Collaborative Discussion – TrueCrypt

Peer Responses

Contents

Post by Michael Justus

Followup

Hi,

Since Microsoft no longer recommends TrueCrypt and recommends using BitLocker--preferably with a TPM--I came across this video from Microsoft Mechanics that described some of the new security enhancement features being brought to the table with Windows 11. Windows 11 relies heavily on a TPM module, and the video does a good job of highlighting why this is so. One takeaway is a deeper understanding of why TrueCrypt was seen as insecure compared to Microsoft's efforts with the use of a UEFI BIOS and TPM. Another interesting point raised concerns Virtualisation Based Security, which enables sensitive portions of an OS to be "containerised" away from hackers.

Windows 11 Security - Our Hacker-in-Chief Runs Attacks and Shows Solutions

https://www.youtube.com/watch?v=tg9QUrnVFho&t=915s

Reply from Cathryn Peoples

Re: Followup

Excellent reference here to the IEEE 1619 disk encryption algorithm, demonstrating your wider reading, Michael.

Could you make a few suggestions as to how a couple of the TrueCrypt challenges may be repaired?

Best wishes,

Cathryn

Response to Kikelomo Obayemi

Post by Kikelomo Obayemi

Hello Michael.

Thanks for this informative post. I like your reference to the use of Trusted Platform Module (TPM) by Bitlocker which got me researching about the feature. The TPM is a secure chip (microprocessor) that is designed to provide hardware-based cryptography (Microsoft ,2021). Some of the advantages of using the TPM technology include: the ability to generate, store and limit the use of cryptographic keys (Microsoft, 2021) and the possibility of storing platform measurements to ensure that the platform remains trustworthy (Trusted Computing Group, 2008)

Moia and Henrique (2016) makes a comparison of 10 encryption tools using a number of security features namely: Algorithm, Key recovery, Algorithm operation modes, Open source, Encryption Subjects, Multi factor authentication, Operating systems, portable software, plausible deniability, file attribution encryption, security levels, hidden containers, TPM use and key stretching.

The ten tools compared are AxCrypt, GnuPG, 7-Zip, R-Crypto, DiskCryptor, LUKS, VeraCrypt, CipherShed, BitLocker and File Vault 2. Worthy of note is the fact that of all ten, only BitLocker uses TPM (Moia and Henrique, 2016). Given the advantages of TPM, some of which are mentioned above, have you ever wondered why these other tools, particularly well-known ones such as

VeraCrypt and CipherShed do not implement this feature? I would love to hear your thoughts.

References

Microsoft (2021) Trusted Platform Module Technology Overview. Available from: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/trusted-platform-module-overview [Accessed 11 October 2021]

Moia, V.H.G. and Henriques, M.A.A. (2016) A comparison of encryption tools for disk data storage from digital forensics point of view. Available from: http://cyberforensicator.com/wp-content/uploads/2017/04/A-comparison-of-encryption-tools-for-disk-data-storage-from-digital-forensics-point-of-view.pdf [Accessed 11 October 2021]

Trusted Computing Group (2008) Trusted Platform Module Summary. Available from: https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/ [Accessed 11 October 2021]

Reply to Kikelomo Obayemi **from** Michael Justus

Hi Kikelomo,

Thank you for the interesting article linked by Moia and Henrique (2016). You raise a very interesting question about why specific tools do not utilise a TPM chip, and according to the report, Bitlocker is the only full-disk encryption tool. Reading the article by Trusted Computing Group (2021), it seems that a TPM chip fulfils a specific role of maintaining security keys and certificates. Still, they note that TPM does not control software on a computing device. Then, looking at a report by Muñoz and Fernandez (2020), they note TPM's role concerns establishing a "chain of trust" that enables the TPM to validate the BIOS, which validates the operating system turn validate software applications. Therefore, from this perspective, it makes sense that tools like AxCrypt, GnuPG, or 7-Zip would not (ever?) leverage a TPM module since they're focused on the encryption of file data. In contrast, DiskCrypto, LUKS and File Vault2 could very likely benefit from the security introduced by a TPM module since the module can keep their private application keys secure--since TPM protect against data theft and remote attestation of online transactions. Considering the view of Muñoz and Fernandez, TPM, as part of the trusted computing movement, is a valuable source of security, but one that requires considerable security expertise to leverage well.

References

Muñoz, A. & Fernandez, E.B. (2020). TPM, a pattern for an architecture for trusted computing. Proceedings of the European Conference on Pattern Languages of Programs, 2020:1-8.

Trusted Computing Group, (2021). Trusted Platform Module (TPM) Summary. Available from https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/ [Accessed 24 Oct. 2021]

Response to Kieron Holmes

Post by Michael Justus

Hi Kieron,

The statements you raised regarding code quality and its role in leading to the demise of TrueCrypt software are interesting. These are two interesting points as they have an impact on the part of software developers. Bosu (2014) considers that vulnerabilities enter a code based on several factors: how many developers change a source file at one time, how the changes deviate from the original author's implementation and how well developers network with each other. The report by Baluda et al. (2015) also investigated how to reduce the complexity of code, reducing the probability of vulnerabilities. Their report referenced the use of code peer code reviews to aid developers in identifying vulnerable code changes. For instance, they state that reviewers could search for suspected issues using specific search terms in the code since developers generally use the words that correlate strongly to vulnerabilities.

