Anti Evil Maid | The Invisible Things Blog

16-20 minutes

Anti Evil Maid is an implementation of a TPM-based static trusted boot with a primary goal to prevent Evil Maid attacks.

The adjective *trusted*, in *trusted boot*, means that the goal of the mechanism is to somehow attest to a user that only desired (trusted) components have been loaded and executed during the system boot. It's a common mistake to confuse it with what is sometimes called s*ecure boot*, whose purpure is to *prevent* any unauthorized component from executing. Secure boot is problematic to implement in practice, because there must be a way to tell which components are authorized for execution. This might be done using digital signatures and some kind of CA infrastructure, but this gets us into problems such as who should run the CA, what should be the policy for issuing certificates, etc.

The adjective s*tatic* means that the whole *chain of trust* is anchored in a special code that executes before all other code on the platform, and which is kept in a non re-flashable memory, whose sole purpure is to make the initial measurement of the next component that is going to be executed, which is the BIOS code. This special code, also known as Core Root of Trust for Measurement (CRTM), might be part of the BIOS (but kept on a special read-only memory, or implemented by some other entity that executes before the BIOS reset vector, such as e.g. Intel ME or the processor microcode even. Once measured, the BIOS code is executed, and it is now its turn to measures the platform configuration, Option ROM code, and MBR. Then the loader (stored in the MBR), such as Trusted GRUB, takes over and measures its own next stages (other than the MBR sector), and the hypervisor, kernel, and initramfs images that are to be loaded, together with their configuration (e.g. kernel arguments).

As explained above, trusted boot can only retrospectively tell the user whether correct (trusted) software has booted or not, but cannot *prevent* any software from executing. But how can it communicate anything reliably to the user, if it might have just been compromised? This is possible thanks to the TPM *unseal* operation that releases secrets to software only if correct software has booted (as indicated by correct hashes in select PCR registers).

So the idea is that if a user can see correct secret message (or perhaps a photo) being displayed on the screen, then it means that correct software must have booted, or otherwise the TPM would not release (*unseal*) the secret. Of course we assume the adversary had no other way to sniff this secret and couldn't simply hardcode it into the Evil Maid – more on this later.

Another way to look at it is to realize that Anti Evil Maid is all about **authenticating machine to the user**, as opposed to the usual case of authenticating the user to the machine/OS (login and password, decryption key, token, etc). We proceed with booting the machine and entering sensitive information, only after we get confidence it is still our *trusted* machine and not some compromised one.

Installing Anti Evil Maid

Anti Evil Maid should work for any Linux system that uses dracut/initramfs, which includes Qubes, Fedora and probably many other distros. You can find the Anti Evil Maid source code in a git repository here. You can also download a tarball with sources and prebuilt rpm packages from here (they all should be signed with the Qubes signing key). Qubes Beta 2, that is coming soon, will have those RPMs already per-installed.

To install Anti Evil Maid, follow the instructions in the README file.

Some Practical considerations

If you decided to use **no password for your TPM SRK key** (so, you passed '-z' to tpm_takeownership, see the README), then you should definitely install Anti Evil Maid on a removable USB stick. Otherwise, if you installed it on your disk boot partition, the attacker would be able to just boot your computer and note down the secret passphrase that will be displayed on the screen. Then the attacker can compromise your BIOS/MBR/kernel images however she likes, and just hardcode the secret passphrase to make it look like if your system was fine.

If you decided to use custom TPM SRK password (so, you did *not* pass -z to tpm_takeownership), then you can install Anti Evil Maid onto your regular boot partition. The attacker would not be able to see your secret passphrase without knowing the SRK password. Now, the attacker can try another Evil Maid attack to steal this password, but this attack is easy to spot and prevent (see the discussion in the next section).

However, there is still a good argument to install Anti Evil Maid on a separate USB stick rather than on your built-in disk boot partition. This is because you can use Anti Evil Maid as a provider of a keyfile to your LUKS disk encryption (as an additional file unsealable by the TPM). This way you could also stop adversary that is able to sniff your keystrokes (e.g. using hidden camera, or electromagnetic leak), and capture your disk decryption passphrase (see the discussion in the next section).

In any case it probably would be a good idea to make a backup stick that you might want to use in case you lose or somehow damage your primary stick. In that case you should have a way to figure out if your system has been compromised in the meantime or not. Use another stick, with another passphrase, and keep it in a vault for this occasion.

Finally, be aware that, depending on which PCRs you decided to seal your secrets to, you might be unable to see the secret even after you changed some minor thing in your BIOS config, such as e.g. the order of boot devices. Every time you change something in your system that affects the boot process, you would need to reseal your secrets to new PCR values as described in the installation instructions.

