CIS 4360
Secure Computer Systems

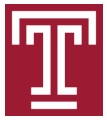
Attacks against Boot and RAM

Professor Qiang Zeng

Spring 2017

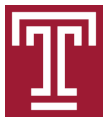
Previous Class

- BIOS-MBR: Generation I system boot
  - What BIOS and MBR are?
  - How does it boot the system? // Jumping to MBR
  - How does multi-boot work? // Chain-loading
- The limitations of BIOS and MBR
  - Disk, memory, file system, multi-booting, security, …
- UEFI-GPT: Generation II system boot
  - What UEFI and GPT are?
  - How does it boot the system? // UEFI boot manager
  - How does multi-boot work? // separate dirs in ESP



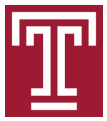
Limitations of BIOS-MBR

- MBR is very limited
  - Support ~2TB disk only
  - 4 primary partitions at most (so four OSes at most)
  - A MBR can store only one boot loader
- BIOS is very restrictive
  - 16-bit processor mode; 1MB memory space (little spare space to accommodate a file system driver)
  - Blindly executes whatever code on MBR



UEFI vs. BIOS

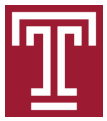
- Disk partitioning schemes
  - GPT (GUID Partition Table): part of UEFI spec.; to replace MBR
  - MBR supports disk size 2^{32} x 512B = 2TB, while UEFI supports much larger disks (2^{64} x 512B = 8,000,000,000 TB)
  - MBR supports 4 partitions, while GPT supports 128
- Memory space
  - BIOS: 20-bit addressing; UEFI: 32-bit or 64-bit
- Pre-OS environment
  - BIOS only provides raw disk access, while UEFI supports the FAT file system (so you can use file names to read files)
- Booting
  - BIOS supports boot through boot sectors (MBR and VBR)
  - UEFI provides a boot partition of hundreds of megabytes (and boot manager and secure boot)



Previous Class

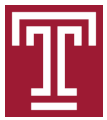
How does dual-booting of Linux and Windows work in UEFI-GPT?

Each vendor has a separate directory storing its own boot loader code and configuration files in the ESP (EFI System Partition). So they will not “step on” each other.



One More Thing about UEFI: UEFI-MBR

- Through the Compatibility Support Module (CSM) mode, UEFI provides legacy BIOS compatibility and can boot from a MBR partitioned disk. It is called UEFI-MBR boot
- In your firmware setting, you can choose UEFI boot or CSM boot as the boot mode
  - What is firmware setting?
  - Press F2, Delete etc. to enter BIOS setting☺
- In CSM mode, UEFI boots the system exactly the same as BIOS-MBR booting, that is, jumping to the code in the MBR sector



Previous Class

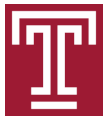
Alice has a Windows 7 installed on a bootable USB using an old non-UEFI computer, can she boot the Windows 7 on a new laptop she recently purchased?

