UEFI Secure Boot Impact on Linux

October 28, 2011.

Jeremy Kerr - Technical Architect - Canonical
Matthew Garrett - Senior Software Engineer - Red Hat
James Bottomley – Kernel Developer

1. Summary

Secure boot technology is now part of the new UEFI firmware specification. Given that Microsoft®'s Windows® 8 will require secure boot to be enabled by default, it is expected that the majority of personal computer devices will ship with it enabled in the first quarter of 2012.

The UEFI specification for secure boot does not define who controls the boot restrictions on UEFI platforms, leaving the platform implementer in control of the exact security model. Unfortunately, Microsoft's recommended implementation of secure boot removes control of the system from the hardware owner, and may prevent open source operating systems from functioning. The Windows 8 requirement for secure boot will pressure OEMs to implement secure boot in this fashion.

We believe that restrictions that prevent users from exercising full control over their hardware is not in the best interest of those users, and works against the spirit of open source software in general.

Therefore, we present a set of recommendations that will allow users the freedom to choose their software, while retaining the security features of UEFI Secure Boot, and complying with open source licenses used in distributions of Linux®.

2. Technology Description

UEFI secure boot is a technology integrated into the latest version (v2.3.1) of the UEFI specification, implementing part of a "verified software stack". A verified software stack authenticates that the critical parts of software running on a particular device are authorised and have not been tampered with in any way. This is achieved by implementing public key cryptography to verify that various signatures are valid when loading each successive layer in the operating system boot process.

Secure boot functionality is concerned with several stages of this process: validation of system firmware, drivers, and of software loaded by the built-in firmware. Typically, this software is the "bootloader", which is responsible for loading the operating system image. With secure boot enabled, public keys in the UEFI firmware are used to validate the bootloader read from disk. This ensures that the bootloader image has been authorised to run on this platform.

In order to provide protection for the full software stack, the bootloader would typically be responsible for validating the signature of the operating system's kernel, which in turn validates the signatures of other software.

If any of these components are replaced or altered and signature verification fails, the system will be prevented from booting. For example, if the bootloader signature does not match what the UEFI firmware is expecting, the system will not boot.

2.1. Implementation and platform control

The UEFI specification states that secure boot functionality is optional, but it is expected that OEMs will be influenced by proprietary operating system vendors to implement it. Specifically, the Windows 8 logo program will require secure boot to be enabled[1]. In theory, the wide deployment of secure boot will be a good step towards increasing the security of general-purpose computing platforms.

Because the UEFI specification does not define the exact security model to be implemented, there is some flexibility in how hardware vendors (typically, OEMs) will configure secure boot on their systems. In particular, the UEFI secure boot specification does not define who will be the custodians of the public keys that will be embedded in the UEFI firmware.

However, the Windows 8 requirements attempt to define some of these implementation details, in a way that restricts users' freedoms. In particular, that the "root" of the authentication system (called the Platform Key, or PK in the UEFI specification) will be controlled by the OEM[2]. Essentially, the owner of the PK has ultimate control of the security restrictions on secure-boot enabled hardware. If the user does not control the PK, then the user may not be able to fully configure secure boot to their requirements.

2.2. Market pressure

With system security being a growing concern, an authenticated boot path may become a requirement of certain compliance standards, such as PCI, Sarbanes-Oxley, FIPS, or HIPPA. Enterprise customers could make secure boot capability a requirement of their purchasing process, following recommendations from their software vendors.

2.3. Key revocation

Part of the UEFI specification, and a mandatory part of any public key cryptography system, is a key revocation process. This allows known compromised keys to be added to a blacklist of keys that can no longer be used to verify boot images. If a key is added to this blacklist, images that rely on that key will no longer work.

This is known in the UEFI specification as the "forbidden signatures database", and is modifiable during execution by authenticated software.

2.4. Full-stack support

For the secure boot concept to be effective, it should be a part of a "full-stack" security system, where each layer of software—from the system firmware to (at least) all components of the operating system kernel—is verified before execution.