Attacks prevented by Anti Evil Maid

The classic Evil Maid attack is fully prevented.

If the attacker is able to steal your Anti Evil Maid stick, and the attacker gets access to your computer, then the attacker would be able to learn your secret passphrase by just booting from the stolen stick. This is not fatal, because user should get alarmed seeing that the stick has been stolen, and use the backup stick to verify the system (with a different secret messages, of course), and later create a new stick for every day use with a new secret message.

A variation of the above attack is when the attacker silently *copies* the content of the stick, so that the user cannot realize that someone got access to the stick. Attacker then uses the copied stick to boot the user's computer and this way can learn the secret passphrase. Now, the attacker can infect the computer with Evil Maid, and can also bypass Anti Evil Maid verification by just hardcoding the secret message into Evil Maid. So, even though TPM would know that incorrect software has booted, and even though it would not unseal the secret, the user would have no way of knowing this (as the secret would still be displayed on screen).

In order to protect against this attack, one might want to use a non-default SRK password – see the installation instructions. Now an extra SRK password would be needed to unseal any secret from the TPM (in addition to PCRs being correct). So the attacker, who doesn't know the SRK password, is now not able to see the secret message and cannot prepare the Evil Maid Attack (doesn't know what secret passphrase to hardcode there).

The attacker might want to perform an additional Evil Maid attack targeted at capturing this SRK password, e.g. by infecting the user's stick. This, however, could be immediately detected by the user, because the user would see that after entering the correct SRK password, there was no correct secret passphrase displayed. The user should then assume the stick got compromised together with the SRK password, and should start the machine from the backup stick, verify that the backup secret is correct, and then create new AEM stick for daily usage.

If an attacker is able to capture the user's keystrokes (hidden camera, electromagnetic leaks), the attacker doesn't need Evil Maid attack anymore, and so doesn't need to bother with compromising the system boot anymore. This is because the attacker can just sniff the disk decryption password, and then steal the laptop and will get full access to all user data.

In order to prevent such a “keystroke sniffing” attack, one can use an additional sealed secret on the Anti Evil Maid stick that would be used as a keyfile for LUKS (in addition to passphrase). In this case the knowledge of the sniffed LUKS passphrase would not be enough for the attacker to decrypt the disk. This has not been implemented, although would be a simple modification to dracut-antievilmaid module. If you decided to use this approach, don't forget to also create a backup passphrase that doesn't need a keyfile, so that you don't lock yourself from access to your data in case you lose your stick, or upgrade your BIOS, or something! You have been warned, anyway.

Attacks that are still possible

An adversary that is able to both: sniff your keystrokes (hidden camera, electromagnetic leak) and is also able to copy/steal/seize your Anti Evil Maid stick, can not be stopped. If a non-democratic government is your adversary, perhaps because you're a freedom fighter in one of those dark countries, then you likely cannot ignore this type of attacks. The only thing you can do, I think, is to use some kind of easy-to-destroy USB stick for keeping Anti Evil Maid. A digestible USB stick, anyone?

Another type of attack that is not addressed by Anti Evil Maid is an attack that works by removing the “gears” from your laptop (the motherboard and disk at the very least), putting there a fake board with a transmitter that connects back to the attacker's system via some radio link and proxies all the keyboard/screen events and USB ports back to the original “gears” that execute now under supervision of the attacker. Another way of thinking about this attack is as if we took the motherboard and disk away, but kept all the cables connecting them with the laptop's keyboard, screen, and other ports, such as USB (yes, very long cables). The attacker then waits until the user boots the machine, passes the machine-to-user authentications (however sophisticated it was), and finally enters the disk decryption key. In practice I wouldn't worry that much about such an attack, but just mentioning it here for completeness.

Finally, if our adversary is able to extract secret keys from the TPM somehow, e.g. using electron microscope, or via some secret backdoor in the TPM, or alternatively is able to install some hardware device on the motherboard that would be performing TPM reset without resetting the platform, then such an attacker would be able to install Evil Maid program and avoid its detection by SRTM. Still, this doesn't automatically give access to the user data, as the attacker would need to obtain the decryption key first (e.g. using Evil Maid attack).

Implementation Specific Attacks

In the discussion above we assumed that the trusted boot has been correctly implemented. This might not be true, especially in case of the BIOS. In that case we would be talking about attacks against a particular implementation of your BIOS (or TrustedGRUB), and not against Anti Evil Maid approach.

One typical problem might be related to how CRTM is implemented – if it is kept in a regular BIOS reflashable memory, than the attacker who can find a way to reflash the BIOS (which might be trivial in case your BIOS doesn't check digital signatures on updates) would be able to install Evil Maid in the BIOS but pretend that all hashes are correct, because the attacker controls the root of trust.