Yes, but if she does not take this course, she may be confused☺

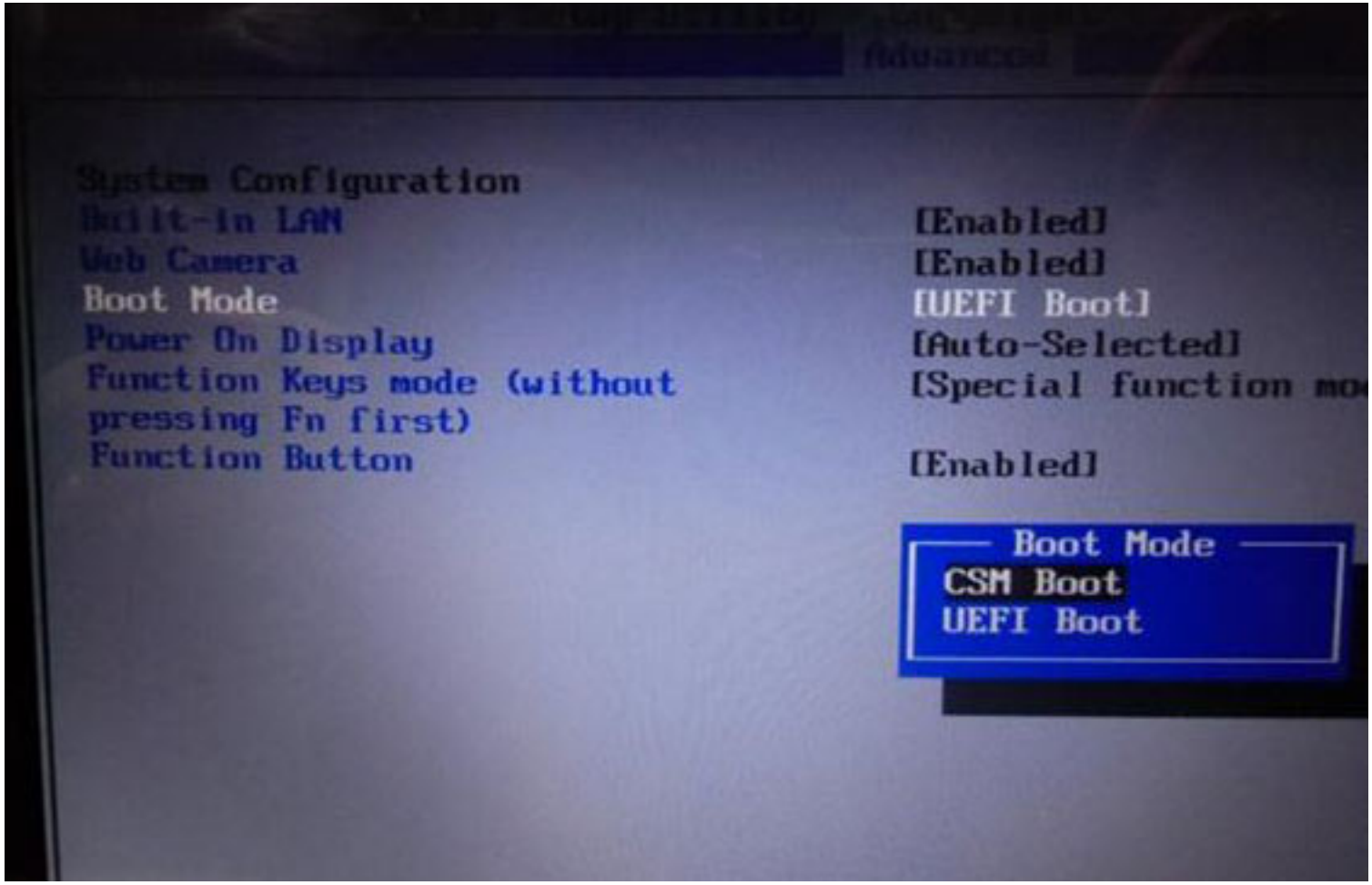
Nowadays all newly produced computers support UEFI. As we learned from the last class, when booting from a removable media (e.g., a USB drive) in the UEFI mode, UEFI first looks for an ESP partition and then the \EFI\boot\boot*.efi file to boot

Since the bootable USB drive was partitioned on a non-UEFI computer, it is impossible that it contains an ESP partition. So by default the firmware of the new computer will skip this device and try the next device

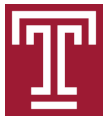
She should switch the firmware to the CSM mode (in addition to setting USB as the first to be tried in the **boot order** and turning off "**secure boot**")



