

MICROSOFT SECURITY

Locked Down but Not Out

Fighting the Hidden War in Your Bootloader

Bill Demirkapi



Who Am I?



- Lead Emerging Threats at the Microsoft Security Response Center.
- Background in low-level OS internals and cloud security.
- Worked with Secure Boot for over a year.



Introduction to UEFI Secure Boot

- What is UEFI?
- What is Secure Boot?
- How does it work?
- Why should I care?

What is Secure Boot?

- BIOS only offered a custom password as an access control.
- UEFI **Secure Boot** is a feature that enforces trust for boot code.
- Boot loaders or UEFI drivers must be signed by a trusted authority.

How do we know what to trust?

UEFI Variables

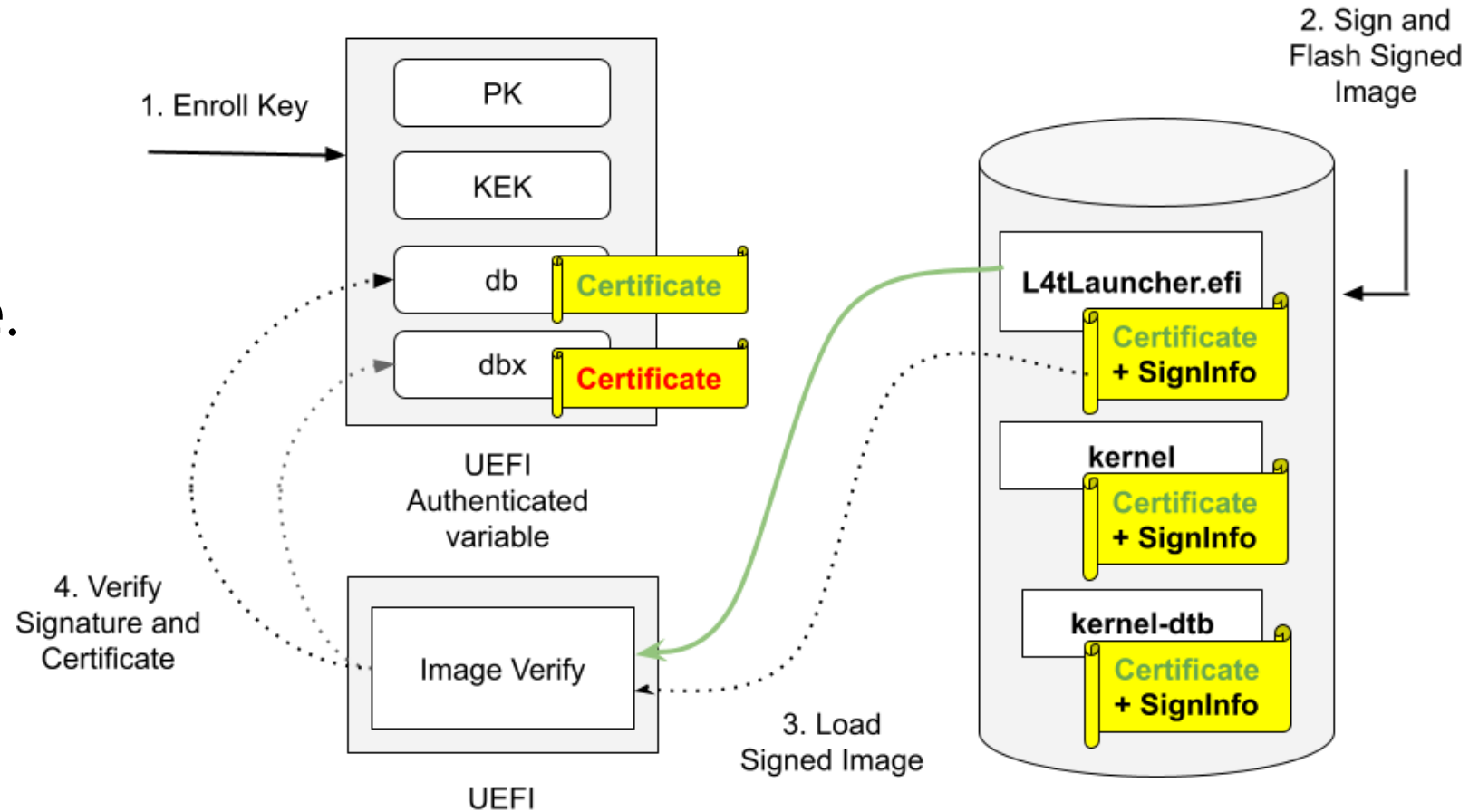
- UEFI Runtime Variables allow the firmware and OS to share data.
- How much data? Until Windows 10 build 1709, ~**32 KB**.

Table 3-1 Global Variables

Variable Name	Attribute	Description
Boot####	NV, BS, RT	A boot load option. #### is a printed hex value. No 0x or h is included in the hex value.
BootOrder	NV, BS, RT	The ordered boot option load list.
KEK	NV, BS, RT,AT	The Key Exchange Key Signature Database.
PK	NV, BS, RT,AT	The public Platform Key.
db	BS, RT	The OEM's secure boot signature store.
dbx	BS, RT	The OEM's secure boot blacklist signature store.

How Do We Define Trust?

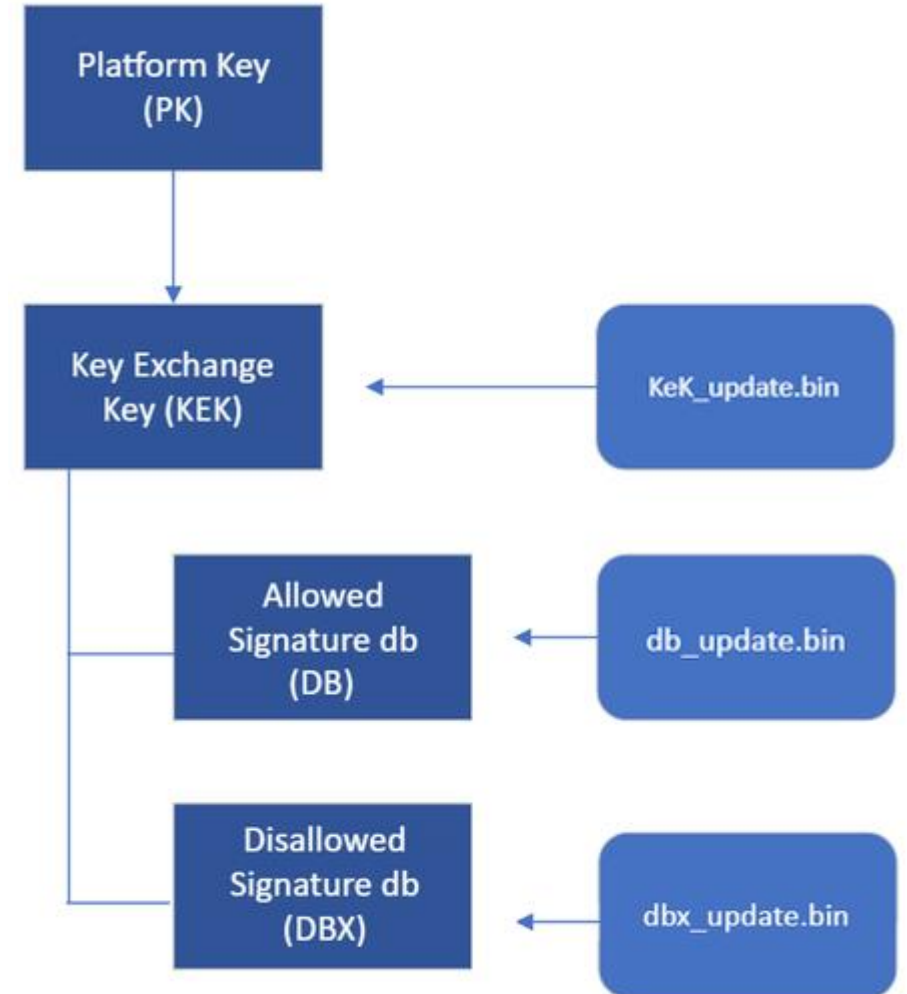
- Images are signed by a **DB** certificate.
- To modify **DB**, you need a **KEK** signature.
- To modify **KEK**, you need a **PK** signature.
- To **revoke** trust, we use **DBX**.



