THERE'S A HOLE IN THE BOOT

"BootHole" vulnerability in the GRUB2 bootloader opens up Windows and Linux devices using Secure Boot to attack. All operating systems using GRUB2 with Secure Boot must release new installers and bootloaders.

Eclypsium researchers have discovered a vulnerability — dubbed "BootHole" — in the GRUB2 bootloader utilized by most Linux systems that can be used to gain arbitrary code execution during the boot process, even when Secure Boot is enabled. Attackers exploiting this vulnerability can install persistent and stealthy bootkits or malicious bootloaders that could give them near-total control over the victim device.

The vulnerability affects systems using Secure Boot, even if they are not using GRUB2. Almost all signed versions of GRUB2 are vulnerable, meaning virtually every Linux distribution is affected. In addition, GRUB2 supports other operating systems, kernels and hypervisors such as Xen. The problem also extends to any Windows device that uses Secure Boot with the standard Microsoft Third Party UEFI Certificate Authority. Thus the majority of laptops, desktops, servers and workstations are affected, as well as network appliances and other special purpose equipment used in industrial, healthcare, financial and other industries. This vulnerability makes these devices susceptible to attackers such as the threat actors recently discovered using malicious UEFI bootloaders.

Eclypsium has coordinated the responsible disclosure of this vulnerability with a variety of industry entities, including OS vendors, computer manufacturers, and CERTs. Mitigation will require new bootloaders to be signed and deployed, and vulnerable bootloaders should be revoked to prevent adversaries from using older, vulnerable versions in an attack. This will likely be a long process and take considerable time for organizations to complete patching.

TABLE OF CONTENTS

Background: Secure Boot, GRUB2, and CAs

Secure Boot can be a fairly deep and technical topic. Our goal here is to give a high-level introduction to the key concepts relevant to this research without going into all the granular details. We have included a variety of external links to provide additional information for those interested. Alternatively, you can go straight to the description of the vulnerability itself.

THREATS TO THE BOOT PROCESS

The boot process is one of the most fundamentally important aspects of security for any device. It relies on a variety of firmware that controls how a device's various components and peripherals are initialized and ultimately coordinates the loading of the operating system itself. In general, the earlier code is loaded, the more privileged it is.

If this process is compromised, attackers can control how the operating system is loaded and subvert all higher-layer security controls. Recent research has identified ransomware in the wild using malicious EFI bootloaders as a way to take control of machines at the time of boot. Previously threat actors used malware tampering with legacy OS bootloaders including APT41 Rockboot, LockBit, FIN1 Nemesis, MBR-ONI, Petya/NotPetya, and Rovnix.

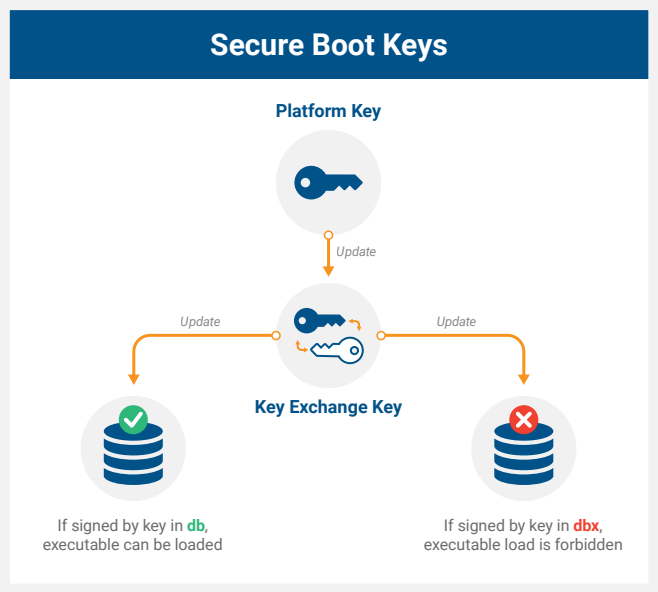
Additional information on threats to the modern PC boot process is available in the "Bootkits and UEFI Secure Boot" section of the System Firmware training.

