Virus Infection Techniques: Boot Record Viruses

Bill Harrison
CS4440/7440 Malware Analysis and Defense

Reading

- Start reading Chapter 4 of Szor

Virus Infection Techniques

- We will survey common locations of virus infections:
  - MBR (Master Boot Record)
  - Boot sector
  - Executable files (*.EXE, *.COM, *.BAT, etc.)
- Most of the examples of these viruses, especially the first two types, are from the DOS and floppy disk era

Why Study Older Viruses?

- Vulnerabilities remain very similar over time, along with the means to exploit them and defend against them
- Modern Internet worms differ mainly in the use of the internet for transport, and are otherwise similar to older viruses
- Older viruses illustrate the virus vs. antivirus battle over many generations

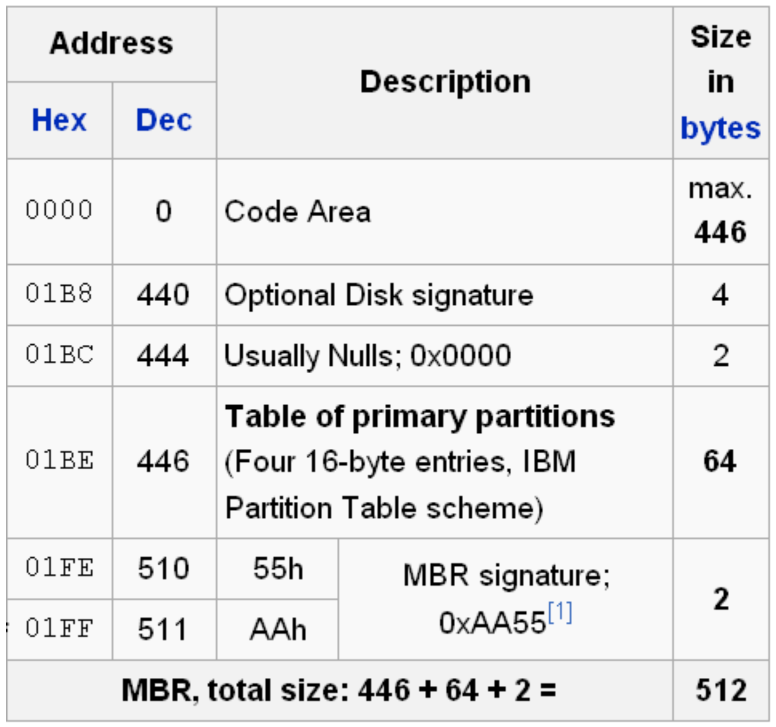
Boot-up Infections and the PC Boot-up Sequence

- PC boot-up sequence:

1. BIOS searches for boot device (might be a diskette, hard disk, or CD-ROM)
2. MBR (Master Boot Record) is read into memory from the beginning of the first disk partition; execution proceeds from memory

