University of Oslo
INF3510 Information Security
Spring 2014
Workshop Questions and Answers

Lecture 4: Computer Security

Question 1

"A trusted system or component is one that can break your security policy".
Explain the meaning of this proposition.

Answer

If the system is trusted, then it is relied upon to enforce the security policy. So there can be a breach of security policy when the trusted system does **not** work as expected. A non-trusted system on the other hand is not relied upon to enforce the security policy, so when it breaks it does not lead to a breach of security policy.

Question 2

Attempts of physical attacks against security hardware components of a computer system can not be prevented when the system is physically accessible to attackers. However, such physical tampering can be prevented with tamper proof devices. Look at the specification for the IBM 4765 Secure Coprocessor at
http://www-03.ibm.com/security/cryptocards/pciecc/FAQ8xxxxxxxRC.shtml

a. In which situations will the IBM 4765 Secure Coprocessor self-destruct, i.e. zeroize memory and permanently disable itself?
b. Suggest mechanisms for tamper resistance of security hardware.

Answer

a. Reasons for self-destruction of the IBM 4765 Secure Coprocessor are:
    - The on-board batteries have run out of power without timely replacement.
    - A too high or too low temperature has been deteceted.
    - A too high or too low voltage has been detected.
    - Physical damage to the shield has been detected.
b. Tamper resistant screws. Hard shield. Remote reporting. Security by obscurity e.g. in the form of confused chip architecture. Make tampering illegal.

Question 3

TPM (Trusted Platform Module) is specified by the TCG (Trusted Computing Group).

a. Explain the three main TPM supported services: 1) Secure/authenticated boot, 2) Sealed storage, 3) Remote attestation.
b. Which TPM service is used by the Windows Bitlocker disk encryption application?
c. Which security threat to Bitlocker does the TPM mechanism address?
d. Assume that a drive contains a zero-day vulnerability that potentially could be exploited to take control of the computer. Say Yes/No whether the boot security for Windows 8 can protect against this threat, and explain why / why not ?

Answer

a. 1) Secure boot: Only boot when software has integrity, Authenticated boot: Boot anyway, and report the integrity status of the software. 2) Sealed storage: decryption with secret keys only with correct integrity, 3) Remote Attestation: reporting to an external party the integrity status of software and data.
b. Bitlocker uses sealed storage.
c. Bitlocker with TPM can protect against the following threats:
   - Removing an encrypted harddrive from computer. It will not decrypt outside the original computer, even with password or USB key, because the TPM is missing.
   - The loss of integrity of specific software or data files on the computer, determined by non-match of the corresponding measurement values with the values stored in the PCR registers.

   These threats are not very relevant. So Bitlocker with TPM is rather meaningless. It is not necessary to run Bitlocker with TPM, it works fine without.
d. Secure boot does not protect against zero-day infection during runtime, because it only protects the boot process. A zero-day malware infection in a software module protected by secure boot will be detected next time the computer boots. A zero-day malware infection in a software module not covered by secure boot will not be detected by secure boot, and will by definition not be detected by anti-malware.

Question 4

A detailed description of the protection mechanisms in the Intel microprocessors can be found in the Intel 64 bit Software Developer Manuals available from http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html (Sections 5.5 – 5.13 in Volume 3A.)
The Intel microprocessor primarily provides 4 protection rings (0-3), plus an additional Ring -1 for virtualization support.

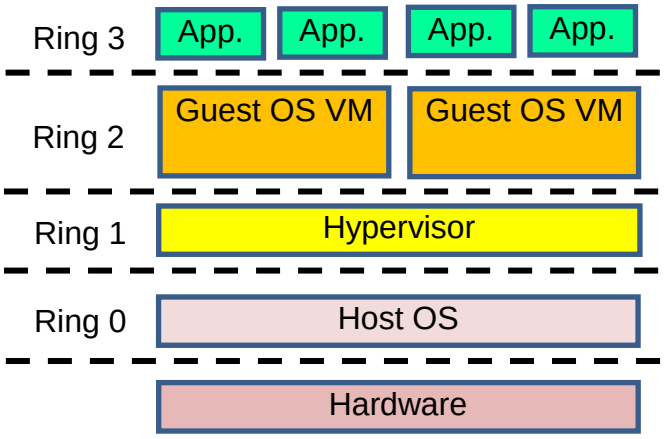
a. What is the main principle for allowing a process running in ring *m* to access a memory segment specified as ring *n* ?
b. Which protection rings are actually used in Linux and Microsoft Windows?
c. What is the correspondence between protection rings and user/supervisor modes in Linux/Windows?

Answer

a. A process running in ring *m* can directly access a memory segment in ring *n* only if m ≤ n. Otherwise it must call an OS gate (OS function). Privileged instructions (e.g. setting values in specific register) can only be executed by processes running in ring 0 (or -1).
b. Only ring 0 and 3 are normally used (ring -1 used by hypervisor). Ring 0 for Kernel, OS and Drivers. Ring 3 for applications.
c. Application processes run in **user mode** (Ring 3) in Linux and Windows. Processes running in **kernel/supervisor mode** (Ring 0) are **root** in Linux, and **Admin** in Windows.

Question 5

a. An alternative to introducing Ring -1 for virtualization could have been to instead use Ring 1 and 2 as illustrated in the diagram below.



Discuss how practical or meaningful this would have been.

Answer

a. It would be very difficult to implement. The main problem is that OSs expect to be able to execute privileged microprocessor instructions that are only available in Ring 0. It would require radical OS redesign to make it run as VM in Ring 1 or 2.

Question 6

a) Assume that an attacker has found a way to produce input data that crashes a program due to a buffer overflow error. The input could e.g. be a particular MSWord Document in MSOffice. Roughly explain how an attacker could exploit this bug to create an attack, assuming that NX (No-Execute) is **not** enabled.
b) Explain the technique used to find bugs that cause program crashes which potentially can be used to create buffer overflow exploits.

Answer:

a) The attacker must determine the structure of the stack frame where the input arguments are stored in memory, and craft the input arguments in such a way that they overwrite the EIP (Extended Instruction Pointer) with a specific address which points to a memory location where the attacker has placed malicious code.
b) The technique is Fuzzing or Fuzz Testing, which consist of generating many different random inputs to applications in order to make them crash, which could indicate a buffer overflow bug.