UEFI SECURE BOOT

UEFI Secure Boot was originally developed by the UEFI Forum as a way to protect the boot process from these types of attacks. There are other implementations of secure boot designed for different environments, but UEFI Secure Boot is the standard for PCs and servers. The goal is to prevent malicious code from being introduced into the boot process by cryptographically checking each piece of firmware and software before it is run. Any code not recognized as valid is not executed in the boot process.

Secure Boot uses cryptographic signatures to verify the integrity of each piece of code as it is needed during the boot process. There are two critical databases involved in this process: the Allow DB (db) of approved components and the Disallow DB (dbx) of vulnerable or malicious components, including firmware, drivers, and bootloaders. Access to modify these databases is protected by a Key Exchange Key (KEK), which in turn is verified by a Platform Key (PK). Although the PK is used as a root of trust for updates to the platform, it's not expressly part of the boot process (but is shown below for reference). It is dbx, db, and KEK that are used to verify the signatures for loaded executables at boot time.

Additional details on the Secure Boot process can be found in this PDF.



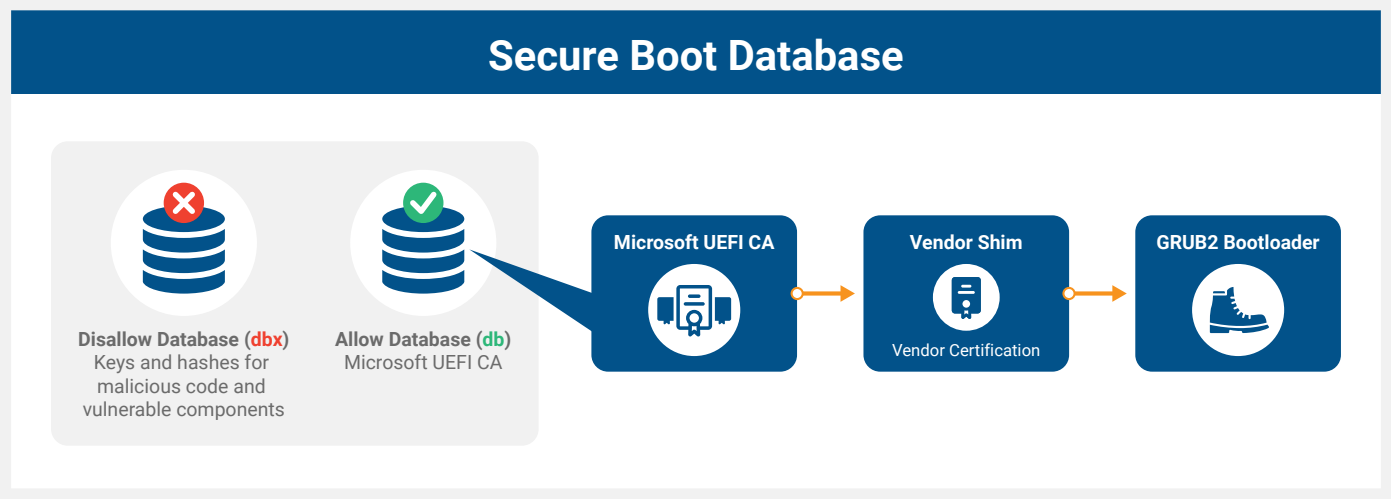
Source: Eclypsium

CHAINS OF TRUST AND GRUB2

Next, OEMs must manage a list of who is permitted to sign code trusted by the Secure Boot Database. Instead of having every OEM manage certificates from every possible firmware, driver, or OS provider, Secure Boot allows for the use of a centralized Certificate Authority (CA). Microsoft's 3rd Party UEFI CA provides the industry standard signing service for Secure Boot. In short, third parties can submit their code to Microsoft, and Microsoft will validate and sign the code with the Microsoft CA. This establishes a chain of trust that only requires OEMs to enroll the Microsoft 3rd Party UEFI CA to their platforms to enable them to boot third-party installation media and operating systems by default when Secure Boot is enabled.