Master Boot Record Structure

Structure of a Master Boot Record



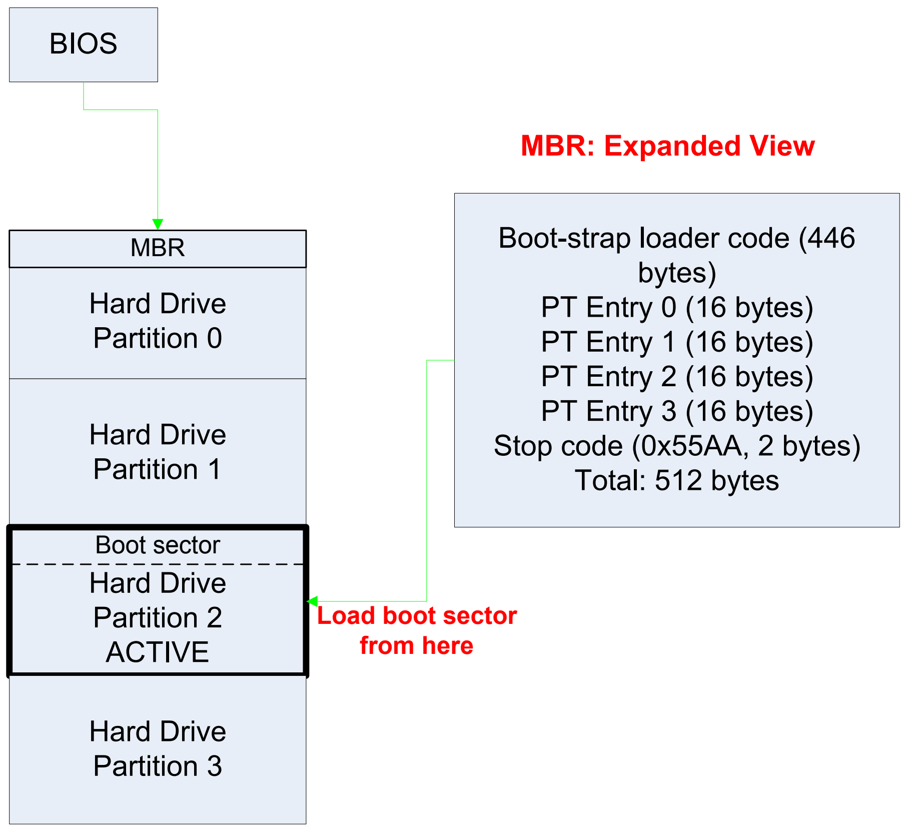
Address		Description		Size in bytes
Hex	Dec			
0000	0	Code Area		max. **446**
01B8	440	Optional Disk signature		4
01BC	444	Usually Nulls; 0x0000		2
01BE	446	**Table of primary partitions** (Four 16-byte entries, IBM Partition Table scheme)		**64**
01FE	510	55h	MBR signature; 0xAA55[1]	**2**
01FF	511	AAh		
MBR, total size: 446 + 64 + 2 =				**512**

Boot-up Sequence cont’d.

3. Beginning of MBR has tiny code called the *boot-strap loader*
4. Data area within MBR has the disk PT (partition table)
5. Boot-strap loader reads PT and finds the <u>active</u> boot partition
6. Boot-strap loader loads the first sector of the active partition into memory and jumps to it; this is called the *boot sector*

Boot-up Sequence cont'd.

- MBR is always at the very first sector of the hard disk (first 512 bytes)
- Boot sector is always the first sector of the active partition

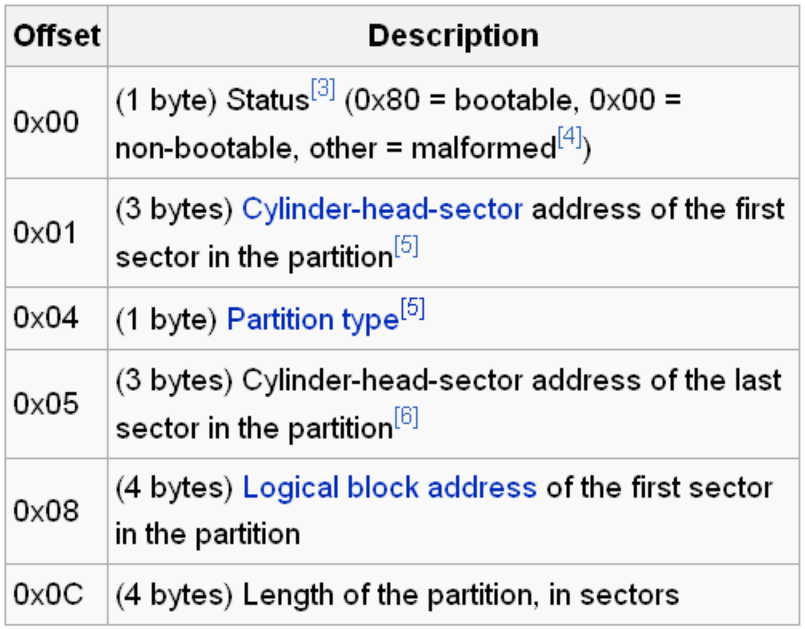


Partition Table

- PC hard disk has maximum 4 partitions
- Each of 4 PT entries tells:
  - All size parameters
  - File system type (FAT-16, FAT-32, NTFS, etc.)
  - Active (yes/no)
- Only one partition is marked active
- Boot-strap loader searches PT for the active partition; its first sector is the boot sector

Partition Table Entry

Layout of one 16-byte partition record



Offset	Description
0x00	(1 byte) Status[3] (0x80 = bootable, 0x00 = non-bootable, other = malformed[4])
0x01	(3 bytes) Cylinder-head-sector address of the first sector in the partition[5]
0x04	(1 byte) Partition type[5]
0x05	(3 bytes) Cylinder-head-sector address of the last sector in the partition[6]
0x08	(4 bytes) Logical block address of the first sector in the partition
0x0C	(4 bytes) Length of the partition, in sectors