Source: NVIDIA Jetson Linux

DB and DBX

- DB & DBX are **authenticated** variables that control what can load.
- Both allow hashes or certificates.
- By default, DB authorities include...
 - Microsoft Production Certificate Authority
 - UEFI Third-Party Certificate Authority
- Updates are usually "append write".



Why is DBX Critical to Security?



Since Secure Boot is intended to stop unauthorized code, it needs to stop far more than unsigned images.



Attackers can use memory corruption vulnerabilities too!



DBX is how we **revoke** the trust of vulnerable images.

Why Does It Matter?

- Boot integrity is critical for establishing a chain of trust.
- Prevents malicious code from hiding in your boot environment.
- Enables important features like disk encryption.

 **Windows**
BitLocker



Secure Boot's Threat Model

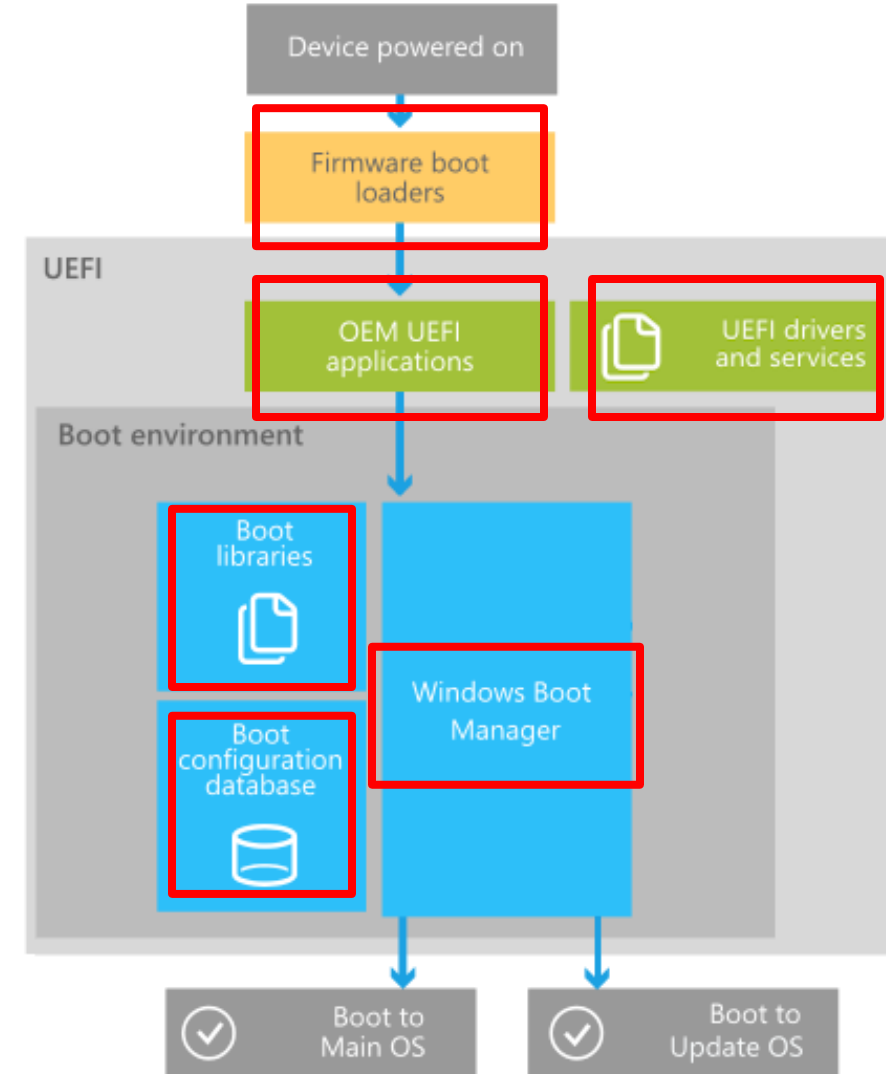
- What threats are relevant to Secure Boot?
- What are its attack surfaces?
- What can we reasonably defend against?



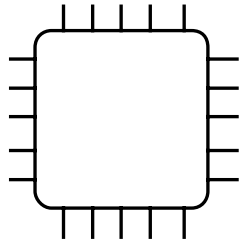
Secure Boot's Unique Challenge

- Windows has several security *boundaries*.
- Boundaries are a logical separation between the code & data of different trust levels.
 - **Boundary:** A *non-admin* user cannot access or tamper with kernel code and data.
 - **Not a boundary:** An *admin* user can tamper with kernel code and data.
- Secure Boot is a security *feature* Microsoft intends to service.
- **It includes Administrators in its threat model.**

What Happens When You Boot Your Computer?



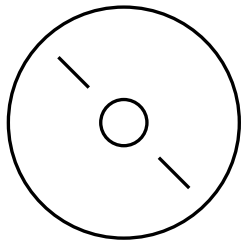
Dissecting Boot Attack Surfaces



Firmware

- Everyone has a fork.
- Decentralized updates.
- Unverifiable supply chain.

Adds risk



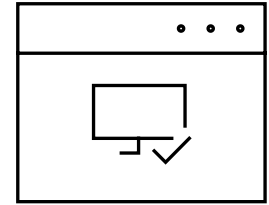
Disk

- Can't trust any file.
- Why revocations exist.
- Can be local or **remote**.

Adds risk



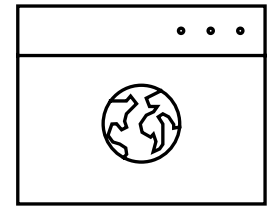
Adds risk



Microsoft Images

- Many unaudited features.
 - "By design" risks.
- Exposed to 3P by default.