Depending on the technical implementation, a signed bootloader may also have to verify signatures on Linux kernel images that it loads. Failure to do so would permit the signed bootloader to execute

[1]From http://video.ch9.ms/build/2011/slides/HW-457T_van_der_Hoeven.pptx, slides 11 and 17.

[2]"The platform is secured through a platform key that the OEM installs in firmware during manufacturing" - http://blogs.msdn.com/b/b8/archive/2011/09/22/protecting-the-pre-os-environment-with-uefi.aspx

arbitrary unsigned code, possibly circumventing secure boot technology. If such a non-verifying bootloader were signed and distributed, malware could simply replace the verifying bootloader with the non-verifying one, and use it to load a compromised operating system. In effect, this non-verifying bootloader becomes the tool to compromise secure boot, and the key used to sign it would likely be added to systems' forbidden signatures database, either before shipping or at runtime by system software.

The way to make the Linux bootloader a part of the secure boot process is to alter it to validate the signature of the software that it loads (typically the Linux kernel), and refuse to execute software that does not pass this validation.

3. Benefits

Secure boot technology presents some beneficial features to both free software and proprietary operating systems. When used in appropriate situations, it could be useful for Linux users.

3.1. Malware prevention

One of the main benefits of secure boot technology is that it can greatly reduce the number of malware "rootkits". In the case that malware modifies critical operating system components, on system restart the boot process is interrupted with signature verification errors and the end user is immediately aware that the system has been compromised. System recovery software could then simply replace any unsigned components.

3.2. Local security policy

Under certain circumstances local security policy may dictate that only authorised operating systems be booted. Secure boot allows IT departments to limit the set of authorised keys in order to aid enforcement of such a policy.

3.3. Perceived benefits for proprietary software

Although there are some end-user benefits of secure boot, there are some consequences that may benefit proprietary software vendors, rather than the user. While we cannot specifically determine these benefits, we perceive that the following items may be attractive to proprietary software vendors.

3.3.1. Hardware/software integration

Controlling the boot environment may make it possible for software to be reliably tied to a specific piece of hardware. This creates the opportunity for a "forced obsolescence" scenario, where hardware upgrades are necessary to install future versions of system software, or vice versa.

3.3.2. Recurring revenue assurance

If secure boot is used to verify all parts of the operating system, including application binaries, the key owner is in a position to enforce that applications can only be installed from an approved "App

Store", thus ensuring recurring revenue from all end user purchases. In this situation, it would be technically infeasible to install applications procured via an alternate source.

4. Disadvantages

Although secure boot is marketed as being beneficial to end users, it has many disadvantages that are not being communicated clearly, if at all. In particular, the currently-recommended implementation of secure boot, where the machine owner does not control the PK, may present the following issues.

4.1. Choice of hardware

Secure boot requires all components that run in the context of the system firmware to be signed. This includes not only the bootloader, but also any firmware-level hardware drivers. For all-in-one systems such as laptops this could be achieved by all drivers being signed with the OEM's key. However, for systems where components may be replaced after purchase, the problem is less tractable. If the component vendors signs their own drivers, then they must ensure that their key is installed on all hardware they wish to support. This approach would prevent new hardware vendors from entering the market until they had distributed their key to a range of OEMs, and has a large per-platform overhead. Even then, if no mechanism is provided to install their key into older systems, only new hardware will support their components.

An alternative is for vendors to have their drivers signed with a key that is included in the majority (if not all) of shipped platforms. This approach avoids the per-platform overhead issues, but means that the hardware vendor is now beholden to the owner of said key to provide a signature. This would be likely to add additional cost and delays to the process of releasing new hardware or drivers.

The most scalable approach would be for vendors to obtain a key with a chain of trust back to a key present in all shipped platforms. This adds relatively little overhead, but requires that the vendor's key be present in order for secure boot validation to operate. If the key used for the root of trust is also used to sign operating systems, this makes it impractical for end users or organisations to remove that key if they do not wish their hardware to run those operating systems.