MBR Infection Techniques

1. Save the bootstrap loader code elsewhere on disk, replace it with virus startup code, leave PT entries alone
2. Overwrite bootstrap loader code without saving it, leave PT entries alone
3. Change only the PT entries (to point to virus code), leave the bootstrap loader alone
4. Save entire MBR (loader and PT) to end of disk, replace with virus version

Saving Bootstrap Loader: *Stoned* Virus

- Stoned virus appeared early in 1988
- Created by a New Zealand college student
- Tried to be non-destructive
- Every 8th time an infected PC booted, it displayed the message: "Your PC has been Stoned! Legalize Marijuana!"
- Infected boot floppies with a variety of techniques that are useful to understand

Stoned Virus Techniques

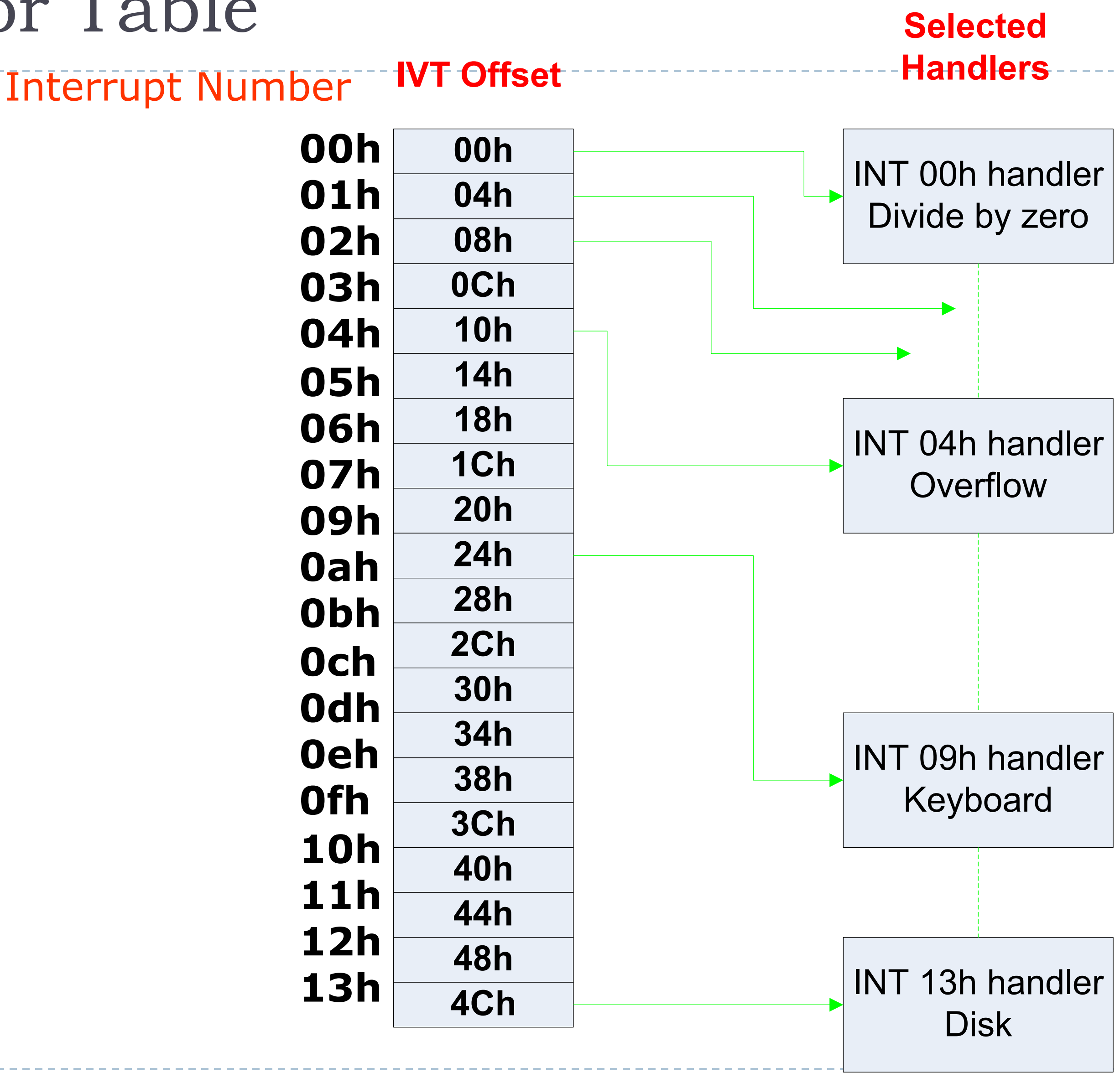
- Started out on a 360KB, 5 - 1/4” floppy disk
- Boot-strap loader was replaced with the virus loader, after saving the original loader at the end of the floppy disk
- The virus loader *intercepted* the disk access interrupt of the operating system (DOS)
- Used its own disk access interrupt handler to infect new floppy disks as they were accessed by the system