Adds risk

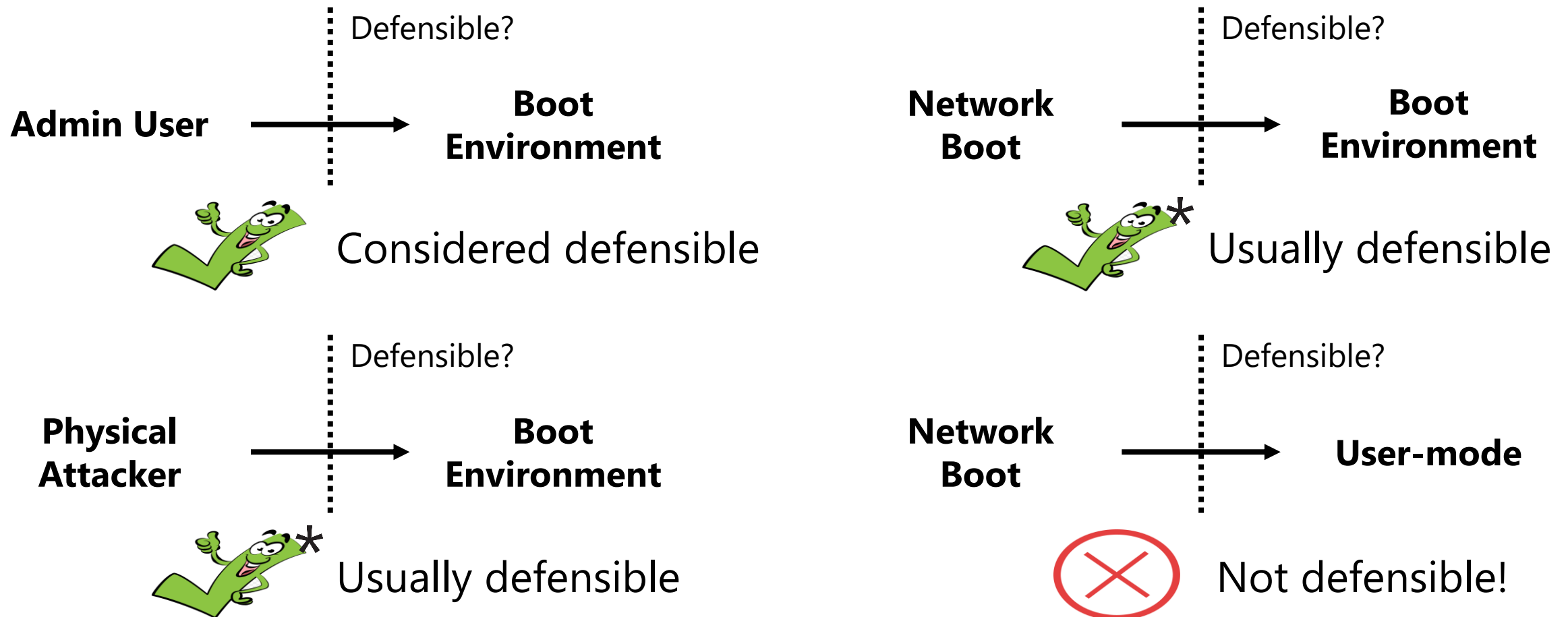


Third-Party Images

- **15,000+** signed binaries.
 - Lack of visibility.
- Exposed to 1P by default.

Where Does Microsoft Draw the Line?

Can an attacker achieve the same outcome by design?



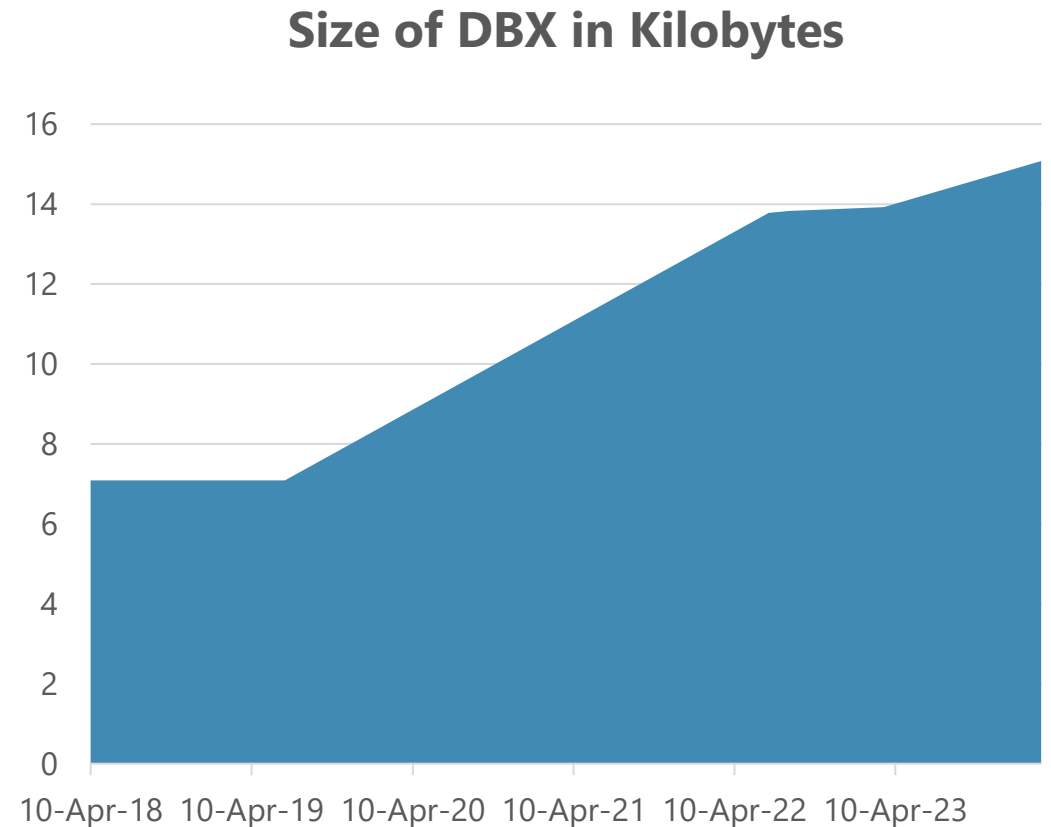


UEFI Architectural Faults

- What makes Secure Boot so hard to defend?
- What are examples of these challenges?
- Why do they exist?

Overview

- Remember – to fix UEFI bugs, we not only need to fix the code, but prevent the old code from executing.
- The UEFI standard says to add bad hashes or certificates to DBX.
- DBX **only has ~32 KB of space** = ~600 to ~800 unique hashes.
- Revoking isn't always possible.
 - What about users running old code?



UEFI Certificate Authorities

- The Production CA is used to sign first-party boot code.
- Microsoft has a lot of products, features, and extensions.
- **Over 10,000 boot images**, many sharing code...
- Third-Party UEFI Certificate Authority is designed to allow organizations to run boot code on Windows machines.
- **Over 16,000 boot images...**

Size: 12.4 GB (13,381,272,399 bytes)

Size on disk: 12.4 GB (13,415,784,448 bytes)

Size of All UEFI CA Images

Volume & Compatibility

- Will all 30,000+ signed images have vulnerabilities? Probably not.
- It is hard to be safe in UEFI when everything is an attack surface.
- A vulnerability in one image is rarely limited to one image.
 - A vulnerability in a library can impact hundreds, if not thousands of images.
 - You not only need to revoke the latest vulnerable version, but every version before it!
- When you revoke, you also break any existing use cases.
- **DBX works on paper, fails in practice.**

Windows: BlackLotus

- ESET published a report about an in-the-wild **bootkit**, BlackLotus.
- BlackLotus abused CVE-2022-21894, a fixed vulnerability. How?

