIOS binaries for known-bad code patterns (code semantic-based approach).

github.com/binarly-io/efiXplorer



github.com/binarly-io/FwHunt

```
BRLY-2022-004:
  meta:
    author: Binarly (https://github.com/binarly-io/FwHunt)
    license: CC0-1.0
    name: BRLY-2022-004
    namespace: vulnerabilities
    CVE number: CVE-2022-24419
    advisory: https://binarly.io/advisories/BRLY-2022-004/index.html
    description: SMM arbitrary code execution in USBRT SMM driver on Dell devices
    volume guids:
      - 04EAAAA1-29A1-11D7-8838-00500473D4EB
  code:
    and:
      - pattern: 488b05.....488b88....00004885c9....4883a0....000000....0fb704250e040000c1e00405040100008b084885c9
        place: sw_smi_handlers
```

github.com/binarly-io/FwHunt

```
SecureBackDoor-CVE-2021-3971:
  meta:
    author: Binarly
    name: SecureBackDoor-CVE-2021-3971
    namespace: vulnerabilities
    CVE number: CVE-2021-3971
    vendor id: LEN-73440
    advisory: https://github.com/eset/vulnerability-disclosures/blob/master/CVE-2021-3971/CVE-2021-3971.md
    description: Disabled SPI flash firmware storage protections
    volume guids:
      - 16F41157-1DE8-484E-B316-DDB77CB4080C
  wide_strings:
    and:
      - utf16le: cE!
  hex_strings:
    and:
      - 5de6cc6a35da394bb64b5ed927a7dc7e
```

fwhunt@binarly.io

github.com/binarly-io/FwHunt


```
BRLY-2021-011:
  meta:
    author: Binarly (https://github.com/binarly-io/FwHunt)
    license: CC0-1.0
    name: BRLY-2021-011
    namespace: vulnerabilities
    CVE number: CVE-2021-33627
    advisory: https://binarly.io/advisories/BRLY-2021-011/index.html
    description: SMM memory corruption vulnerability in combined DXE/SMM driver (SMRAM write)
    volume guids:
      - 74D936FA-D8BD-4633-B64D-6424BDD23D24
  variants:
    variant1:
      code:
        and:
          - pattern: 488b5310498d48204d8b4018e8....0000
            place: child_sw_smi_handlers
          - pattern: 4981392010000075
            place: child_sw_smi_handlers
    variant2:
      code:
        - pattern: 488b5310498d40204c8bc948894424..4533c033c9e8
          place: child_sw_smi_handlers
```

github.com/binarly-io/FwHunt

Next Steps

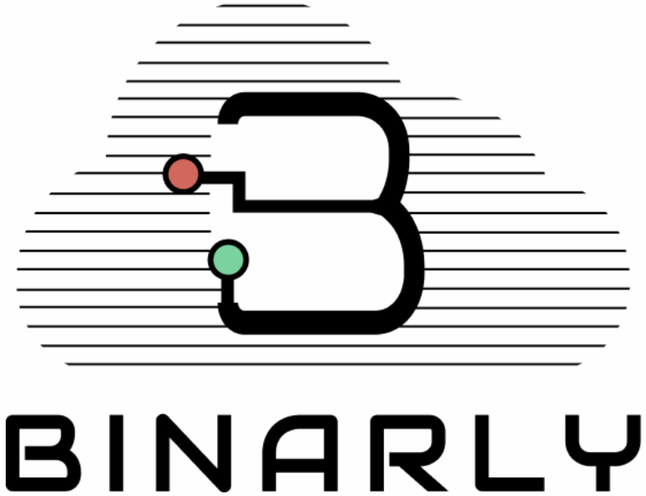
- For OEMs/ODMs, find the ingredients that make up your firmware.
- For IT/end-users, find out how to be notified of vulnerabilities in your firmware and its ingredients.
- At any stage of the supply chain, use tools to simplify vulnerability tracking on your platforms.

Questions?

Thanks for attending the UEFI 2022 Virtual Summit

For more information on UEFI Forum and UEFI Specifications, visit https://www.uefi.org

presented by



Reference – SBOMs, Tools, Gov't

- **Standards**
  - SPDX
  - SWID
- **Tools**
  - CHIPSEC
  - FwHunt
  - efiXplorer
- **Government**
  - ASSESSMENT OF THE CRITICAL SUPPLY CHAINS SUPPORTING THE U.S. INFORMATION AND COMMUNICATIONS TECHNOLOGY INDUSTRY

Reference – Binary SBOMs

- Traceable Firmware Bill of Materials Overview – UEFI 2021 Virtual Plugfest
- General Supply Chain Guidelines
  - ISO/IEC 28000:2007 - Specification for security management systems for the supply chain
  - NIST SP800-161 - Supply Chain Risk Management Practices for Federal Information Systems and Organizations
  - UK NCSC – Supply Chain Security Guideline
- Standards / Guidelines
  - NIST SP800-155 (draft) – BIOS Integrity Measurement Guideline
  - TCG PC Client Platform Firmware Profile (PFP)
  - TCG PC Client Firmware Integrity Measurement (FIM)
  - TCG PC Client Reference Integrity Manifest (RIM)
  - TCG Platform Certificate Profile
  - TCG DICE Attestation Architecture
  - TCG DICE Layering Architecture
  - TCG DICE Certificate Profile
  - IETF RATS Remote Attestation Architecture
  - IETF SACM Concise SWID
  - IETF RATS Concise RIM
  - DMTF Secure Protocol and Data Model