This includes the ability to sign bootloaders from non-Microsoft operating systems such as Linux. In almost every modern Linux distribution, GRUB (the Grand Unified Bootloader) is the bootloader that loads and transfers control to the operating system. In this document, all references to GRUB are intended to refer to GRUB2, which was a complete rewrite from the previous version commonly referred to as "GRUB Legacy." Starting in 2009, all widely used Linux distributions have transitioned to using GRUB2. GRUB Legacy has been deprecated and is generally only found in older releases.

Due to legal issues arising from license incompatibilities, open-source projects and other third parties build a small application called a "shim," which contains the vendor's certificate and code that verifies and runs the bootloader (GRUB2). The vendor's shim is verified using the Microsoft 3rd Party UEFI CA and then the shim loads and verifies the GRUB2 bootloader using the vendor certificate embedded inside itself.



Additional detail on the role of the Microsoft UEFI CA in the boot process is available here.

Source: Eclypsium

CHALLENGES OF SECURE BOOT

As with any technical process, Secure Boot is not without its potential problems. The process involves many pieces of code, and a vulnerability in any one of them presents a single point of failure that could allow an attacker to bypass Secure Boot. Additionally, although UEFI Secure Boot attempts to provide certain integrity guarantees to the boot process, other misconfigurations of the hardware or missing protection features can undermine boot security. One such example is a DMA attack using tools such as PCIe Microblaze. Additionally, as we will show in this blog post, a vulnerability in the boot process that enables arbitrary code execution can allow attackers to control the boot process and operating system, even when secure boot signatures are verified.

Attackers can also use a vulnerable bootloader against the system. For example, if a valid bootloader was found to have a vulnerability, a piece of malware could replace the device's existing bootloader with the vulnerable version. The bootloader would be allowed by Secure Boot and give the malware complete control over the system and OS. Mitigating this requires very active management of the dbx database used to identify malicious or vulnerable code.

Source: Eclypsium

Additionally, updates and fixes to the Secure Boot process can be particularly complex and run the risk of inadvertently breaking machines. The boot process naturally involves a variety of players and components including device OEMs, operating system vendors, and administrators. Given the fundamental nature of the boot process, any sort of problems run a high risk of rendering a device unusable. As a result, updates to Secure Boot are typically slow and require extensive industry testing.

Breaking Secure Boot Through GRUB2

In the course of Eclypsium's analysis, we have identified a buffer overflow vulnerability in the way that GRUB2 parses content from the GRUB2 config file (grub.cfg). Of note: The GRUB2 config file is a text file and typically is not signed like other files and executables. This vulnerability enables arbitrary code execution within GRUB2 and thus control over the booting of the operating system. As a result, an attacker could modify the contents of the GRUB2 configuration file to ensure that attack code is run before the operating system is loaded. In this way, attackers gain persistence on the device.

Such an attack would require an attacker to have elevated privileges. However, it would provide the attacker with a powerful additional escalation of privilege and persistence on the device, even with Secure Boot enabled and properly performing signature verification on all loaded executables. One of the explicit design goals of Secure Boot is to prevent unauthorized code, even running with administrator privileges, from gaining additional privileges and pre-OS persistence by disabling Secure Boot or otherwise modifying the boot chain.

With the sole exception of one bootable tool vendor who added custom code to perform a signature verification of the grub.cfg config file in addition to the signature verification performed on the GRUB2 executable, all versions of GRUB2 that load commands from an external grub.cfg configuration file are vulnerable. As such, this will require the release of new installers and bootloaders for all versions of Linux. Vendors will need to release new versions of their bootloader shims to be signed by the Microsoft 3rd Party UEFI CA. It is important to note that until all affected versions are added to the dbx revocation list, an attacker would be able to use a vulnerable version of shim and GRUB2 to attack the system. This means that every device that trusts the Microsoft 3rd Party UEFI CA will be vulnerable for that period of time.

In addition to vendors using shims signed by the Microsoft 3rd Party UEFI CA, some OEMs that control both the hardware and the software stack in their devices use their own key that is provisioned into the hardware at the factory to sign GRUB2 directly. They will need to provide updates and revocation of previous vulnerable versions of GRUB2 for these systems as well.

This vulnerability was assigned CVE-2020-10713 "GRUB2: crafted grub.cfg file can lead to arbitrary code execution during boot process" with a CVSS rating of 8.2 (High) / CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H.