ESET RESEARCH

BlackLotus UEFI bootkit: Myth confirmed

The first in-the-wild UEFI bootkit bypassing UEFI Secure Boot on fully updated UEFI systems is now a reality

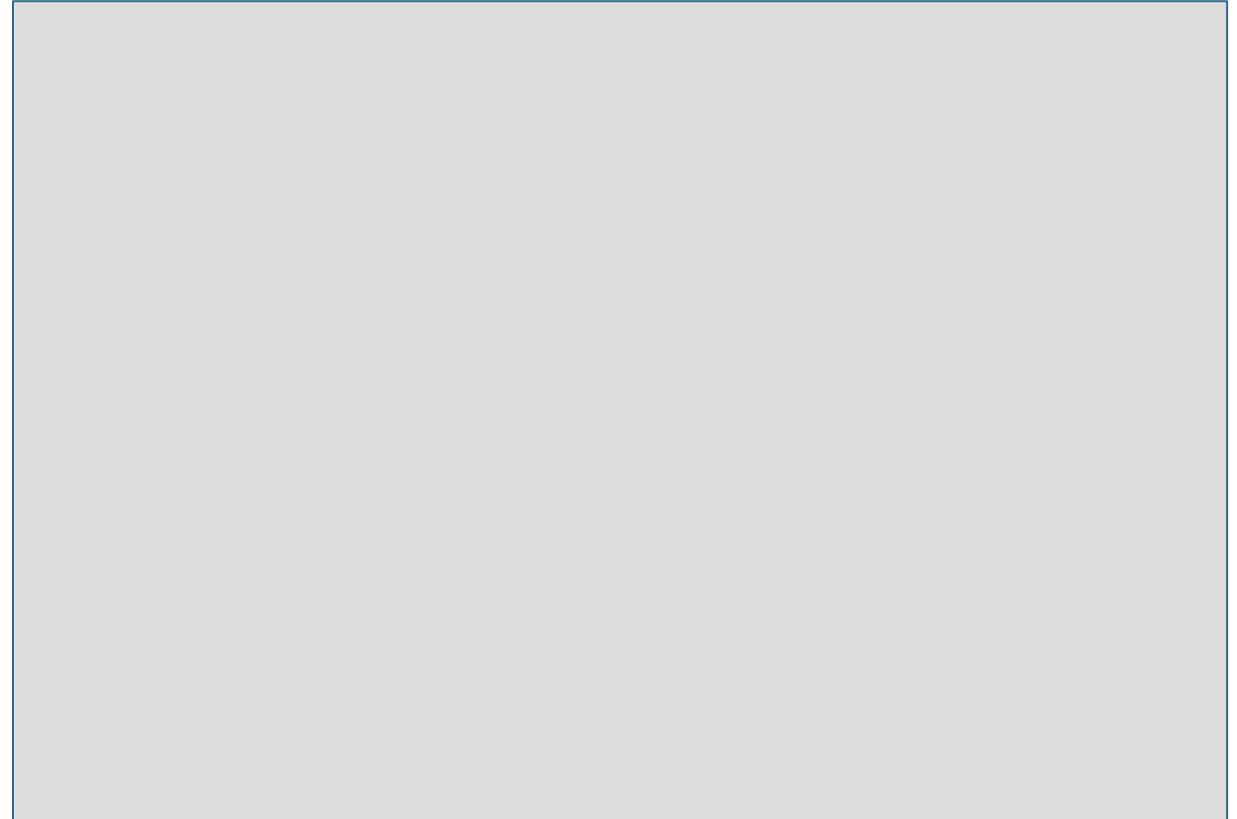
Martin Smolár

01 Mar 2023 , 40 min. read

- It's capable of running on the latest, fully patched Windows 11 systems with UEFI Secure Boot enabled.
- Although the vulnerability was fixed in Microsoft's January 2022 update, its exploitation is still possible as the affected, **validly signed** binaries have still not been added to the [UEFI revocation list](#). BlackLotus takes advantage of this, bringing its own copies of legitimate – but vulnerable – binaries to the system in order to exploit the vulnerability.

Windows: BlackLotus

- Fixing vulnerabilities isn't enough; we need to revoke!
 - **One bug = thousands of executables!**
- Microsoft released an opt-in *mitigation* for BlackLotus in May 2023.
- We are still working on a **default** secure revocation story.



Learn more at <https://aka.ms/CVE-2023-24932-guidance>

Linux Shim: Remote Code Execution Vulnerability

- The Linux shim is the first-stage boot loader for effectively every distribution that supports Secure Boot.
- It is designed to load secondary drivers, like GRUB2, by manually verifying and mapping them to memory.

shim, a first-stage UEFI bootloader

shim is a trivial EFI application that, when run, attempts to open and execute another application. It will initially attempt to do this via the standard EFI `LoadImage()` and `StartImage()` calls. If these fail (because Secure Boot is enabled and the binary is not signed with an appropriate key, for instance) it will then validate the binary against a built-in certificate. If this succeeds and if the binary or signing key are not forbidden then shim will relocate and execute the binary.

Linux Shim: Remote Code Execution Vulnerability

- We found a critical vulnerability in the decade-old loader.
- Allows remote or local attackers to execute code.
- Fixed™ in January 2024.
- Revocation date? N/A

CVE-2023-40547 - avoid incorrectly trusting HTTP headers

When retrieving files via HTTP or related protocols, shim attempts to allocate a buffer to store the received data. Unfortunately, this means getting the size from an HTTP header, which can be manipulated to specify a size that's smaller than the received data. In this case, the code accidentally uses the header for the allocation but the protocol metadata to copy it from the rx buffer, resulting in an out-of-bounds write.

This patch adds an additional check to test that the rx buffer is not larger than the allocation.

Resolves: [CVE-2023-40547](#)

Reported-by: Bill Demirkapi, Microsoft Security Response Center

Signed-off-by: Peter Jones <pjones@redhat.com>

Linux Shim: Remote Code Execution Vulnerability

- Linux doesn't need DBX. They have a custom revocation scheme.
 - Secure Boot Advanced Targeting (SBAT) revokes by version, rather than hash.
 - Unfortunately, SBAT is an unauthenticated UEFI variable, allowing tampering (unlike DBX).
- They still can't revoke. Why? Too many users with the old shim!

```
sbat,1,SBAT Version,sbat,1,https://github.com/rhboot/shim/blob/main/SBAT.md  
grub,2,Free Software Foundation,grub,2.04,https://www.gnu.org/software/grub/
```

Example SBAT Revocation

Recap

- The revocation challenge is an ecosystem problem.
 - For some, it's about not having enough space for every vulnerable image.
 - For some, it's about breaking legitimate use cases.
- UEFI exposes an enormous attack surface and forces developers to shoot themselves in the leg to secure it.
- **Result:** Substantially reduces defender response capability

Overview

- The attack vectors from the threat model introduce enormous risk.
- In UEFI, almost anything configurable is an attack surface.
- Every “feature” in every boot image is more surface!

UEFI Certificate Authority Surfaces

- Over 16,000 signed, but unaudited boot images.
- Sometimes justified use cases!
 - Linux is a critical part of the ecosystem.
 - Option ROMs are often necessary for modern hardware.
- Sometimes... less justified...
 - Debug-related images, like UEFI shells, that are vulnerable by design.
 - Random utilities no one really uses.
- Remember – customers are exposed to them *by default*.

Windows Production Certificate Authority Surfaces

- Even though we sign first-party code, Windows is one of the largest operating systems.
- Inherently, it includes several features for extensibility.
- How many of these options will most customers really need?
- Remember – customers are exposed them *by default*.

Why Do We Expose Everything By Default?

A vast majority of attack surfaces won't be relevant for a vast majority of consumers.