Stoned Virus Techniques: Interrupt Handler Interception

- After boot-up, the DOS loader (running from the boot sector) loads an interrupt vector table (IVT) at address 0 in physical memory
- Each 32-bit entry in the IVT is a pointer to the interrupt handling code for the corresponding interrupt
- E.g., at address 24h (hexadecimal) is the pointer to the code called when INT 09h is executed. This is the keyboard interrupt. INT 13h is the disk interrupt.

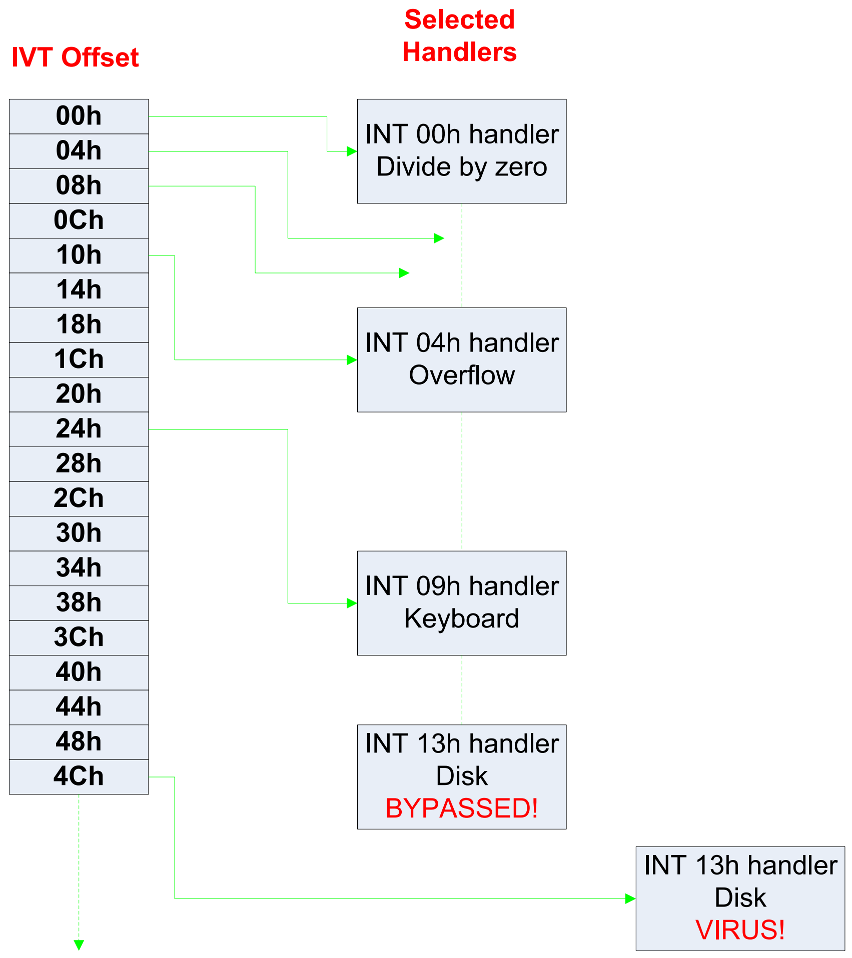
Interrupt Vector Table

- Before infection, interrupt handlers are located in memory in order of their interrupt number:



Infected Interrupt Vector Table

- After *Stoned* infection, one IVT entry points to a handler provided by the virus elsewhere in memory:



Stoned Disk Interrupt Handler

- Pseudocode:

```
if (disk operation other than read or write)
  then jump to saved system handler
elseif (any error condition on floppy drive)
  then jump to saved system handler
else  /* floppy ready for read/write access */
  call Infect()
  jump to saved system handler
endif
```

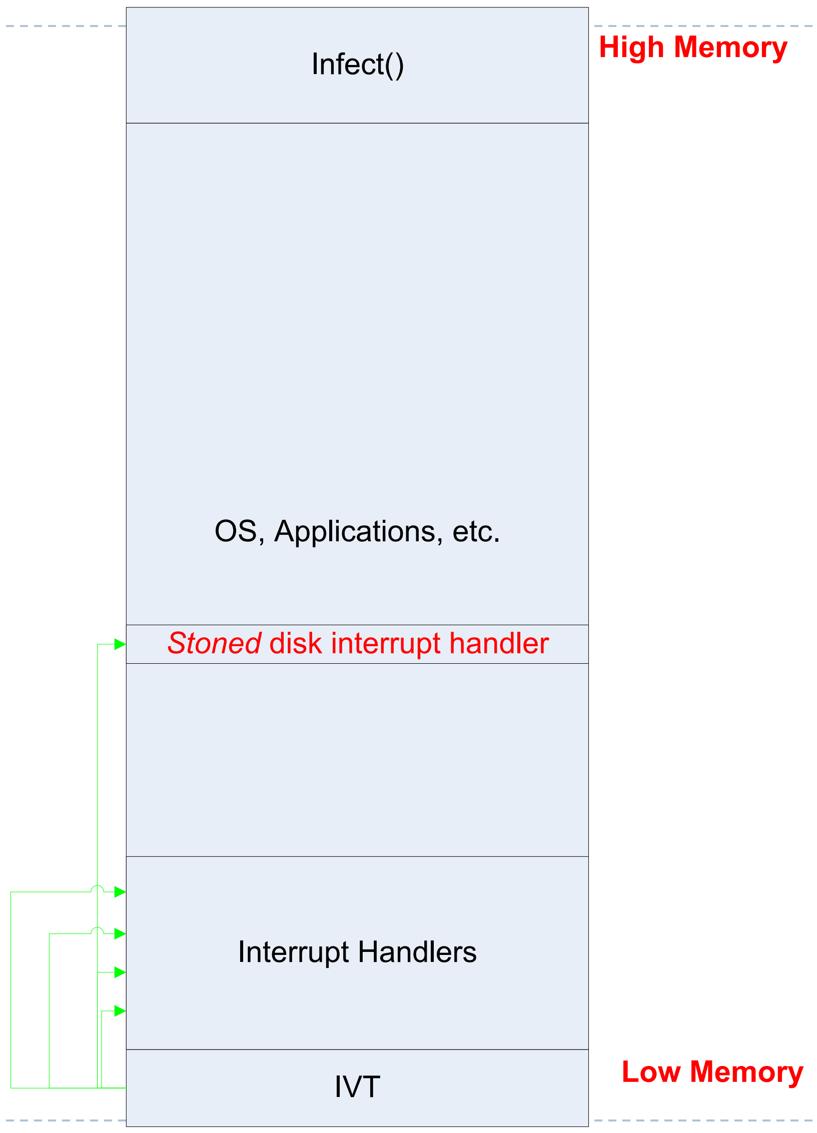
- Uninfected system handler does all the legitimate disk access work; virus handler just determines if an infection might be possible

Stoned: The Infect() Function

- The Infect() function was installed into the highest memory address on a DOS system.
- The available memory for the system is reduced by 2KB at each bootup so that the system would not overwrite the virus memory.
- Infect() first tries to infect the active floppy disk by moving its boot-strap loader and replacing it with the virus loader, and putting the virus at the end of the root partition of the floppy
- Then, if the infection counter is now 8, it displays its message and resets the counter, else it increments the counter and stays silent. Thus, a message is only displayed every 8th bootup.

DOS Memory after *Stoned* Infection

- The Infect() function and the modified IVT are easily visible in this memory diagram:



Stoned: Stealth Features

- The design incorporates *stealth* (i.e. attempts to evade detection) in several ways:

1. It only tries to infect a new floppy when a disk read or write has been called for. User is expecting the noisy activity of the floppy drive at this point, but would be suspicious at other times.
2. The Infect() function propagates the virus to the new floppy, but only displays the payload message, "Your Computer has been Stoned!", every eighth time it infects a floppy. This permits propagation to other users before the original user can react.
3. The designer <u>tried</u> to make it non-destructive, which makes it stealthier than a destructive virus.