Click here to go straight to the Impact and Mitigations sections.

VULNERABILITY ANALYSIS

The vulnerability is a buffer overflow that occurs in GRUB2 when parsing the grub.cfg file. This configuration file is an external file commonly located in the EFI System Partition and can therefore be modified by an attacker with administrator privileges without altering the integrity of the signed vendor shim and GRUB2 bootloader executables. The buffer overflow allows the attacker to gain arbitrary code execution within the UEFI execution environment, which could be used to run malware, alter the boot process, directly patch the OS kernel, or execute any number of other malicious actions.

To dig a little deeper into the vulnerability itself, we'll take a closer look at how the code works internally. In order to process commands from the external configuration file, GRUB2 uses flex and bison to generate a parsing engine for a domain-specific language (DSL) from language description files and helper functions.

This is generally considered to be a better approach than manually writing a custom parser for each DSL. However, GRUB2, flex, and bison are all complex software packages with their own design assumptions that can be easy to overlook. And those mismatched design assumptions can result in vulnerable code.

The parser engine generated by flex includes this define as part of the token processing code:

```
#define YY_DO_BEFORE_ACTION \
     yyg->yytext_ptr = yy_bp; \
     yyleng = (int) (yy_cp - yy_bp); \
     yyg->yy_hold_char = *yy_cp; \
     *yy_cp = '\0'; \
     if ( yyleng >= YYLMAX ) \
        YY_FATAL_ERROR( "token too large, exceeds
        YYLMAX" ); \
     yy_flex_strncpy( yytext, yyg->yytext_
        ptr, yyleng + 1 , yyscanner); \
     yyg->yy_c_buf_p = yy_cp;
```

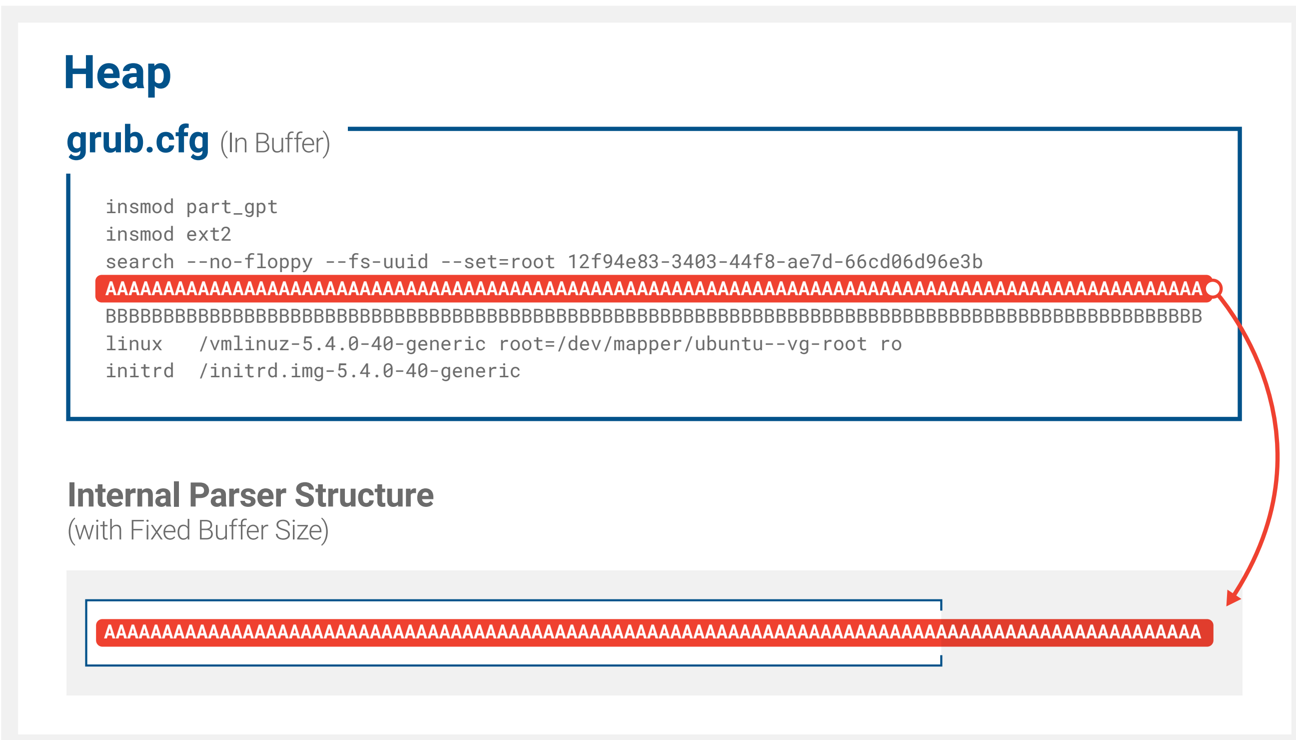
In this macro, the generated code detects that it has encountered a token that is too large to fit into flex's internal parse buffer and calls **YY_FATAL_ERROR()**, which is a helper function provided by the software that is using the flex-generated parser.