With this context, do you think the authors of the TrueCrypt project would have been successful if they had implemented recommendations such as peer reviews, test cases and simple code?

References

Bosu, A. (2014) Characteristics of the vulnerable code changes identified through peer code review. Companion Proceedings of the 36th International Conference on Software Engineering:736-738.

Baluda, M., Fuchs, A., Holzinger, P., Nguyen, L., Othmane, L.b., Poller, A., Repp, J., Spath, J., Steffan, J., Triller, S. & Bodden, E. (2015) Security Analysis of TrueCrypt. Available from: http://home.eng.iastate.edu/~othmanel/files/CPRE562/Truecrypt.pdf [Accessed 18 Oct. 2021].

Response to Kikelomo Obayemi

Post by Michael Justus

Hi Kike,

I found terrific your conclusion that both reports did not conclude that TrueCrypt was insecure. Also, the summary statement about Zhang et al. (2019) cautioned readers about two critical vulnerabilities (identified by Google Project Zero) that grant local privilege escalation. And because of these vulnerabilities, the use of TrueCrypt is strongly discouraged. This was quite helpful to understand a reason to move away from using TrueCrypt.

Based on your post, software audits are a valuable contribution to ensuring vulnerabilities are kept to a minimum. Lin et al. (2020) consider a mitigation solution is to apply vulnerability detection to solutions before deployment. Detection techniques could leverage approaches such as “rule/template-based analysis”, source code analysis—it suffers from many false positives—and fuzzy testing. They also mention the use of machine learning for automated vulnerability discovery. Although, the use of machine learning to detect vulnerabilities is still dependent on human expertise, experience and domain knowledge. Recurrent neural networks are one deep learning technique that may come in quite handy (Dahl et al., 2020)

What, in your opinion, could the TrueCrypt authors have done to ensure their project did not fall by the wayside, given that both reports did not conclude it was an insecure product?

References

Dahl, W.A., Erdodi, L. & Zennaro, F.M. (2020). Stack-based Buffer Overflow Detection using Recurrent Neural Networks. arXiv:2012.15116.

Lin, G., Wen, S., Han, Q.L., Zhang, J. & Xiang, Y. (2020). Software vulnerability detection using deep neural networks: A survey. Proceedings of the IEEE, 108(10):1825-1848.

Zhang, L., Deng, X. and Tan, C., 2019, October. An Extensive Analysis of TrueCrypt Encryption Forensics. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering 86: 1-6. DOI: https://doi.org/10.1145/3331453.3361328

Response to Andrey Smirnov

Post by Cathryn Peoples from Andrey Smirnov Parent of this post↑

Re: Initial Post

Hi Cathryn,

Thanks for the question. There were several code quality issues uncovered by the TrueCrypt audits. Most of the them are far from esoteric and should have been relatively straightforward to address. For example, any experienced C/C++ programmer would be able to fix mismatches between signed and unsigned integer types (found in both the bootloader and the driver code), or ensure that array bounds are checked before performing *read* and *write* operations in the bootloader decompressor. The use of deprecated (and insecure) APIs and suppression of compiler warnings uncovered in the audits is another red flag that hints at certain carelessness and overall lack of oversight. While upgrading dependencies in a large codebase and addressing every complier warning can be a time consuming process, the long-term benefits of doing this easily outweigh the initial maintenance costs. Moreover, most modern IDEs can automate some of these tedious tasks and therefore simplify the overall process.

Even though none of the deficiencies described above directly resulted in security problems, it is rather surprising that they were allowed to slip into production code. Normally, I would expect quality issues of this sort - along with the overall lack of comments in TrueCrypt's source code - to be flagged during regular code review. With that in mind, introducing a more rigorous quality assurance process would be my primary recommendation. In addition to manual code reviews, I would also advise on using a static code analysis tool such as cppcheck. Static analysis tools can be instrumental in highlighting general readability issues, which was another problematic area noted by the engineers performing the TrueCrypt audits.

Reply to Andrey Smirnov from Michael Justus

Hi Andrey,

The closing statement referred to the role of code reviews in detecting vulnerabilities before deployment into production. Paul et al. (2021) also acknowledge that several popular open-source projects such as Chromium, Android, Qt, oVirt and Mozilla integrate code reviews in their software pipeline. Although the code reviews prevent several security defects, they note that many projects still report post-release vulnerabilities and flaws as logged by the Common vulnerabilities and Exposure (CVE) database. In unison with the statement "it is rather surprising that they were allowed to slip into production code", Paul et al. (2021) state that "project managers may wonder if there was any weakness or inconsistency during a code review that missed a security vulnerability".

In their second research question ("Which factors influence … identification of security defects during a code review"), their most significant finding is that the greater the number of changes to be reviewed, the higher the probability of missing a vulnerability. On the flip side, the probability of identifying a vulnerability increases if more time is spent on reviews, more reviewers participate, cyclomatic code complexity is reduced, and reviewers are familiar with the code. Given these last points identified by Paul et al. (2021) and the time-is-money concerns of business, perhaps it is no wonder that software constantly requires patching?

References

Paul, R., Turzo, A.K. & Bosu, A. (2021). Why security defects go unnoticed during code reviews? a case-control study of the chromium os project. IEEE/ACM 43rd International Conference on Software Engineering (ICSE):1373-1385.