Stoned: Accidentally Destructive

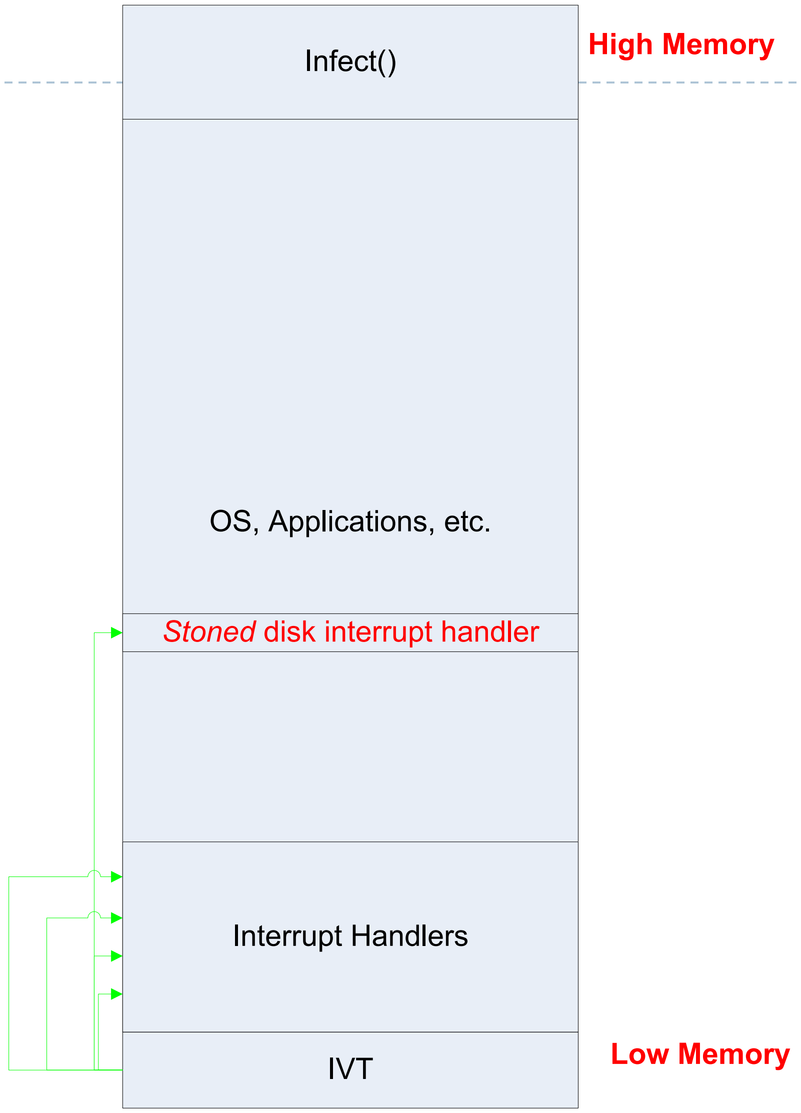
- The original boot-strap loader was saved to the end of the 360KB floppy, on the last sector, which was never used on that size floppy diskette.
- Unfortunately, after the virus spread, the 1.2MB 5-1/4" floppy was invented.
- The sector to which the loader was saved was now in the middle of the disk, and destructive overwriting was now frequent.

Stoned: Virus Maintenance

- The 1.2MB floppy drive invention was addressed by new virus writers, who updated the *Stoned* virus to work correctly with both kinds of floppy drives.
- By this time, it was common for PCs to boot first from the MBR on the hard disk, rather than from a floppy, so *Stoned* was updated again to infect the hard disk first and infect floppies from there as they were read and written by programs.