However, the **YY_FATAL_ERROR()** implementation provided in the GRUB2 software package is:

```
#define YY_FATAL_ERROR(msg)                    \
     do {                                      \
        grub_printf (_("fatal error: %s\n"),
          _(msg));                             \
    } while (0)
```

Rather than halting execution or exiting, it just prints an error to the console and returns to the calling function. Unfortunately, the flex code has been written with the expectation that any calls to `YY_FATAL_ERROR()` will never return. This results in `yy_flex_strncpy()` being called and copying the source string from the configuration file into a buffer that is too small to contain it.

*Contents of **grub.cfg** are read from disk into heap buffer and then parsed by vulnerable code resulting in overflow of internal parser structure.*

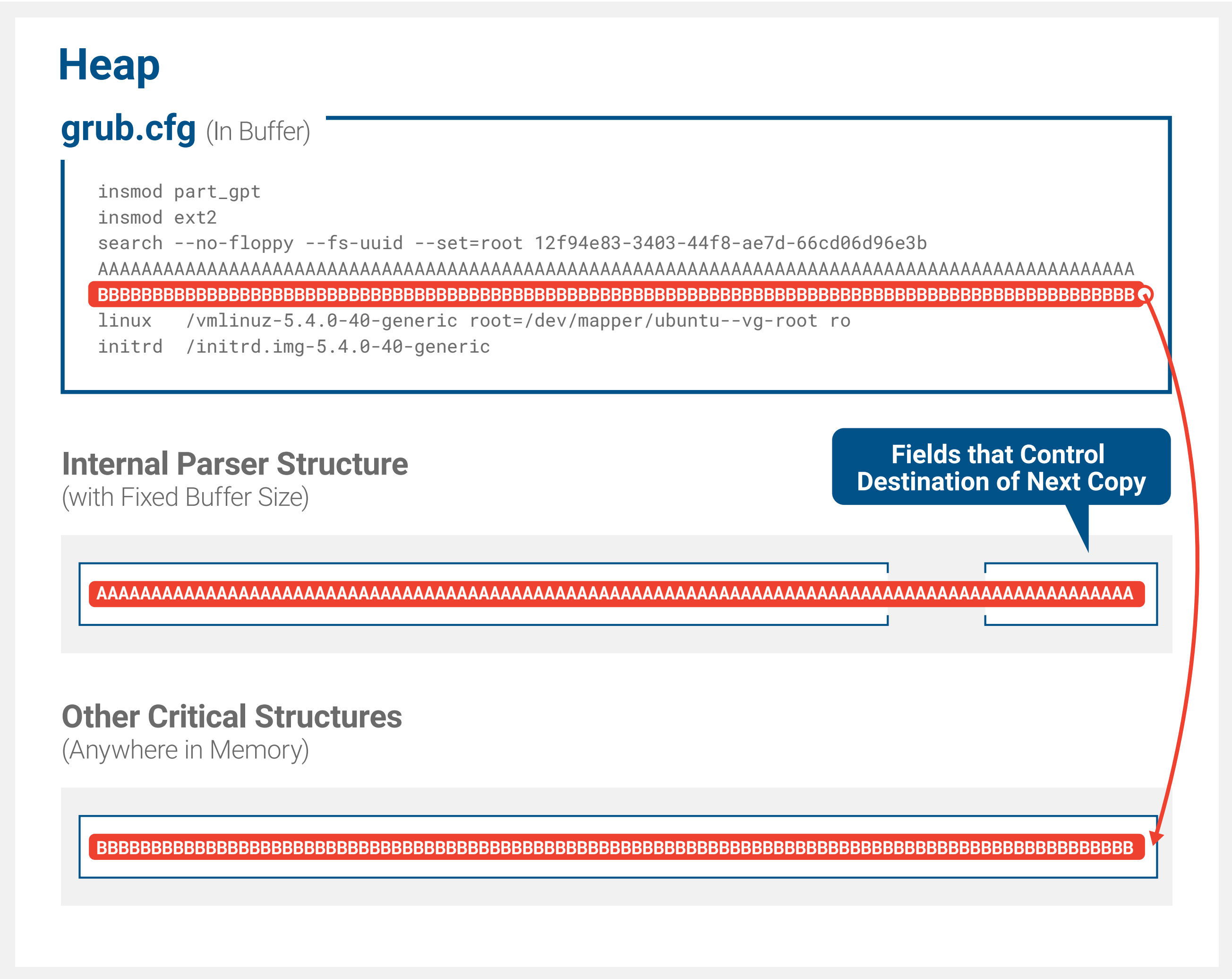


Source: Eclypsium

Beyond just this specific path, a number of additional places throughout the flex-generated code also expect that any calls to `YY_FATAL_ERROR()` never return and perform unsafe operations when that expectation is broken. Mismatched assumptions between producers and consumers of an API are a very common source of vulnerabilities.

Ultimately, by providing a configuration file with input tokens that are too long to be handled by the parser, this buffer overflow overwrites critical structures in the heap. These overwritten fields include internal parser structure elements, which can be used as an arbitrary write-what-where primitive to gain arbitrary code execution and hijack the boot process.

Fields overwritten in internal parse structure can be used to write arbitrary data anywhere in memory.



