R209 Essay: Adversarial reasoning

Chongyang Shi (*cs940*)

October 13, 2017

This essay provides a synthesis of three papers from distinct eras of computing. Adversarial reasoning, a process of applying computational approaches to inferring and anticipating an enemy's perceptions, intents and actions [1], was applied to three very different systems: an early mainframe Multics system in the 1970s [2], a modern computerised car [3], and virtual machines running on a hypervisor server [4].

1 Summaries of research

Research by Karger and Schell [2] was a comprehensive vulnerability analysis of the Multics mainframe system, one of the first to implement multi-level security at the time. Karger and Schell discovered that despite a sound specification of security control [2, 3.1], non-conformance [2, 3.2.1] or even optimisation attempts [2, 3.3.2] in hardware and software implementations have introduced vulnerabilities into the system. These vulnerabilities allow a perpetrator to access unauthorised information, insert trap doors, or deny system use by causing crash, while being able to modify audit trails to avoid detection [2, 3.4.4]. The cost of discovering the vulnerabilities was low [2, Tb. 3]. Possibilities of security compromise during the development and servicing of the system were also identified. Recommendations were made towards making Multics more secure.

Research by Koscher et. al. [3] studied the scale of vulnerabilities of modern vehicles with computerised control. Both laboratory and field tests were conducted to demonstrate the grave possible compromises on vehicle safety, as a result of weak security requirements in system standards [3, IV. B.] as well as further non-compliant implementation of security deviating from standards [3, IV. C.]. It was discovered that almost all compliant security features could be bypassed through brute force attacks [3, IV. B.] or reverse engineering [3, V. A.]. Researchers were able to produce a wide range of dangerous vehicle behaviours through the vulnerabilities. Concerns on economical and legal aspects of vehicle security were discussed.

Razavi et. al. [4] developed a hardware-based attack on software security systems such as RSA, through the use of shared physical memory on hypervisor environments. Combined software and hardware attacks were deployed by exploiting Linux memory optimisation features and intentionally inducing memory errors. Under ideal conditions it is possible to weaken and defeat public key securities with strengths currently considered as secure [4, Sec. 5]. Both hardware and software mitigations to the attack developed were discussed. The attack however requires very specific conditions to work, namely a high-memory VM [4, Fig. 4] running on the same hypervisor as the victim, with KSM and THP turned on [4, 4.3.1], on specific hardware [4, 6.1.1], along with a notable chance to fail [4, 5.2]. Therefore, a survey on the size of potential victims could help assessing the practicalness of this attack.

2 Key themes of research

2.1 Was adversarial reasoning applied at all during the design of systems?

For some system designers, adversarial reasoning is a key part of the design process. While in other cases, it may not have been considered at all, often with disastrous consequences. An example of adversarial reasoning can be seen in the specification of Multics [2, 3.1]. Unfortunately, errors in both hardware and software implementation [2, 3.2.2, 3.3.1] resulted in the vulnerabilities identified by the authors. This demonstrates the need to apply adversarial reasoning on both the specification and the implementation of a system.

On the two more recent systems studied, adversarial reasoning was either absent or based on a more restrictive threat model. The vehicle control system studied by Koscher et. al. [3] lacked protection against short or well-hidden physical access to the intra-vehicle network, adding to the fact that implemented security features are relatively easy to compromise [3, IV. B.]. Software security systems on platforms attacked by Razavi et. al. [4] were not resistant against memory-based attacks, despite the fact that integrity verification could have mitigated the issues without altering the underlying hardware [4, 6.2].

2.2 Bypassing software security through hardware attacks

It was demonstrated in all three papers that by compromising the underlying hardware, a software system can be significantly weakened in security. Address check bypass in Multics was made possible by an error in hardware implementation [2, 3.2.2]. Design flaws such as broadcast of all packets allowed circumvention of challenge-response features in issuing unsafe commands to car components [3, V. B.]. While the downsizing of DRAM modules gave realistic possibilities to Rowhammer attacks on physical memory [4, 6.2], thus weakening software cryptography systems. Hardware has become a significant point of entry for unconventional attacks to occur.

2.3 Performance and economic constraints could compromise security

Due to the desire to improve performance or the need to make designs more economical, vulnerabilities were introduced into the systems studied by the three papers. In Multics, software optimisations made to speed up fault processing introduced a privilege escalation exploit [2, 3.3.2]. In computerised vehicle control, weak security of commonly used CAN bus can be partly attributed to economic pressure on design [3, II. A.]. And in Linux hypervisors, default-on features for memory performance optimisation such as KSM and THP made the attack by [4] possible. The high cost nature of error-correcting memory also increased the number of vulnerable systems [4, 6.1.1].

3 Ideas of current context

Due to the large time gap between the study on Multics [2] and the two later studies [3][4], two ideas from the Multics study were reflected very differently in the later studies.

The ability for a perpetrator to completely erase the audit trail of his attack was as much possible in 1970s [2, 3.4.4] as in 2010s [3, VI. C.]. The method to achieve this was also largely the same: deleting records through privilege escalation. The lack of an audit trail can make detection and investigation of intrusions difficult or impossible. In order to protect audit trails in the case of an intrusion, recent designs and researches focus on temper-resistant hardware audit systems either hardcoded in the processor [5] or securely tied to the system [6].

Karger and Schell considered physical memory exploits impractical, due to the little or no control users exercise on hardware. However, in modern computer systems, this has become possible due to unintended consequences of memory optimisation techniques as well as downsizing of memory modules, as applied by Razavi et. al. [4]. Similar attacks under modern memory systems are now common, such as one utilised earlier by Govindavajhala and Appel [7] against Java virtual machines.

Another general observation made by Karger and Schell [2, 3.3.1] remains largely accurate to this day: while it is often possible to prove the security of a security concept or protocol, implementations of such often introduce vulnerabilities into the system, making security assurances impossible. The presence of users can also compromise the security of a system [8]. Most current security researches concentrate on verifying the security of a protocol, rather than the security of an implemented system such as Multics, as the former is inevitably easier to approach.

4 Literature review

References

[1] A. Kott and M. Ownby, "Toward a research agenda in adversarial reasoning: Computational approaches to anticipating the opponent's intent and actions," *arXiv preprint arXiv:1512.07943*, 2015.

[2] P. Karger and R. Schell, "Multics security evaluation, volume ii: Vulnerability analysis." Electronic Systems Division, Air Force Systems Command, Tech. Rep. ESD-TR-74-193, v. II, 1974.

[3] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on.* IEEE, 2010, pp. 447–462.

[4] K. Razavi *et al.*, "Flip feng shui: Hammering a needle in the software stack." in *USENIX Security Symposium*, 2016, pp. 1–18.

[5] D. Bordsen, T. Cooper, R. Esson, M. Hill, J. Jordan, J. Kessler, D. Konrad, R. Sipple, R. Swenson, J. Torgerson *et al.*, "Cache memory with data compaction for use in the audit trail of a data processing system having record locking capabilities," Mar. 9 1993, uS Patent 5,193,162. [Online]. Available: https://www.google.com/patents/US5193162

[6] C. N. Chong, Z. Peng, and P. H. Hartel, "Secure audit logging with tamper-resistant hardware," in *Security and Privacy in the Age of Uncertainty.* Springer, 2003, pp. 73–84.

[7] S. Govindavajhala and A. W. Appel, "Using memory errors to attack a virtual machine," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on.* IEEE, 2003, pp. 154–165.

[8] A. Adams and M. A. Sasse, “Users are not the enemy,” *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.