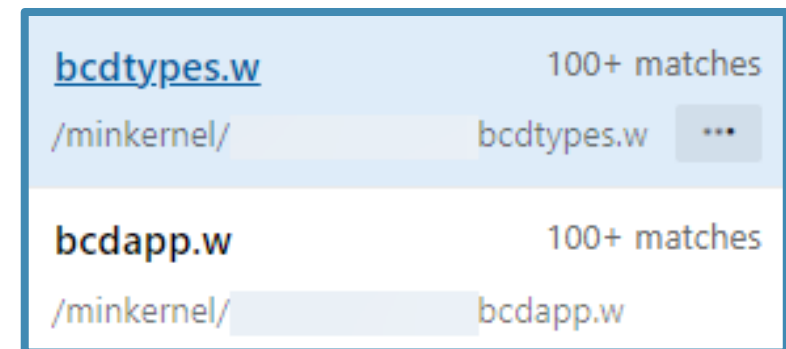
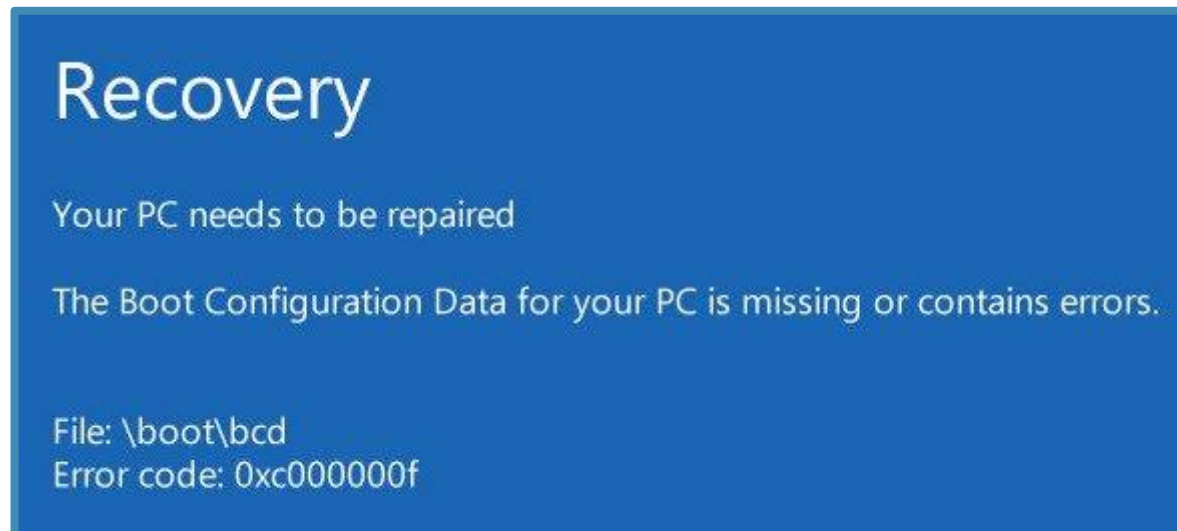
- Does every single Windows computer in the world really need to be able to run Joe Shmoe's Disk Recovery Utility by default?
- Do all Linux users, who just want to use Linux, really need to be exposed to Microsoft's attack surface by default?

Windows Boot Code

- Thanks to our incredible researcher community, we have fixed several vulnerabilities in Microsoft's boot code.
- Without spoiling their findings, let's review a few examples of the common types of vulnerabilities we've seen!

Windows Boot Code: Configuration Data

- The Windows boot manager is an extensible platform that supports launching several OS applications.
- Boot options are configured via the Boot Configuration Data (BCD).
- BCD files are registry “hives” that store all platform-specific options.



Search for unique boot config options

Windows Boot Code: Lack of Input Validation

- Developers can easily forget everything configurable is untrustworthy.
- Can you spot the bug?

```
NTSTATUS
BlGetConfiguredDeviceStruct (
    _Out_ PCUSTOM_DEVICE_STRUCT DeviceInfo
)
{
    NTSTATUS Status;
    PBOOT_ENVIRONMENT_DEVICE Device; // untrusted device
    PCUSTOM_DEVICE_STRUCT LocalInfo; // untrusted struct from device

    // Retrieve the configured device (untrusted).
    Status = GetDeviceFromBCD(BCD_TYPE_MY_IMPORTANT_DEVICE, &Device);

    if (NT_SUCCESS(Status))
    {
        // Retrieve pointer to custom info (untrusted).
        LocalInfo = &Device->u.CustomInfo;

        // Allocate space for a copy.
        *DeviceInfo = malloc(sizeof(CUSTOM_DEVICE_STRUCT)
                             + MAX_CUSTOM_SIZE);

        if (*DeviceInfo == NULL)
        {
            return Status;
        }

        // Copy the custom structure.
        memcpy(*DeviceInfo, LocalInfo, LocalInfo->Size);
    }

    return Status;
}
```

Windows Boot Code: Extensibility

- Another major threat for boot code is extensibility.
- For example, did you know some variants of boot manager support 10+ unique filesystems?
- Why do we expose this by default?

```
const PFILESYSTEM_TABLE FsTable[] = {  
    &NetRegisterFunctionTable,  
    &CompositeFsRegisterFunctionTable,  
    &VmbfsRegisterFunctionTable,  
    &CimFSRegisterFunctionTable,  
    &NtfsRegisterFunctionTable,  
    &EfiFsRegisterFunctionTable,  
    &FatRegisterFunctionTable,  
    &RefsRegisterFunctionTable,  
    &FppRegisterFunctionTable,  
    &WimRegisterFunctionTable,  
    &UdfsRegisterFunctionTable,  
    &EtfsRegisterFunctionTable,  
    NULL  
};
```

UEFI CA

- MSRC obtained 15,000+ UEFI CA images for review.
- We found many trivial problems.
- Let's go through a few examples.

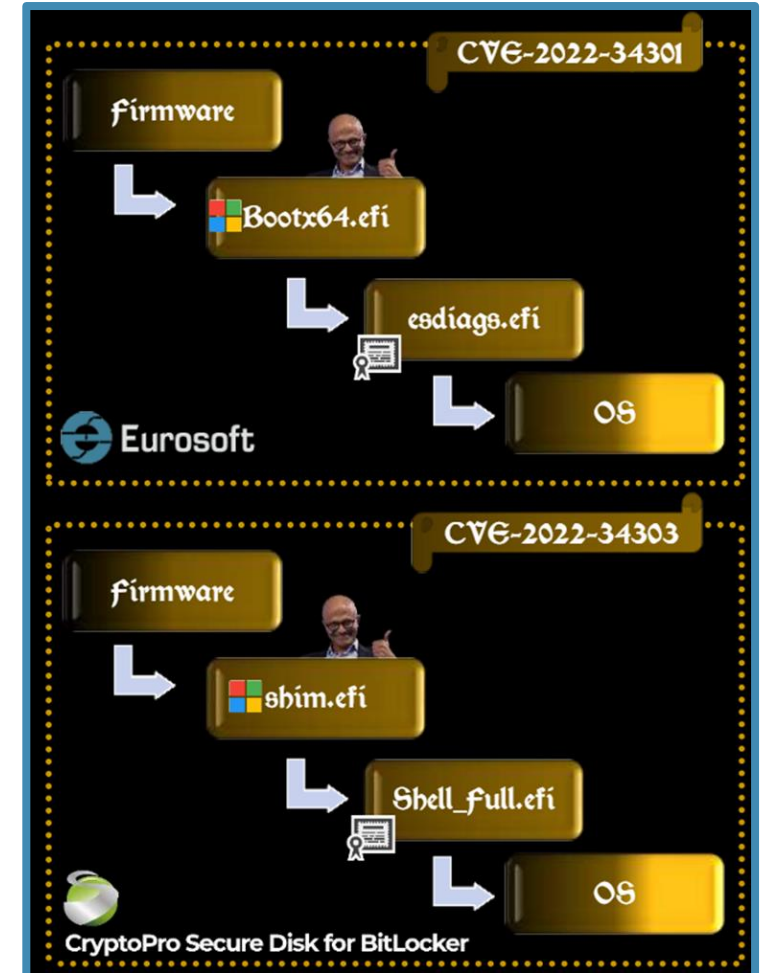
Watch [Dissecting Secure Boot's Third-Party Attack Surface](#)
for a dedicated review of the third-party story!