Source: Eclypsium

Of further note, the UEFI execution environment does not have Address Space Layout Randomization (ASLR) or Data Execution Prevention (DEP/NX) or other exploit mitigation technologies typically found in modern operating systems, so creating exploits for this kind of vulnerability is significantly easier. The heap is fully executable without the need to build ROP chains.

Finally, rather than being architecture-specific, this vulnerability is in a common code path and was also confirmed using a signed ARM64 version of GRUB2.

ADDITIONAL VULNERABILITIES

There have been a couple of examples of previous vulnerabilities found in GRUB2 that result in arbitrary code execution, but with a much smaller scope.

In April 2019, a vulnerability in how GRUB2 was used by the Kaspersky Rescue Disk was publicly disclosed. In February 2020, more than six months after a fixed version had been released, Microsoft pushed an update to revoke the vulnerable bootloader across all Windows systems by updating the UEFI revocation list (dbx) to block the known-vulnerable Kaspersky bootloader. Unfortunately, this resulted in systems from multiple vendors encountering unexpected errors, including bricked devices, and the update was removed from the update servers.

Additionally, in May 2020, Dmytro Oleksiuk disclosed that certain HPE ProLiant servers contained a version of GRUB2 signed by a HP CA that allows the use of the "insmod" command to load unsigned code. This issue was assigned CVE-2020-7205 and is also embargoed until July 29th.

In response to our initial vulnerability report, additional scrutiny was applied to the GRUB2 code and a number of additional vulnerabilities were discovered by the Canonical security team:

- **CVE-2020-14308 GRUB2:** grub_malloc does not validate allocation size allowing for arithmetic overflow and subsequent heap-based buffer overflow
  —6.4 (Medium) / CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H
- **CVE-2020-14309 GRUB2:** Integer overflow in grub_squash_read_symlink may lead to heap based overflow
  —5.7 (Medium) / CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:N/I:H/A:H
- **CVE-2020-14310 GRUB2:** Integer overflow read_section_from_string may lead to heap based overflow
  —5.7 (Medium) / CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:N/I:H/A:H
- **CVE-2020-14311 GRUB2:** Integer overflow in grub_ext2_read_link leads to heap based buffer overflow,
  —5.7 (Medium) / CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:N/I:H/A:H
- **CVE-2020-15705 GRUB2:** avoid loading unsigned kernels when grub is booted directly under secureboot without shim
  —6.4 (Medium) /CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H
- **CVE-2020-15706 GRUB2 script:** Avoid a use-after-free when redefining a function during execution
  —6.4 (Medium) /CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H
- **CVE-2020-15707 GRUB2:** Integer overflow in initrd size handling.
  —5.7 (Medium) /CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:N/I:H/A:H

Given the difficulty of this kind of ecosystem-wide update/revocation, there is a strong desire to avoid having to do it again six months later. To that end, a large effort — spanning multiple security teams at Oracle, Red Hat, Canonical, VMware and Debian — using static analysis tools and manual review helped identify and fix dozens of further vulnerabilities and dangerous operations throughout the codebase that do not yet have individual CVEs assigned.

IMPACT

Due to a weakness in the way GRUB2 parses its configuration file, an attacker can execute arbitrary code that bypasses signature verification. The BootHole vulnerability discovered by Eclypsium can be used to install persistent and stealthy bootkits or malicious bootloaders that operate even when Secure Boot is enabled and functioning correctly. This can ensure attacker code runs before the operating system and can allow the attacker to control how the operating system is loaded, directly patch the operating system, or even direct the bootloader to alternate OS images. It gives the attacker virtually unlimited control over the victim device. Malicious bootloaders have recently been observed in the wild, and this vulnerability would make devices susceptible to these types of threats.

All signed versions of GRUB2 that read commands from an external grub.cfg file are vulnerable, affecting every Linux distribution. To date, more than 80 shims are known to be affected. In addition to Linux systems, any system that uses Secure Boot with the standard Microsoft UEFI CA is vulnerable to this issue. As a result, we believe that the majority of modern systems in use today, including servers and workstations, laptops and desktops, and a large number of Linux-based OT and IoT systems, are potentially affected by these vulnerabilities.

Additionally, any hardware root of trust mechanisms that rely on UEFI Secure Boot could be bypassed as well.

Mitigation

Full mitigation of this issue will require coordinated efforts from a variety of entities: affected open-source projects, Microsoft, and the owners of affected systems, among others. This will include:

1. Updates to GRUB2 to address the vulnerability.
2. Linux distributions and other vendors using GRUB2 will need to update their installers, bootloaders, and shims.
3. New shims will need to be signed by the Microsoft 3rd Party UEFI CA.
4. Administrators of affected devices will need to update installed versions of operating systems in the field as well as installer images, including disaster recovery media.
5. Eventually the UEFI revocation list (dbx) needs to be updated in the firmware of each affected system to prevent running this vulnerable code during boot.

On the Coordinated Release Date (CRD) of July 29, we expect to see advisories and/or updates from the following affected parties:

- Microsoft
- UEFI Security Response Team (USRT)
- Oracle
- Red Hat (Fedora and RHEL)
- Canonical (Ubuntu)
- SuSE (SLES and openSUSE)
- Debian
- Citrix
- VMware
- Various OEMs
- Software vendors, including security software, are also impacted by this vulnerability and will be updating their bootloaders.
- … more to be added once we have a full list …

However, full deployment of this revocation process will likely be very slow. UEFI-related updates have had a history of making devices unusable, and vendors will need to be very cautious. If the revocation list (dbx) is updated before a given Linux bootloader and shim are updated, then the operating system will not load. As a result, updates to the revocation list will take place over time to prevent breaking systems that have yet to be updated. There are also edge cases where updating the dbx can be difficult, such as with dual-boot or deprovisioned machines. When any OS is installed or launched, the bootloader and OS need to be updated before the revocation is applied to the system.

Further complicating matters, enterprise disaster recovery processes can run into issues where approved recovery media no longer boots on a system if dbx updates have been applied. In addition when a device swap is needed due to failing hardware, new systems of the same model may have already had dbx updates applied and will fail when attempting to boot previously-installed operating systems. Before dbx updates are pushed out to enterprise fleet systems, recovery and installation media must be updated and verified as well.

