Michael Denzel

Malware Tolerance

1 Motivation

Recent attacks on computer systems, such as the Debug Register (DR) Rootkit [1] and the Advanced Configuration and Power Interface (ACPI) Rootkit [2], show that we cannot be completely sure that a platform is free of malware. In some cases it is possible to defend against those attacks [3], but computer architecture limits defence techniques. In the end, we cannot assume we can trust the entire computer.

This raises the question of whether we have to trust the malware-infected systems that we use. Can we use a malicious system securely? In what circumstances, and for what purposes, might it be possible to securely use a platform that is suspected to have malware?

2 Purpose

The thesis aims to clarify the question of how to use a computer, that has malware, in a secure way. Even though this seems unlikely there are examples that demonstrate that this can be done in special scenarios. The following objectives were set to earn more knowledge in this topic.

Classify cases: First, a classification of scenarios will be examined. As an example, e-mails, online-banking, and electronic voting need to be ranked for their level of security. Surprisingly, online-banking is not as critical as usually thought of, because incorrect or fraudulent transactions can be reverted. Accesses to e-mail in contrast cannot be reversed and might thus be more critical depending on its content. Scenarios have to be categorized for confidence, integrity, and daily frequency.

Identify techniques: Next, techniques and protocols to achieve security will be identified and verified. This step also includes prototype implementations if applicable.

Systematic evaluation: Lastly, a systematic scheme to evaluate all kinds of scenarios is needed. For that, especially usability is important, because well-working but inconvenient techniques will not have big impact.

3 Ideas

Additional (embedded) devices: One idea, which partly already exists, is to use another device in combination with the affected platform. There are embedded devices like chips inside credit cards and other smartcards, security tokens for online banking, and even phones. E. g. the German Post is sending SMS with one-time passwords to enable customers to collect packages at self-service points.

Trusted security agent: A trusted system is helpful for verifying or computing. The term system stands for a key server, Virtual Private Server (VPS) or even the router at home. One idea is an approach similar to the Kerberos authentication server. A device, e. g. the router at home with additional software, could do some pre-processing and enable client and server to communicated authenticated and/or encrypted.

Further, humans are a strong security agent, because friends are a very trustworthy medium. Nevertheless, human beings cannot compute complicated results nor have a high throughput.

Obfuscation: This technique could also work even when hardware and operating system are malicious. Assuming we trust own code or open-source programs, there are at least two possibilities:

- Reveal the meaning of a computation only after the operation took place. E. g. a multiplication is equal to various additions and only the result will show the true meaning.
  $4 \cdot 3 = 4 + 4 + 4 = (4 + 4) + 4$
  At the stage of $(4 + 4)$ it is not clear that we compute $4 \cdot 3$. Therefore, an attacker can only sabotage every computation or random ones which will lead to system instability.
  In combination with techniques used in multiprocessor programming where interrupts at any point do not damage the data structure, techniques to create a resistant system should be explored.
- There is an idea we named "Reverse Sandbox". Instead of locking the malicious code in, we will try to lock it out and create a save environment to hide computations.

4 Evidence

Even though this approach seems impossible there are proofs that at least some scenarios are solvable.

Analogy: Biased coin

A good example for the basic idea is a biased coin which has unknown probability $p \neq 0.5$. Can we still make it a "fair" coin? This exercise appears unsolvable, however, it is not. All we need to do is to flip the coin multiple times and find two event sequences that have the same probability. In conclusion, we can wrap the biased coin to "remove" the bias.

Already known scenarios:

In IT security, techniques to use untrusted machines already exist. One example is online-banking with a security token. As long as the server and the security token are not corrupted the integrity of the transaction is guaranteed. Attention, privacy is not guaranteed. An attacker controlling the computer will learn the online-password and can then read all transactions. Nevertheless, the attacker cannot create ("write") valid ones.