UEFI CA: Unrevoked Variants of Revoked Images

- One trivial path to finding UEFI CA zero days was taking revoked images and looking for variants.
- Remember – you need to revoke every version of a vulnerable image, not just the latest!

Search results						
Name	Size	Matches	Path	Ext	Encoding	Date modified
Bootauth32_1.efi	397 KB	2	\.	efi	BINARY	8/6/2014 2:57:32 PM
Bootauth32.efi	397 KB	2	\.	efi	BINARY	8/8/2014 3:20:20 PM
Bootauth.efi	1.00 MB	1	\.	efi	BINARY	10/22/2020 5:48:00 PM
Bootauth_1.efi	1.00 MB	1	\.	efi	BINARY	4/8/2020 8:09:02 PM
Bootauth_2.efi	951 KB	1	\.	efi	BINARY	7/3/2019 8:30:30 PM
Bootauth_29.efi	1.34 MB	1	\.	efi	BINARY	12/21/2012 2:46:46 PM

Searched 15430 files, skipped 0 files. Found 20 matches in 18 files.



Source: *One Bootloader To Rule Them All*
by Eclyspium

UEFI CA: Linux Shim Attack Surfaces

- Remember the Linux shim? It's designed to load arbitrary boot code signed by individual distributions. **No audit log. No visibility.**

— Fedora Secure Boot CA	
Name	Fedora Secure Boot CA
Issuer	Fedora Secure Boot CA
Valid From	2012-12-07 16:25:54
Valid To	2022-12-05 16:25:54
Algorithm	sha256RSA
Thumbprint	7E68651D52685F7BF58EA01D784D2F90D3F40F0A
Serial Number	99 76 F2 F4

- Fedora used the same certificate from 2012 to late 2023.
- Allowed attackers to abuse vulnerabilities from 2015-2020.

UEFI CA: Linux Shim Extensibility

- Remember MSRC's shim vulnerability?
- Can you spot the bug?

```
static EFI_STATUS
receive_http_response(
    EFI_HTTP_PROTOCOL *http,
    VOID **buffer,
    UINT64 *buf_size
)
{
    // http request code ...

    /* Check the length of the response */
    for (i = 0; i < rx_message.HeaderCount; i++) {
        if (!strcasecmp(rx_message.Headers[i].FieldName,
            (CHAR8 *)"Content-Length")) {
            *buf_size = ascii_to_int(rx_message.Headers[i].FieldValue);
        }
    }

    if (*buf_size == 0) {    } // exit if 0

    // Allocate buffer based on content length
    *buffer = AllocatePool(*buf_size);
    if (!*buffer) {    } // exit if NULL

    // Copy current chunk to buffer
    downloaded = rx_message.BodyLength;
    CopyMem(*buffer, rx_buffer, downloaded);
}
```

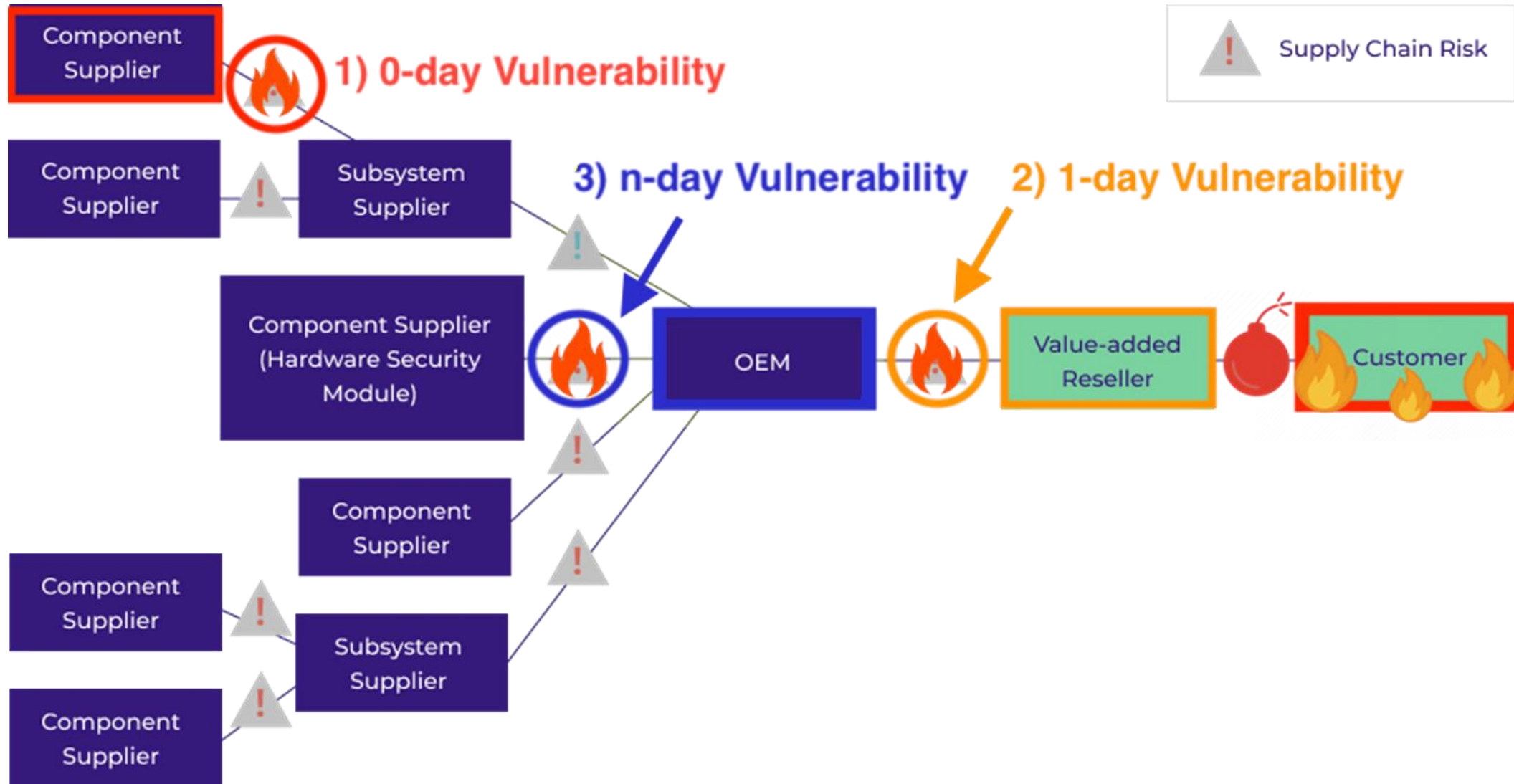

Recap

- It is really hard to write secure code for UEFI, where everything is an attack surface.
- The status quo includes an enormous amount of “by default” attack surface that most users will never need.
- Balancing customer choice with security is non-trivial.