Another possible implementation problem might be similar to the attack we used some years ago to reflash a secure Intel BIOS (that verified digital signatures on updates) by presenting a malformed input to the BIOS that caused a buffer overflow and allowed to execute arbitrary code within the BIOS. For such an attack to work, however, the BIOS should not measure the input that is used as an attack vector. I think this was the situation with the logo picture that was used in our attack. Otherwise, even if there was a buffer overflow, the chain of trust would be broken and thus the attack detected. In other words, the possibility of such an attack seems to be rather slim in practice.

What about Intel TXT?

Intel TXT takes an alternative approach to trusted boot. It relies on a *Dynamic* instead of *Static* Root of Trust for Measurement (DRTM vs. SRTM), which is implemented by the SENTER instruction and special dynamic PCR registers that can be set to zero only by SENTER. Intel TXT doesn't rely anymore on the BIOS or CRTM. This offers a huge advantage that one doesn't need to trust the BIOS, nor the boot loader, and yet can still perform a trusted boot. Amazing, huh?

Unfortunately, this amazing property doesn't hold in practice. As we have demonstrated almost 3 years ago (!), it is not really true that Intel TXT can remove the BIOS away from the chain of trust. This is because Intel TXT is prone to attacks through a compromised SMM, and anybody who managed to compromise the BIOS would be trivially able to also compromise the SMM (because it is the BIOS that is supposed to provide the SMI handler).

Thus, if one compares SRTM with Intel TXT, then the conclusion is that **Intel TXT cannot be more secure than SRTM**. This is because if an attacker can compromise the BIOS, then the attacker can also bypass Intel TXT (via a SMM attack). On the other hand, a BIOS compromise alone doesn't automatically allow to bypass SRTM, as it has been discussed in a paragraph above.

It really is a pity, because otherwise Intel TXT would be just a great technology. Shame on you Intel, really!

Alternative approaches to mitigate Evil Maid Attacks

Various people suggested other methods to prevent Evil Maid attacks, so lets quickly recap and discuss some of them...

The most straight forward approach suggested by most people, has been to disable booting from external devices in BIOS, together with locking the BIOS setup with an admin password.

There are two problems with such an approach. First, all the BIOSes have a long history of so called default passwords (AKA maintenance passwords). You don't want to rely on the lack of BIOS default passwords when protecting your sensitive data, do you?

Second, even if your BIOS doesn't have a backdoor (maintenance password), it is still possible to just take your disk away and connect to another laptop and infect its boot partition.

Another suggested approach has been to keep your boot partition on a separate USB stick. This solution obviously doesn't take into account the fact that the attacker might install Evil Maid into your BIOS. Many consumer laptop BIOSes do not require digital signatures on BIOS firmware updates (my Sony Vaio Z, a rather high-end machine, is among them), making it simple to install Evil Maid there (the most trivial attack is to make the BIOS always boot from the HDD instead of whatever other device the user wanted to boot from).

Finally, some people pointed out that many modern laptops comes with SATA disks that offer ability to “lock” the disk so that it could only be used with a specific SATA controller. Using this, combined with setting your BIOS to only boot from your internal disk, plus locking access to BIOS setup, should provide reasonable protection. This solution, of course, doesn't solve the problem of a potential maintenance password in your BIOS. Also being skeptical and paranoid as I am, I would not trust this mechanism to be really robust – I would expect it would be fairly simple to unlock the disk so that it could be paired with another, unauthorized controller, and that this probably is a matter of NOP-ing a few instructions in the controller firmware... In fact it seems like you can buy software to unlock this mechanism for some $50... And apparently (and not very surprisingly) some drives seems to continue on the 'default passwords' tradition.

FAQ

Q: Bitlocker implemented this already several years ago, right?
A: No.

Q: But, two-factor authentication can also be used to prevent Evil Maid, right?
A: No.

Q: Does it make any sense to use Anti Evil Maid without a full disk encryption?
A: No.

Q: Are you going to answer 'no' for each question I ask?
A: No.

Q: Why there are no negative indicators (e.g. a big scary warning) when the unseal process fails?
A: The lack of negative indicators is intentional. The user should keep in mind that if somebody compromised their computer, then the attacker would be able to display whatever she wants on the screen, and especially to skip displaying of any warning messages. The only thing the attacker would not be able to display would be the secret message. Thus, it would make no sense to use negative indicators, as they would likely not work in case of a real attack. One solution here would be to use the unsealed secret as a keyfile for disk encryption (as discussed above), which would make it impossible to decrypt the user disk (and so generally proceed with the boot) without successfully unsealing the secret from the TPM.