4.2. Choice of operating system

A system shipped with secure boot enabled by default will only boot signed operating systems. Since there is no centralised UEFI signing authority, it is therefore necessary for each OS vendor to either ensure that the public half of their signing key is included in the system firmware, or for the OEM to provide a mechanism for the end-user to install new signing keys.

Microsoft have assured the community that OEMs will be able to decide the level of restrictions on boot images, within the constraints of the Windows 8 requirements. While OEMs may decide to allow secure boot to be disabled, this still presents a significant barrier to installing Linux on machines that have shipped with Windows 8, as it is only possible by manually switching an option in the system firmware. Requiring users to reconfigure their system firmware in order to use an alternative to Windows will undoubtedly reduce the feasibility of these alternatives.

Providing a mechanism for end-user installed keys is problematic. In order to prevent pre-operating-system malware, it must be impossible for the malware to install new signing keys. Any solution for end-user installation of keys must guard against programmatic installation by malware.

If no mechanism is provided for the end-user to install keys, every operating system vendor faces the same predicament as described in section 4.1. Providing one key per operating system results in scalability issues, with each platform requiring the key to be preinstalled. Signing all operating systems with a single third-party key adds significant cost and delays to entering the market.

4.3. Usability of alternate operating systems

If secure boot must be disabled before an alternate operating system can be booted, then those alternatives will become restricted to technically-minded users who are able to reconfigure their firmware to disable secure boot.

Likewise, dual-boot configurations will become difficult to manage, as users must enable and disable secure boot when switching between signed and unsigned boot images.

4.4. Operating system updates

Each component of the operating system involved in the firmware boot process must be signed. This makes it impossible for these components to be dynamically generated on the end-user hardware. Instead, static images must be generated and signed before being provided to the end-user. This is incompatible with the current GRUB 2 design, which generates a system-specific boot loader image at install time.

4.5. Consequences of key compromise

In the event of the private half of a trusted signing key being disclosed, it may be used to sign malware which will then successfully validate, despite secure boot restrictions. Avoiding this requires that the signing key be revoked and blacklisted, which will also cause any other code signed with the key to cease validation. If the key belongs to an operating system vendor then this will prevent the operating system from booting, while if it belongs to a hardware vendor then that vendor's firmware-level drivers will no longer validate and the component will no longer function in the firmware environment. In the worst case scenario, key revocation may render some machines unbootable.

4.6. Infrastructure requirements

Implementing a secure boot system requires a non-trivial amount of supporting infrastructure and maintenance effort. This includes overheads for maintaining a secret key, authorising the key, and signing boot images.

Recent compromises to well-known browser certificate authorities[1] show that there are significant consequences if this effort is not conducted in a secure (and therefore potentially costly) manner.

For smaller Linux distributions, supporting secure boot may not be feasible due to these overheads.

4.7. Security vulnerability disclosure

Usually, a proportion of security-conscious users will work alongside software vendors to assist in the reporting and responsible disclosure of security vulnerabilities of a software product.

However, the enforcement of externally-controlled secure boot restrictions may decrease the likelihood of these vulnerabilities being reported to software vendors. If the only method of running unauthorised or customised software on a hardware device is to exploit a vulnerability, then these vulnerabilities become valuable to allow users to run modified software on their systems.

Instead of reporting security issues, researchers working on restricted systems may choose not to disclose flaws, but instead use them to circumvent secure boot restrictions. In this case, the software vendor may only be indirectly notified by these vulnerabilities, one at a time. Examples include attacks on Apple®'s iOS hardware, and Sony®'s Playstation™ 3.

5. Implementation factors

As secure boot has not yet been implemented "in the wild", there are a number of factors that will change the impact of secure boot on Linux.

5.1. Factory default configuration

The default secure boot configuration of systems will determine whether or not they can be used "out of the box" for booting Linux, or other non-signed operating system images.