Overview

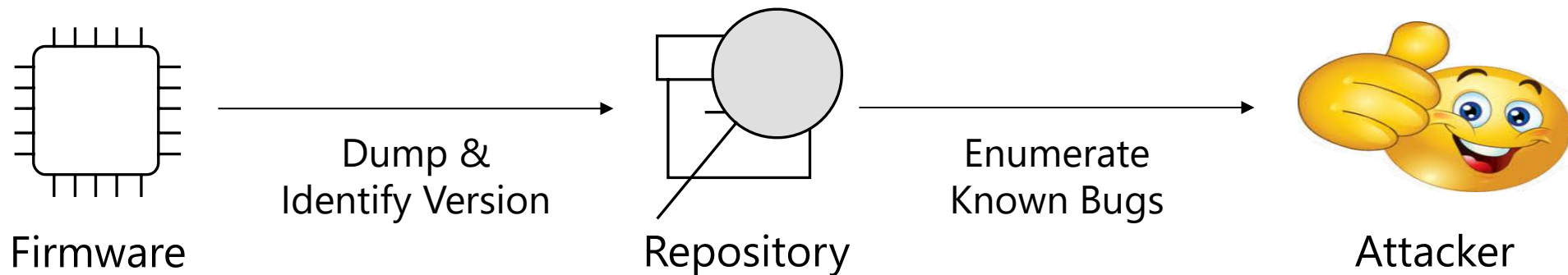
- Users are only as secure as the weakest link.
- How can Microsoft maintain a strong security story given partner-introduced risk?
 - MSRC has observed an inconsistent commitment to customer security in the ecosystem.
 - Some partners, like the Linux community, take security seriously.
 - Others... not so much...

You Are Only As Secure As Your OEM's Partner's Partner



You Are Only As Secure As Your OEM's Partner's Partner

- One problem identified by researchers is how the complexity of the firmware supply chain leaves severe vulnerabilities exploitable.


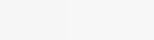
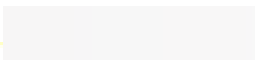
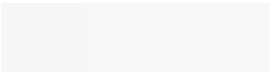


Read more: *The Firmware Supply-Chain Security is broken: Can we fix it?* by Binarly

You Are Only As Secure As Your OEM's Partner's Partner

- There are dozens of fixed boot vulnerabilities abusable via rollback.
- Why haven't we revoked old versions?
- Revoking means blocking every first-party image we have signed.
- Good news! We were already planning to roll the CA in 2025-2026.
 - Why not expedite this process?

Microsoft will be rolling out Secure Boot database updates in phases to add trust for the new DB and KEK certificates. The first DB update will add the **Microsoft Windows UEFI CA 2023** to

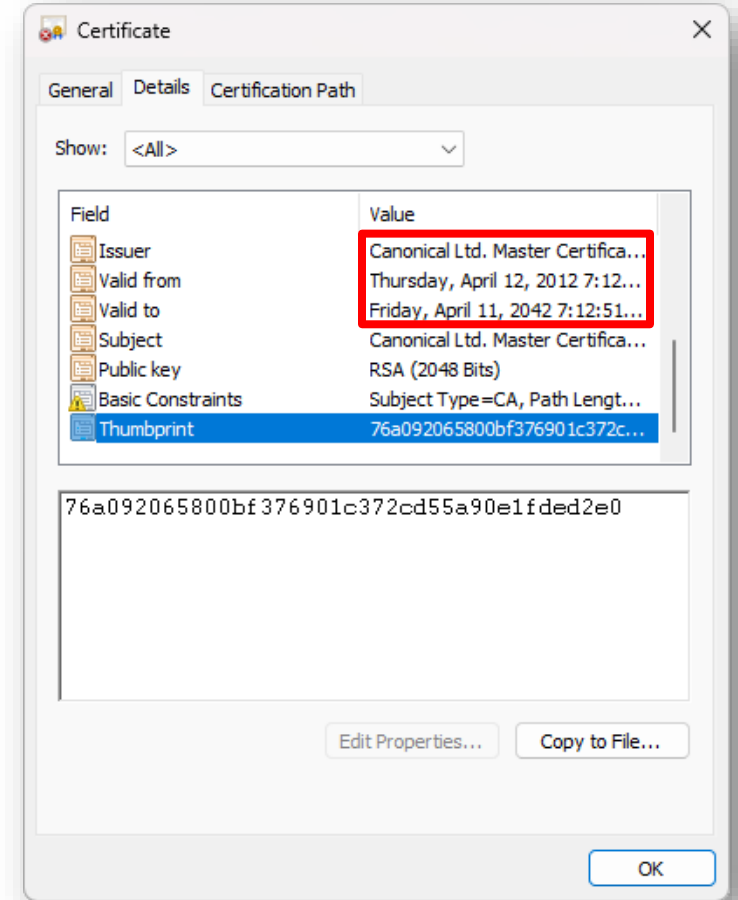
the s  BUG 
com   are bricking with SB on, after taking a DB Update wly

roll out this DB update as we validate devices and firmware compatibility glo



You Are Only As Secure As Your OEM's ~~Partner's Partner~~

- The Platform Key (**PK**) controls the Key Exchange Key (**KEK**) which controls the allowed signatures database (**DB**).
- Many OEMs choose to manage their own keys.
- Unfortunately, they aren't always maintained.



Vulnerable Canonical Certificate
Included By Some OEMs

You Are Only As Secure As Your OEM's ~~Partner's Partner~~

- Researchers from Binarly recently discovered that several major OEMs have the PK named, "`CN=DO NOT TRUST - AMI Test PK`".
- The AMI Test key was stored as a PFX file included in UEFI test packages, which were leaked.
- With PK, attackers can bypass Secure Boot by updating KEK/DB.



FACEPALM GOES HERE —

Secure Boot is completely broken on 200+ models from 5 big device makers

Keys were labeled "DO NOT TRUST." Nearly 500 device models use them anyway.

Read more: *PKfail: Untrusted Platform Keys Undermine Secure Boot on UEFI Ecosystem* by Binarly

Recap

- Microsoft will work with partners to challenge UEFI's status quo.
- The decentralized nature of the ecosystem presents a severe threat to customer security.
- Balancing customer choice with security is non-trivial.
- Users are only as secure as the weakest link.

How Microsoft is Fixing It Anyways

- What are we doing for Secure Boot?
- How are we working with partners?
- What can you do to protect your organization?



Unified Interface, Divided Implementation

- While a decentralized ecosystem is great for customer choice, it's not so good for customer security.
- Industry response time to UEFI bugs highlights inefficiency.
- UEFI may be a unified interface, but we need unified execution.
 1. Use Windows Updates to deploy UEFI firmware!
 2. Collaborate on a secure implementation; Don't fork!
 3. Invest in firmware transparency & auditability.