Additionally, there is evidence in recent research:

MP-Auth [4] hides long-term passwords from untrusted machines by employing (trusted) mobile phones.

Furthermore, there is a method to use online services of a server by only providing it with encrypted data [5]. Operations are carried out on this encrypted data and the result is exclusively decrypted on the client.

Ultimately, Du-Vote [6] shows that an untrusted system can securely process the very critical scenario of voting.

5 Timeline

There are four general steps to achieve during the work. Figure 1 shows their distribution over the six semesters.

1. Classify scenarios, devices, etc.
2. Search for techniques, verify and evaluate them (including prototype implementations if applicable).
3. Create a systematic evaluation process paying attention to security-usability balance and evaluate already existing, as well as new, techniques and prototype implementations with this scheme.
4. Write the thesis.

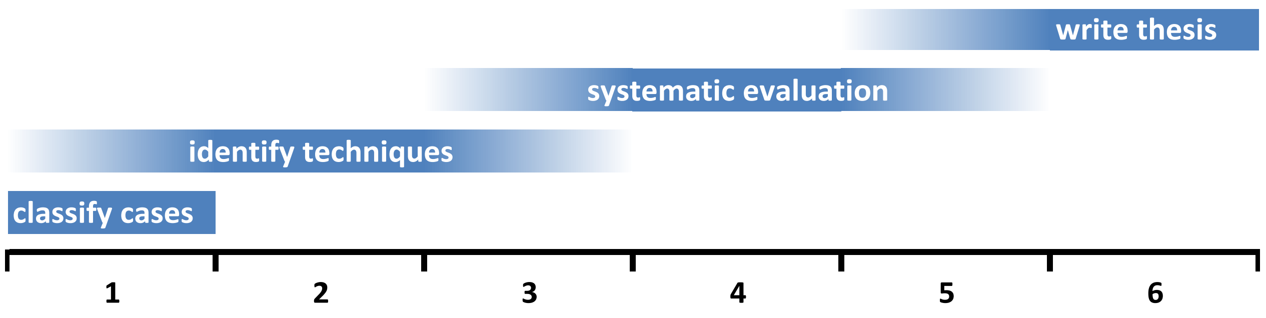

Figure 1: Timeline over six semesters

6 Conclusion

All in all, this work is connected to my previous work [3] as ACPI Rootkits show that there is a barrier if we cannot trust our hardware. We cannot verify every computation to make sure the system is working correctly and is not corrupted. Thus, it is essential to develop new ideas in order to deal with incorrect or manipulated behaviour.

Evidence for malware tolerant systems is linked to my prospect supervisor Prof. Dr. Ryan as he developed one example of malware tolerance. To generalise malware tolerance, the missing trust has to be compensated. Imaginable are special protocols, software designs, or, most likely, embedded devices.
Dr. Garcia is dealing with the security of embedded devices and a cooperation throughout the dissertation would be desired.

In this thesis, the question for malware tolerant system should be settled or narrowed down to approaches and scenarios that are possible.

7 References

[1] "Immunity Debugger Team", „Linux 2.6 rootkit released,“ http://seclists.org/dailydave/2008/q3/215, 2008.

[2] J. Heasman, „Implementing and detecting an acpi bios rootkit,“ in s *Black Hat Federal*, 2006.

[3] M. Denzel, „Detecting and Analysing compromised Firmware in Memory Forensics,“ Friedrich Alexander University of Erlangen-Nuremberg, Nuremberg, 2013.

[4] M. Mannan and P. C. v. Oorschot, „Using a Personal Device to Strengthen Password,“ Springer, 2007.

[5] C. Boyens and O. Günther, „Using online services in untrusted environments: a privacy-preserving architecture,“ ECIS, 2003.

[6] G. S. Grewal, M. D. Ryan, L. Chen and M. R. Clarkson, „Du-Vote: Remote Electronic Voting with Untrusted Computers,“ Under review by IEEE Security and Privacy, 2014.