Microsoft will be releasing a set of signed dbx updates, which can be applied to systems to block shims that can be used to load the vulnerable versions of GRUB2. Due to the risk of bricking systems or otherwise breaking operational or recovery workflows, these dbx updates will initially be made available for interested parties to manually apply to their systems rather than pushing the revocation entries and applying them automatically. This will allow IT professionals, enthusiasts, and others the opportunity to test the revocation updates on their individual systems and identify any issues before making the revocations mandatory.

Organizations should additionally ensure they have appropriate capabilities for monitoring UEFI bootloaders and firmware and verifying UEFI configurations, including revocation lists, in their systems. Organizations should also test recovery capabilities as updates become available, including the "reset to factory defaults" functionality in the UEFI setup. This will ensure that they can recover devices if a device is negatively impacted by an update. Finally, organizations should be monitoring their systems for threats and ransomware that use vulnerable bootloaders to infect or damage systems.

RECOMMENDATIONS

1. Right away, start monitoring the contents of the bootloader partition (EFI system partition). This will buy time for the rest of the process and help identify affected systems in your environment. For those who have deployed the Eclypsium solution, you can see this monitoring under the "MBR/Bootloader" component of a device.

2. Continue to install OS updates as usual across desktops, laptops, servers, and appliances. Attackers can leverage privilege escalation flaws in the OS and applications to take advantage of this vulnerability so preventing them from gaining administrative level access to your systems is critical. Systems are still vulnerable after this, but it is a necessary first step. Once the revocation update is installed later, the old bootloader should stop working. This includes rescue disks, installers, enterprise gold images, virtual machines, or other bootable media.

3. Test the revocation list update. Be sure to specifically test the same firmware versions and models that are used in the field. It may help to update to the latest firmware first in order to reduce the number of test cases.

4. To close this vulnerability, you need to deploy the revocation update. Make sure that all bootable media has received OS updates first, roll it out slowly to only a small number of devices at a time, and incorporate lessons learned from testing as part of this process.

5. Engage with your third-party vendors to validate they are aware of, and are addressing, this issue. They should provide you a response as to its applicability to the services/solutions they provide you as well as their plans for remediation of this high rated vulnerability.

Eclypsium has powershell and bash scripts available which can be used to detect bootloaders that are being revoked by this dbxupdate.

Conclusions

While Secure Boot is easily taken for granted by most users, it is the foundation of security within most devices. Once compromised, attackers can gain virtually complete control over the device, its operating system, and its applications and data. And as this research shows, when problems are found in the boot process, they can have far-reaching effects across many types of devices.

We will update this blog post as more information becomes available, and we encourage users and administrators to closely monitor alerts and notifications from their hardware vendors, the Microsoft MSRC, and any relevant open-source projects.

REFERENCES

Microsoft
Security advisory: https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV200011

UEFI Forum
Updated Revocation List: https://uefi.org/revocationlistfile

Debian
Security advisory: https://www.debian.org/security/2020-GRUB-UEFI-SecureBoot

Canonical:
Security advisory: https://ubuntu.com/security/notices/USN-4432-1
KnowledgeBase article: https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/GRUB2SecureBootBypass

Red Hat
Customer documentation: https://access.redhat.com/security/vulnerabilities/grub2bootloader
CVE information: https://access.redhat.com/security/cve/cve-2020-10713
Vulnerability response article: https://access.redhat.com/security/vulnerabilities/grub2bootloader

SUSE
Security advisory: https://www.suse.com/c/suse-addresses-grub2-secure-boot-issue/
Knowledge Base article: https://www.suse.com/support/kb/doc/?id=000019673

HP
Security advisory: HPSBHF03678 rev. 1 - GRUB2 Bootloader Arbitrary Code Execution

HPE
Security advisory: https://techhub.hpe.com/eginfolib/securityalerts/Boot_Hole/boot_hole.html

VMware
Knowledge Base article: https://kb.vmware.com/s/article/80181

Upstream Grub2 project
GRUB2 Git Repository
GRUB Developer Mailing List