Future Facing: Supporting Customer Security

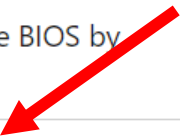
- How do we empower partners to value customer security?
 1. Prioritizing customer security with firm deadlines.
 2. Partners who are prepared will be added on an ongoing basis.
 3. Customers receive protection as soon as feasible – rather than waiting for complete participation.

Secured Core Attack Surface Reduction

- Special line of Windows computers with strengthened defaults.
- On laptops, UEFI CA is not included by default.
 - Depends on whether Option ROMs are required.

What makes a Secured-core PC

Benefit	Feature	Hardware/Firmware requirement	Baseline Windows Security	Secured-core PCs
Create a hardware backed root of trust				
	Secure Boot	Secure Boot is enabled in the BIOS by default.	✓	✓
	Secure Boot	3rd party UEFI CA not trusted by default, with BIOS option for enabling trust		✓



Implemented: What Do We Do About DBX?

- Revocation via Embedded Secure Version Information (**REVISE**)
- MSRC design to **extend DBX** and allow revocation by version.
 - How? We can revoke any hash we want via DBX.
 - SHA-256 hashes have 32 bytes of space.
 - What if we included version data in a “fake hash” that only our code recognized?
- We still run into DBX space limitations, but with one hash entry, we can revoke thousands of images by version.

```
#pragma pack(1)
typedef struct _EFI_SIGNATURE_DATA {
    EFI_GUID          SignatureOwner;
    UINT8             SignatureData[...];
} EFI_SIGNATURE_DATA;
```

Use special GUID to mark entries containing version data.

Fake hash with version data.

Implemented: What Do We Do About DBX?

- Revocation via Embedded Secure Version Information (REVISE)
- MSRC de
 - How? W
 - SHA-256
 - What if
- We still r
 - can revo

```
#pragma pack(1)
typedef struct {
    EFI_GUID ApplicationGuid;
    UINT8 Signature[32];
} EFI_SIGNATURE_DATA;
```

```
result = DbxFetchSvn((EFI_GUID *) &ApplicationGuidVal, &SvnData);
if ( result >= 0 )
{
    BinMajorVer = HIWORD(BinSvnVer);
    MinMinorVer = SvnData.MinorSvn;
    DbxSvn = SvnData;
    if ( HIWORD(BinSvnVer) < SvnData.MajorSvn )
    {
        DbxSvnEfiConOut->ClearScreen(DbxSvnEfiConOut);
        DbxSvnEfiConOut->SetAttribute(DbxSvnEfiConOut, 4ui64);
        DbxSvnPrintf(
            L"Security Error: Secure boot version check failed.\r\n"
            "Your system security may be compromised!\r\n"
            "\r\n"
            "Current version: %lu.%lu - Minimum allowed version: %lu.%lu\r\n"
            "Visit https://aka.ms/secure-boot-version-violation for more information.\r\n"
            "\r\n",
            BinMajorVer,
            BinMinorVer,
            DbxSvn.MajorSvn,
            MinMinorVer);
    }
}
```

Released in April!

Implemented: Support for Linux Revocations

- Remember the Linux shim vulnerability and a lack of revocations?
- Patch Tuesday includes a boot loader with latest Linux revocations!
 - Revocations provided by the Linux community in the shim repository.
- Only applies to machines without Linux as a second boot option.

19.4 Embedded information for generation number based revocation

The Secure Boot Advanced Targeting (SBAT) is a mechanism to allow the revocation of components in the boot path by using generation numbers embedded into the EFI binaries. The SBAT metadata is located in an .sbat data section that has set of UTF-8 strings as comma-separated values (CSV). See

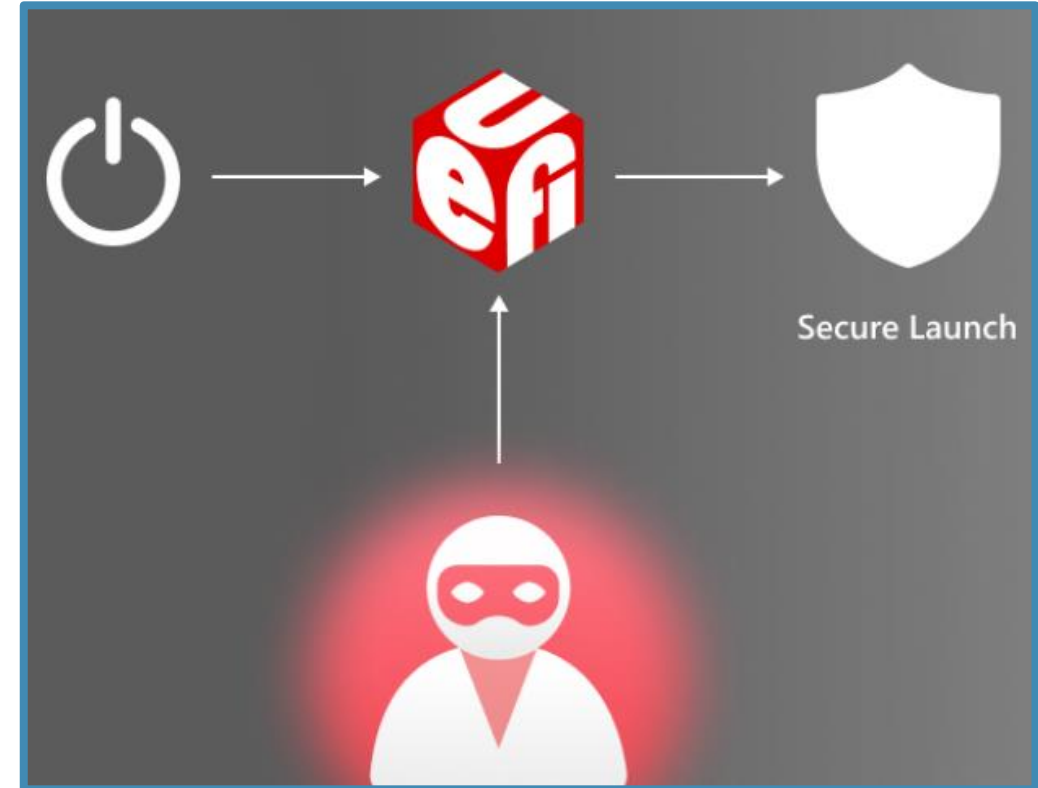
<https://github.com/rhboot/shim/blob/main/SBAT.md> for more details.

Partnering With the Community

- Partner with researchers to find problematic surfaces and hold us accountable.
- Partner with OEMs and share the right tools to balance a secure outcome with compatibility.
- Partner with other organizations, like CISA, to help push the industry in the right direction.
- Partner with the Linux community to maximize choice without forgetting security.

Future Facing: Measured Boot & Disk Encryption

- One area we didn't cover is Measured Boot, which uses TPMs to protect important secrets behind trusted "measurements".
- BitLocker is a good example.
- Long-term investments, like Dynamic Root of Trust for Measurement, are promising for the future.



What Can You Do to Protect Your Organization?

- You can leverage BitLocker to stop many physical attacks.
 - Use TPM+PIN for maximum security.
- Secured Core PCs reduce third-party attack surfaces*.
- Apply opt-in protections from Microsoft.
- Advanced organizations that need the best may consider managing their own certificates.



**UEFI Secure Boot
Customization**

Concluding Thoughts



The Elephant in the Room

- We keep focusing on short-term fixes.
- Secure Boot needs an overhaul to remain defensible.
- We must work together.

MICROSOFT SECURITY



Questions?