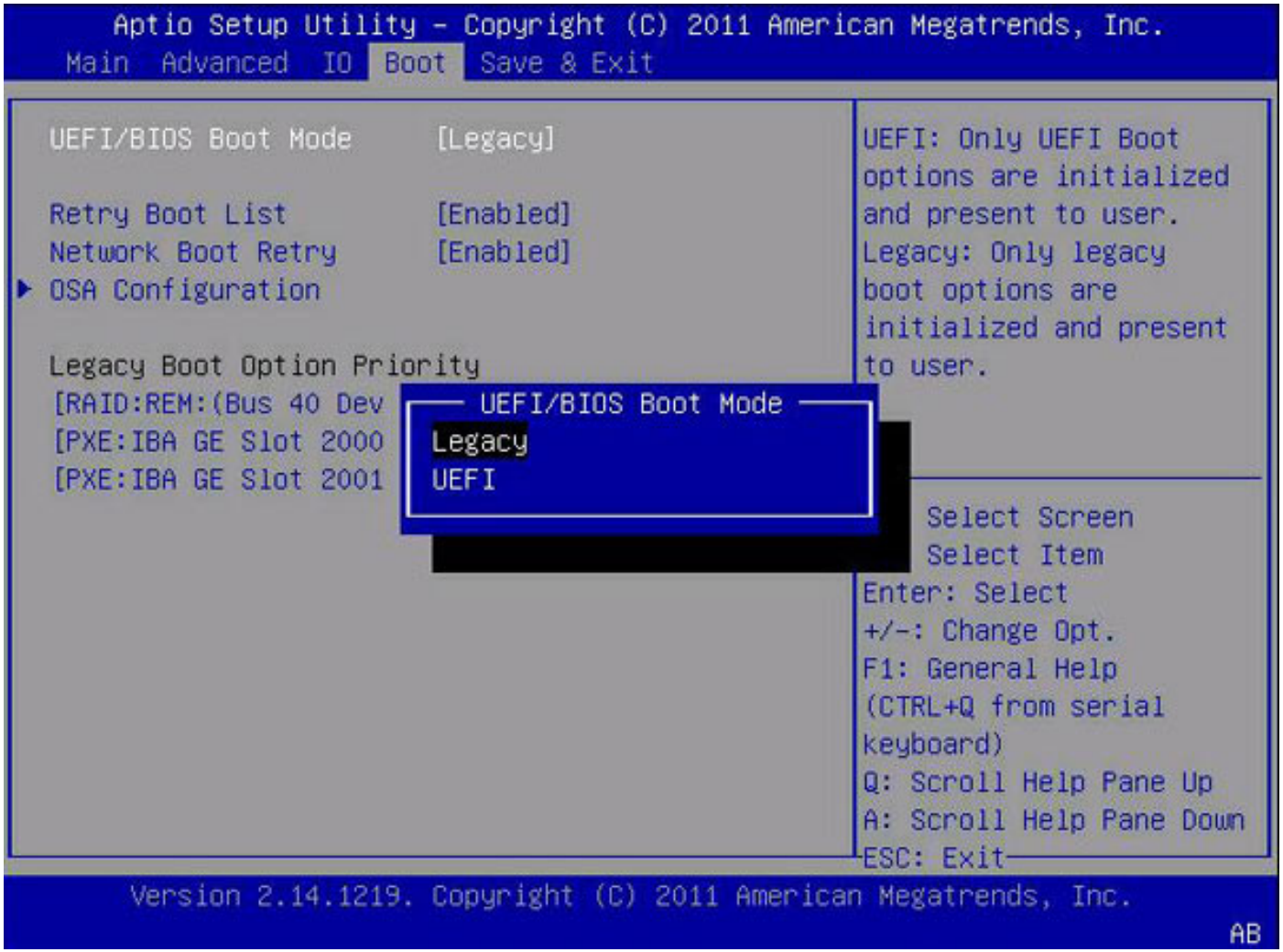
Selecting Boot Mode

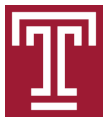


The interface and terms differ among vendors. E.g., CSM may be called legacy BIOS



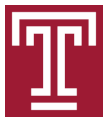
Selecting Boot Mode – Another Example





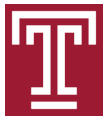
One More Thing about GPT: BIOS-GPT

- It is even possible to boot some systems (e.g., Linux) on a GPT-partitioned disk from a non-UEFI machine. It is called BIOS-GPT
- To support BIOS-GPT, Linux stores GPT-aware Stage 1 code on the MBR sector (note that GPT partitioning does not use the MBR sector), which locates a BIOS boot partition (~1MB) dedicated to storing GRUB's Stage 2 boot loader code
  - The post-MBR gap used to store GRUB's Stage 2 code in MBR disks is now occupied by the GPT partition table
- Officially, Windows does not BIOS-GPT



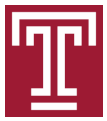
System Boot Types

	BIOS	UEFI
MBR	Firmware passes the control by jumping to the code stored in the MBR MBR can only support one vendor's boot loader	(the same as BIOS-MBR) You need to configure your firmware setting to CSM boot
GPT	Firmware passes the control by jumping to the code stored in the MBR, which is GPT-aware The control then goes to the Stage 2 code stored in a dedicated BIOS boot partition that stores the Stage 2's code	GPT allocates a dedicated ESP (EFI System Partition) formatted as FAT, and Firmware has the FAT file system driver. Thus firmware can locate all the boot loaders stored in ESP. Each OS vendor has a separate directory in ESP

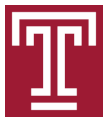


Outline

- Attacks against Boot
  - Bootkit
  - Evil Maid Attack
  - Bios-kit
- Attacks against RAM
  - DMA Attack
  - Cold Boot Attack

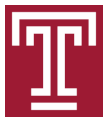


Boot Sequence



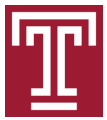
Bootkits

- The bootkit is a type of malware that infects boot loaders (such as MBR and VBR). It allows malicious code to take control even before the operating system
- A bootkit can be used to steal user passwords or to further infect the other components of the system



