Making Anti Evil Maid protection of Qubes OS resistant against shoulder surfing and video surveillance

Patrik Hagara

April 3, 2017

Abstract

Observing the user during early boot should not be sufficient to defeat the protection offered by Anti Evil Maid. Possible avenues for improvement are explored and a solution introduced for both high-risk users, leveraging three extra devices to increase multistage evil maid attack resistance, and a much simpler scheme for users whose threat model does not include such multistage attacks. Detailed time line is put forward for implementing this Google Summer of Code proposal.

1 Introduction

Anti Evil Maid (AEM) is an opt-in feature of the Qubes OS [1] project aiming to provide resistance against so called evil maid type of security circumvention attacks, when an attacker has physical access to target computer.

The current implementation of AEM protection in Qubes OS does not offer any protection whatsoever against shoulder surfing or video surveillance based attacks (referred to as "observation attacks" throughout this document), as both the static trusted boot secret text or image unsealed by the Trusted Platform Module (TPM) and the disk encryption passphrase are seen in the clear on the screen or keyboard, respectively. This allows an evil maid attacker to either modify the boot process and insert a malicious payload while still showing the right static secret or to trivially decrypt computer storage using captured passphrase.

This Google Summer of Code (GSoC) project aims to reduce the impact of observation attacks by changing the way TPM is used for machine to user authentication from simple visual static secret verification to a method resistant to such attacks. Similarly, the user to machine authentication is upgraded from a simple passphrase to two factor authentication (2FA) so that an attacker knowing only a passphrase cannot compromise the system.

Given the unique security properties and capabilities of TPM devices, multiple possible avenues for machine to user authentication present themselves. For instance, the TPM can be used to unseal a secret seed for time based one time passphrase (TOTP) generator, a six digit output of which the user would verify on a separate 2FA device (perhaps a closely guarded smartphone with TOTP application). As for the options of user to machine authentication, TPM-sealed and passphrase-protected Linux Unified Key Setup (LUKS) key files could be used. An attacker would then have to both know the passphrase and obtain a copy of the key file itself, which would be again stored on a small device the user always carries with them and plugs into the computer only after a verified trusted boot.

Additionally to the GSoC project idea [2] written by the Qubes OS team, this proposal strives to also provide the best possible user experience (UX) for users whose threat model does not include multistage software based evil maid attacks.

2 Project goals

The main goal of this GSoC project is to substantially reduce the impact of observation attacks against an AEM enabled Qubes OS installations while not unnecessarily increasing the user per-

ceived complexity of AEM boot process. This goal will be achieved by implementing support for and documenting opt-in machine to user authentication mechanism leveraging Intel Trusted eXecution Technology (TXT) dynamic root of trust measurement (DRTM) and TPM assisted measured boot to unseal a LUKS key file stored on removable media. The LUKS key file will be protected by an additional passphrase in order to provide user to machine authentication mechanism. Fallback support must also be developed to allow recovery in case of LUKS key file loss or unavailability.

Since the primary goal implementation will not be able to fully protect users whose threat model includes multistage software based evil maid attacks, the following features listed should also be implemented either during the three month GSoC work period (should the time permit) or at any point before or after the event.

Stretch goals are to implement, document and provide fallback options for:

- opt-in usage of TOTP tokens for machine authentication
- opt-in storing TPM-sealed and passphrase-protected LUKS key file on a secondary AEM media (inserted only after TOTP code verification)

Both of the above features trade ease of use for increased resistance against software based multistage evil maid attacks. However, while not completely preventing such attacks, they do extend the attack surface by requiring multiple devices which can be independently copied, altered or seized by an attacker. Thus, potential users need to carefully evaluate which of the AEM setup options are most suited for their particular threat model, if any.

Deliverables (applicable to both primary and stretch goals):

- AEM packages with listed features implemented
- documentation for enrolling and upgrading users

GitHub issues (tracked in qubes-issues repository [3]) affected by this project:

- Option to use AEM secret as LUKS key file [4]

3 Implementation

Since Qubes OS aims to be "a reasonably secure operating system" with an emphasis on being user friendly, the primary goal implementation aims to deliver the best possible UX for users whose threat model does not include high enough probability of multistage evil maid attacks. Pros and cons of this choice were discussed [5] on the qubes-devel mailing list. The workflow shall be as follows:

1. user inserts their closely guarded AEM boot media containing tboot bootloader, Xen hypervisor, dom0 Linux kernel, initrd and associated configuration along with TPM-sealed and passphrase-protected LUKS key file and makes computer boot from it
2. measured boot is performed and then, assuming neither the computer firmware nor software has been maliciously modified, the Platform Configuration Register (PCR) values inside the TPM will match the state needed to unseal the LUKS key file
3. once the key file is unsealed, user is prompted to enter their passphrase in order to decrypt the LUKS key file
4. assuming the passphrase supplied by user was able to decrypt the key file, internal computer drive is unlocked and computer continues booting the OS

Workflow of the stretch goal implementation which is able to resist some of the possible multistage evil maid attacks is slightly more involved:

1. user inserts their closely guarded AEM boot media into the computer and makes computer boot from it
2. bootloader performs a measured boot, attempts to unseal TOTP seed using TPM and, if successful, displays a derived six digit code
3. user checks whether displayed TOTP code matches the one computed by their 2FA device
4. if and only if the TOTP codes match, user can safely insert secondary AEM storage media containing the TPM-sealed, passphrase-encrypted LUKS key file

5. the key file is automatically unsealed by TPM
6. user is prompted for key file decryption passphrase
7. assuming correct passphrase was entered and the key file is able to unlock LUKS-protected internal disk, the OS continues booting

In both cases, an attacker observing both screen and keyboard during early boot cannot mount and evil maid attack without either:

- breaking the measured boot via a vulnerability in Intel TXT, TPM or another vital part of the trusted boot process
- copying or seizing the AEM boot media and the LUKS key file (possibly stored on separate media), enabling the attacker to access encrypted computer contents upon seizing it

Applicable to the stretch goal implementation only, following additional attack is possible:

- gaining access to or modifying the TOTP seed, changing time and/or date of the 2FA device or replacing the whole device, allowing exfiltration of AEM boot media contents and LUKS key file, thus enabling the attacker to seize the computer and access its encrypted contents

If using TOTP token for machine authentication, user should take care to first memorize the six digit code shown on the computer screen and only then take out their 2FA device to generate a verification code. This avoids a possible attack whereas an attacker is able to remotely alter computer screen contents in (near) real time.

Please note that AEM protection must be installed on removable media in all cases as observation attacks would capture the TPM Storage Root Key (SRK) passphrase and make it trivial for an attacker to unseal the LUKS key file, should they come to possess it. Combined with observing the passphrase used for the key file, the attacker will be able to unlock the encrypted storage of the target computer upon seizing it (simply copying contents of the internal computer drive is not enough as possessing the TPM chip is still required for key file unsealing).

4 Timeline

May 30 - Jun 6: research and implement a proof of concept (PoC) for generating, enrolling, passphrase-protecting and TPM-sealing a LUKS key file

Jun 6 - Jun 13: merge the above PoC into AEM installer, rebuild the package and test

Jun 13 - Jun 27: implement PoC for disk unlocking using a TPM-sealed, passphrase-protected LUKS key file; research how to integrate it into Dracut initrd and Plymouth; merge the PoC into AEM, rebuild the package and test; first evaluation period starts

Jun 27 - Jul 4: implement fallback options; merge fallback option changes into AEM package, rebuild and test; first evaluation period ends

Jul 4 - Jul 11: document enrollment and upgrade procedures; rebuild the AEM package to include documentation

Jul 11 - Jul 18: engage the community in both reviewing documentation/code and testing the implementation; fix any bugs found

Jul 18 - Jul 24: implement PoC for generating/importing TOTP seed, transferring generated seed into 2FA device, TPM-sealing the seed and generating TOTP codes

Jul 24 - Jul 28: second evaluation period

Jul 28 - Aug 4: implement fallback options for TOTP, displaying and refreshing the TOTP code on Plymouth boot screen; merge that into AEM package, rebuild, test

Aug 4 - Aug 11: implement support for secondary AEM device holding LUKS key file; integrate it into the AEM package, rebuild, test

Aug 11 - Aug 21: implement fallback options for secondary AEM device; merge it into the AEM package, rebuild, test; document enrollment and upgrade steps

Aug 21 onward: engage the community in both reviewing documentation/code and testing the implementation; fix bugs found by the community

Although currently having a twenty hours per week internship, I am confident I will be able to dedicate another thirty to forty hours per week to GSoC, as that would be roughly equal to my schedule during the school year. However, should this be deemed a concern, I could suspend my internship for the three month GSoC working period. No other commitments scheduled during the relevant time frame.

I will be sending progress updates to the qubes-devel mailing list, plus weekly formal summaries.

5 About me

Second year undergraduate in Information Security at the Faculty of Electrical Engineering and Communication, Brno University of Technology. Previously studied Computer Networks and Communication for two academic years at the Faculty of Informatics, Masaryk University.

Passionate about free and open-source software, computer security and digital privacy. Started coding at the age of twelve, presently focusing on Python. Installed Qubes OS in mid-2015 and began using it as a primary OS shortly thereafter.

Almost four years of quality engineering internship at Red Hat. Comfortable working both independently and as part of a team, large code bases are fine, too. Management distance of eight time zones has never been an issue, neither has the fact of not being a native English speaker.

Not submitting GSoC proposal to any other participating organization.

Living in Brno, Czech Republic (UTC+2 / CEST time zone during the GSoC event).

Contact information:

- name: Patrik Hagara
- e-mail: patrihagar@gmail.com
- GPG: `0x5C1E71DF031F9AE5`
- GitHub [6] and LinkedIn [7]

References

[1] `https://www.qubes-os.org/`

[2] `https://www.qubes-os.org/gsoc/`

[3] `https://github.com/QubesOS/qubes-issues`

[4] `https://github.com/QubesOS/qubes-issues/issues/1979`

[5] `https://groups.google.com/d/topic/qubes-devel/0kOgeiTstJ8/discussion`

[6] `https://github.com/phagara`

[7] `https://www.linkedin.com/in/patrikhagara`