We do not currently have an estimate of the proportion of systems that will ship with secure boot restrictions enforced by default.

5.2. Ability to disable secure boot

The ability to disable secure boot restrictions is vital to ensure Linux's continued ability to run on a wide range of platforms. If users cannot disable secure boot, then fewer platforms will be able to run open source operating systems. Additionally, open source software developers will no longer be able to build and test modified (and therefore not signed) software on these restricted platforms.

If the process required to disable secure boot is difficult for non-technical users, then we risk restricting use of unsigned software to a small portion of the market. Currently it is up to each OEM as to whether or not to create a user interface to enable or disable secure boot.

[1] DigiNotar, a well-known CA, had its keys compromised and used to sign forged SSL certificates: http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/

5.3. Ability to reconfigure keys

Some Linux users may wish to use secure boot functionality with custom images. In this case, they will never be using the OEM-defined keys. The UEFI specification defines a “setup mode” where keys can be configured.

For this use case, users will need to be able to reconfigure the platform’s firmware to include their own keys, which will require an interface to do so. However, since this will be a relatively infrequently-used feature, it does not seem likely that BIOS vendors will implement it for general-purpose usage.

5.4. Key revocation policy

The UEFI specification does not define criteria for adding a key to the forbidden signatures database, so it is left to the platform manufacturer to decide which keys are initially included, and the operating system vendor to decide which keys may be added after purchase.

Generally, it makes sense when managing the forbidden signature database to add known-compromised keys to it, but different organisations may have different standards for a “compromised key”.

It is also possible, although unlikely, that some organisations may add competitors’ keys to the systems’ forbidden signatures database, even though the keys have not been compromised. This would prevent competitors’ software from running on some systems.

5.5. Automated deployment

Performing automated installations of operating systems to a large number of machines may be difficult in the event of customers wishing to use their own signing key. The difficulty of automatically installing new signing keys without providing the opportunity for malware to use the same mechanism leaves this an open question. It is undesirable for IT departments to have to perform manual key installation.

6. Recommendations

Secure boot technology can be beneficial for increasing the security of Linux installations. Linux distributions should gain secure boot compatibility in order to increase protection against malware and disk encryption circumvention, provided that users’ freedoms are protected.

Unfortunately, the current implementation recommended for secure boot makes installation of Linux more difficult and may prevent users from modifying their own systems.

So, we recommend that secure boot implementations are designed around the *hardware owner* having full control of the security restrictions.

We recommend that all OEMs allow secure boot to be easily disabled and enabled through a firmware configuration interface

It is essential that users are able to remove secure boot restrictions, and boot the software of their choice on the devices that they own. Furthermore, the interface to configure this option should be easily accessible by non-technical users.

Of course, this option should only be available to users with physical access to the hardware, and not be accessible via programmatic means.

We recommend that OEMs (with assistance from BIOS vendors) provide a standardised mechanism for configuring keys in system firmware
For secure boot to be useful in a user-controlled environment, it must be possible for users to add custom keys (KEK, db and dbx entries) to the system firmware. Keys may then be shipped with an operating system or generated by the user. This allows the user to maintain control of the code run on their systems without giving up the benefits of secure boot.

For support purposes, the mechanism provided for key management must be consistent across platforms, and provide a simple method of booting custom software, including from removable media. A suggested implementation may be to scan removable media for signing keys and prompt the user for their installation, or using the specification-defined setup mode to allow key reconfiguration.

We recommend that hardware ship in setup mode, with the operating system taking responsibility for initial key installation
Shipping hardware in setup mode allows key policy to be determined by the operating system vendor or end user. Pre-installed operating systems could then install their own signing keys on first boot. This permits the user to avoid the situation where pre-installed signing keys do not match the user's desired security policy.

For further information on secure boot and Linux, contact the authors:

- Jeremy Kerr – jeremy.kerr@canonical.com
- Matthew Garrett – mjg@redhat.com
- James Bottomley – james.bottomley@hansenpartnership.com