R209 Essay: Adversarial reasoning

Chongyang Shi (*cs940*)

October 13, 2017

This essay provides a synthesis of three papers from different 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: a mainframe system Multics 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 most secure 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 the implementation have introduced vulnerabilities. These vulnerabilities allow a perpetrator to access unauthorised information, insert trap doors, or deny system usage, while modifying audit trails to avoid detection [2, 3.4.4]. The cost of identifying the vulnerabilities was low [2, Tb. 3]. Security vulnerabilities during the development and servicing of the system were also identified.

Research by Koscher et al. [3] studied the scale of vulnerabilities in modern vehicles with computerised control. Both laboratory and field tests were conducted to demonstrate the grave dangers to vehicle safety, as a result of weak security requirements in system standards [3, IV. B.] as well as non-compliant security implementations [3, IV. C.]. It was further discovered that almost all compliant security features could be bypassed [3, IV. B., 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 to exploit Linux memory optimisation features and intentionally induce memory errors. Under ideal conditions it was possible to defeat public key security with levels of strength 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, such as a high-memory VM [4, Fig. 4] running on the same hypervisor, with KSM and THP turned on [4, 4.3.1], on specific hardwares [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 Non-application or limited application of adversarial reasoning

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]. However, errors in both hardware and software implementations [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 recent systems studied, adversarial reasoning were either absent or based on a more restrictive threat model. The vehicle control system studied by Koscher et al. [3] lacked protection against short-period or well-hidden physical access to the intra-vehicle network, in addition to the easy-to-compromise nature of implemented security features [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 security features when issuing unsafe commands to car components [3, V. B.]. Downsizing of DRAM modules gave realistic possibilities to Rowhammer attacks on physical memory [4, 6.2], thus weakening software cryptography systems. In all, hardware have become a significant point of entry for unconventional attacks.

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, a privilege escalation exploit was introduced by software optimisations made to speed up fault processing [2, 3.3.2]. In computerised vehicle control, weak security of commonly used CAN buses can be partly attributed to economic pressures 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 Razavi et al. [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 have been largely the same: delete records through privilege escalation. The lack of an audit trail can make detection and investigation of intrusions difficult. In order to protect audit trails in the case of an intrusion, recent designs and researches focused on temper-resistant hardware auditors 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 exercised on hardware. However, in modern computer systems, this is now possible due to unintended consequences of memory optimisation techniques as well as downsizing of memory modules, as demonstrated 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, their implementations 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 protocol designs, while verifiable implementations tend to take the form of easier-to-analyse sub-systems such as the security kernel suggested.

4 Literature review

Karger and Schell [2] developed the concepts of back-doors and malicious code planting [9, Sec. 2], which became the focus of modern security research [10]. In 2002, they published a survey paper [11] summarising the changes in security systems since the original Multics study. In this study, Karger and Schell considered the lack of operating system-level security in modern personal computers a step back from Multics. New terminologies such as buffer overflow and malware were discussed with regarding to their association with the original study.

In addition to wide-spread media attention, field-tested attacks on vulnerable vehicle control systems by Koscher et al. [3] also provided a starting point for further practical research. Checkoway et al. developed a practical method for external interface attacks on vehicle ECUs without physical access [12]. To mitigate the lack of authentication in CAN bus communications, an authentication protocol was proposed by Herrewege et al. [13]. The methodology of original analysis was also repeated on many other Internet of Things devices [14].

Since Razavi et al. [4] established the method for attacking public key systems through memory massaging, it was found that the same hardware vulnerability can be exploited by unprivileged users to gain kernel privilege and take over the entire system [15, II. B.]. The vulnerability was also successfully exploited on mobile platforms [16]. Solutions have been

proposed to mitigate this problem, such as pre-allocation of memory and processor cores by Szefer et al. [17].

(1245 words according to texcount.)

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 *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 *et al.*, "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.

[9] D. E. Bell, "Looking back at the bell-la padula model," in *Computer Security Applications Conference, 21st Annual*. IEEE, 2005, pp. 15–pp.

[10] M. Egele *et al.*, "A survey on automated dynamic malware-analysis techniques and tools," *ACM computing surveys (CSUR)*, vol. 44, no. 2, p. 6, 2012.

[11] P. A. Karger and R. R. Schell, "Thirty years later: Lessons from the multics security evaluation," in *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*. IEEE, 2002, pp. 119–126.

[12] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[13] A. Van Herrewege *et al.*, "Canauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011.

[14] A.-R. Sadeghi *et al.*, "Security and privacy challenges in industrial internet of things," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[15] O. Mutlu, "The rowhammer problem and other issues we may face as memory becomes denser," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 1116–1121.

[16] V. van der Veen *et al.*, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1675–1689.

[17] J. Szefer *et al.*, "Eliminating the hypervisor attack surface for a more secure cloud," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 401–412.