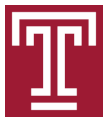
Bootkits

- After hardware initialization, the firmware passes the control to the bootkit
- It may subvert all security measures (e.g., access control) employed by the OS even before they are activated
- It can also intercept all API calls that may reveal its existence. So, theoretically, you cannot rely on an OS booted from an infected boot loader to detect the bootkit



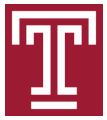
Case Study: TDL-4

- Spread through websites (e.g., drive-by download)
- Infect the MBR of your disk once obtaining the raw disk access privilege
- It creates its own private file system for storing other malicious code
- It turns off code integrity checks by instructing winload.exe to load the kernel in pre-installation mode, in order to replace “kdcom.dll” with a malicious one
- It then bypasses Windows’ code signing policy for drivers in order to install a malicious driver



Case Study: TDL-4

- The bootkit controls the IO of the hard drive:
  - When an application reads the boot sector, the bootkit counterfeits data and returns the original contents of the sector (i.e. as prior to infection), and it also protects the sector from overwriting
  - On any attempt to read sectors of the disk where the hidden file system is located, the bootkit returns a zeroed buffer as well as protecting the area from overwriting
- More details: https://www.welivesecurity.com/media_files/white-papers/The_Evolution_of_TDL.pdf



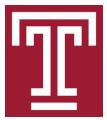
Take-away messages:

(1) A bootkit can subvert all the security measures employed by the OS and hide itself well

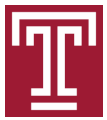
(2) Theoretically, you cannot rely on the OS (all anti-malware programs launched by the OS) to defeat or detect a boot kit, because a bootkit takes the control even before the OS boots

Outline

- Bootkit
- Evil Maid Attack
- Bios-kit
- DMA Attack
- Cold Boot Attack

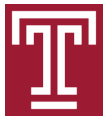


Evil Maid Attack



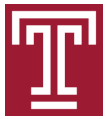
Evil Maid Attack

- The TDL-4 bootkit cannot be installed until some remote attack already gets very high privilege (e.g., raw disk access)
- The Evil Maid Attack does not rely on such privilege obtained, but it relies on physical access to the victim's computer
  - Many other famous attacks also rely on physical access, e.g., "cold-boot attack", "DMA attack"
- "You leave your laptop (can be even fully powered down) in a hotel room and go down for a breakfast... Meanwhile an Evil Maid enters your room." – Rutkowska, 2009



Evil Maid Attack

- The goal of is to obtain the password used to decrypt the disk (assuming the disk has been encrypted using TrueCrypt)
  - Since the disk is encrypted, it seems that you cannot get the user data even if you steal the disk. Is that real?
- Step 1: Attacker gains physical access to your shut-down computer and boots it from a malicious USB drive. Then, a hacked boot loader is written onto your system
  - If the computer is protected by BIOS password, the attacker can either reset the BIOS password or insert the disk to the attacker's computer
- Step 2: The victim later boots the computer using the malicious boot loader, which logs the password and sends it over the Internet (or does whatever the attacker wants)
- More details: https://www.schneier.com/blog/archives/2009/10/evil_maid_attac.html

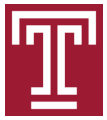


Question

Is two-factor authentication useful against the Evil Maid attack?

While a two-factor authentication or one time passwords are generally a good idea (e.g. they can prevent various keylogger attacks), they alone do not provide protection from Evil Maid-like attacks, because the attacker might modify his or her sniffer to look for the final decryption key (that would be calculated after the 2-factor authentication completes).

More details: "Evil Maid goes after TrueCrypt!" and "Anti Evil Maid"



Take-away messages:

Both Bootkits (e.g., TDL-4) and the Evil Maid attack take advantage of malicious system boot, which alerts us that the boot loaders should be well protected

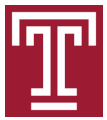
But who is to protect boot loaders?

OS?
Malicious boot is executed before OS, so certainly no!

BIOS?
Hmmm…

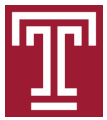
Outline

- Bootkit
- Evil Maid Attack
- Bios-kit
- DMA Attack
- Cold Boot Attack



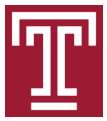
BIOS-kit

- BIOS-kit: a type of malware that infects your computer's firmware
- More generally and precisely, it should be called Firmware Rootkit
- Since firmware is the first piece of code executed after power-on, it can be exploited to fully control your system



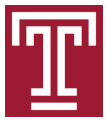
Case Study: Mebromi

- 2011
- The first BIOS rootkit in the wild
  - since the first PoC (proof-of-concept) BIOS-kit attack, IceLoard, in 2007
- It encapsulates a BIOS rootkit, an MBR bootkit, a kernel mode root kit, a PE file injector and a Trojan downloader
- It cheats victims by claiming itself as a plug-in for a well-known game and convinces users to turn off antivirus software to install the plug-in



Case Study: Mebromi

- After it infects the BIOS using a file called Cbrom.exe (which is a legitimate tool developed by Phoenix Technologies designed to modify the Award/Phoenix system's ROM binaries), it moves to infecting the MBR of the device and beyond
- Infection path: BIOS-MBR-Windows (BMW)
- It is almost impossible to be removed without a full reformat and BIOS flash at the same time
- More details: http://cleanbytes.net/bios-mbr-windows-bmw-or-mebromi-a-new-virus-targeting-the-computer-bios



Take-away messages:

To make use of firmware to verify the integrity of boot loaders on disks, you need first protect the integrity of firmware itself

There is a chain of trust during the system boot:
CPU -> firmware -> boot loaders ->
OS kernels -> drivers and apps

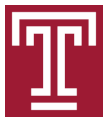
To make sure there is no malicious code in your drivers and apps, you need ensure the security in each step of the chain

Mission Impossible?

No, it is possible!

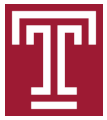
Outline

- Bootkit
- Evil Maid Attack
- Bios-kit
- DMA Attack
- Cold Boot Attack



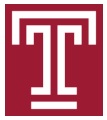
DMA

- DMA: Direct Memory Access. DMA lets devices (such as storage devices, network card, graphic card) transfer data between the devices and RAM at a high speed
- Ports that allows DMA:
  - FireWire
  - Thunderbolt
  - ExpressCard
  - PCI
  - PCI Express



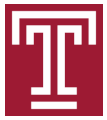
DMA Attacks

- The security concern is that it allows an external DMA device to bypass the OS's security measures (including the lock screen) to access RAM directly
- What kind of attacks can be launched?
  - Read data (to steal secret including keys in memory)
  - Modify data to disable security measures
  - Install malware
- Attacking tool: Inception
  - https://github.com/carmaa/inception
- Demo:
  - https://www.youtube.com/watch?v=HDhpy7RpUjM



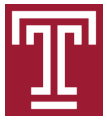
Mitigation of DMA Attacks

- Lock your computers; never leave them unattended (Physical Security)
- Disable unneeded ports through BIOS setting
- Disable DMA Attack when the screen is locked
  - E.g., [OS X > 10.7.2] and [Windows > 8.1] disable DMA when the screen is locked
- Use IOMMU (Intel's VT-d or AMD's AMD-Vi) to block unauthorized I/O transactions
  - E.g., OS X Mavericks > 10.8.2 on Ivy Bridge (>= 2012 Macs) have enabled VT-D



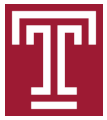
Outline

- Bootkit
- Evil Maid Attack
- Bios-kit
- DMA Attack
- Cold Boot Attack



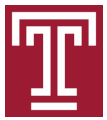
Cold Boot Attacks

- An attacker with physical access to a victim computer can use this attack to retrieve data from the running victim machine. There are two versions of Cold Boot Attacks
  - Reboot your computer and boot a system from a malicious USB
  - Plug out the RAM modules from the victim computer and plug them into the attacker's computer
- The second is more practical, as the first version may require BIOS setting change



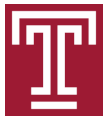
Cold Boot Attack

- It makes use of the data remanence of RAM
  - Content on RAM remains readable in the seconds to minutes after power off
  - It can be extended by cooling the RAM modules with a can of compressed air



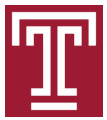
Cold Boot Attack

- The memory modules are removed from the original system and quickly placed in a compatible machine under the attacker's control, which is then booted to access the memory.
- Further analysis can then be performed against the information that was dumped from memory to find various sensitive data, such as the keys contained in it.
- Demo: https://youtu.be/JDaicPIgn9U



Mitigation

- Soldered memory modules
- Full memory encryption
- Make sure your computer is well shutdown when you leave your computer
  - If it simply goes sleep, Cold Boot Attack can be launched



Summary

- Attacks against System Boot
  - Bootkit, Evil Maid Attack
  - Bios-kit
- Attacks against RAM
  - DMA Attack
  - Cold Boot Attack
- Attacks that require physical access
  - Evil Maid Attack
  - DMA Attack
  - Cold Boot Attack