Stoned: Anti-Virus Detection

- How does anti-virus software detect a non-destructive virus with some stealthy features?
- Every virus changes something in the system; this change should be detectable
- Therefore, "undetectable viruses" are a myth
- Stoned has made several key changes to the system that should be detectable

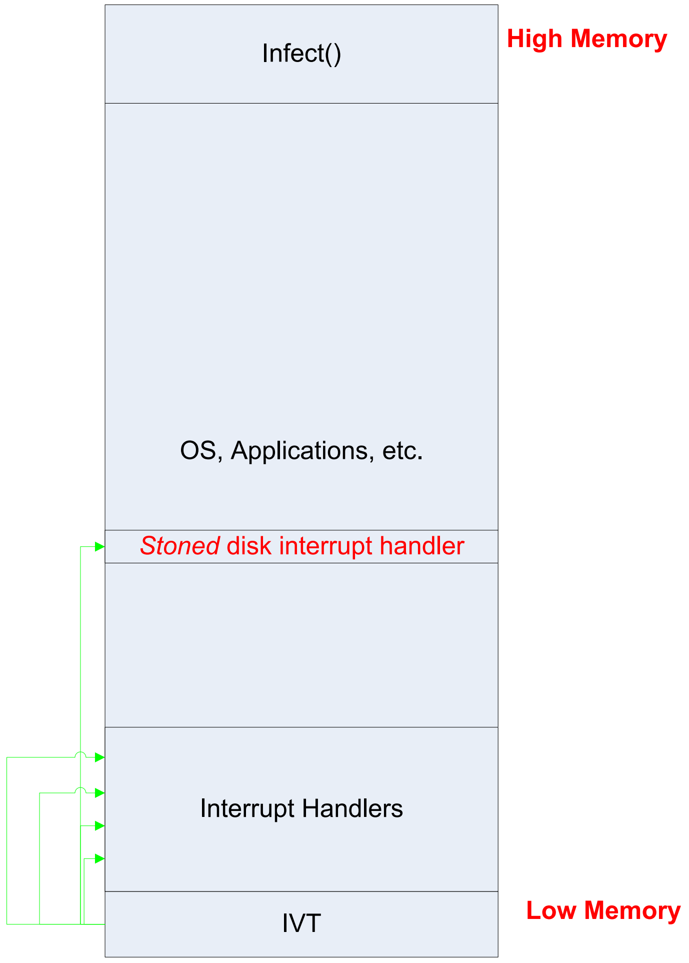

Infect()
High Memory
OS, Applications, etc.
Stoned disk interrupt handler
Interrupt Handlers
IVT
Low Memory

Stoned: Detectable Changes

1. The IVT entry for INT 13h has been "hooked", i.e. changed to point to the virus version of the disk interrupt handler.
    1. I.e., It is not pointing to a location between the INT 12h and INT 14h handlers any more.
2. The available memory has been decreased by 2 KB, so that there is a 2 KB dead spot at the top of memory.
3. The Infect() function can be found in this dead spot.

Stoned: Detectable Changes

- It is actually quite simple for anti-virus software to detect the first two changes:



Detecting a Specific Virus

- Sometimes, it is sufficient for disinfection purposes to detect the system changes, without knowing which particular virus is infecting the system
  - The IVT, PT and boot-strap loader can be repaired
  - The system memory limit can be restored
  - The dead spot in high memory, where the Infect() function resides, can be cleared out
  - The virus disk interrupt handler can be cleared from memory
- Anti-virus software often needs to know what virus infects the system, in order to know what damage to search for elsewhere

Detecting a Specific Virus

- The anti-virus software would need to analyze the code found in the dead spot in high memory in order to determine which virus has infected the system.
- *Pattern matching.* Virus pattern database for known viruses compared to portions of the code found in the virus memory area.
  - Research idea: are some representations better to match over than others; decompiling machine code to ???
- We will return to this topic in a week

Preventing a *Stoned* Infection

- A simple technique for avoiding *Stoned* infections is to write-protect all bootable floppies. Only bootable floppies have an MBR. There is usually no need to write to a boot floppy anyway.

Now that floppy disk viruses are no longer a hot topic, most users have reverted to their old bad habits and do not follow this advice. User behavior is always a prime area of vulnerability!

- 1993 freeware called PC Scavenger could replace your system MBR with a better MBR that prevents infection (by checking IVT and PT pointers, etc.)
- Modern anti-virus software has made the original floppy-disk-only version of *Stoned* extinct (but 90+ variants have arisen for hard disk MBRs!)

Lessons from *Stoned*

1. Boot-up time is a time of great vulnerability, because the boot-up software must have access to sensitive BIOS and hardware functions before the OS has even started
2. User behavior is another type of vulnerability that is often found in combination with system vulnerabilities.
3. A virus can be accidentally destructive, especially when hardware or OS assumptions are invalidated by changes occurring after virus creation.

Lessons from *Stoned*

4. Virus creators often design stealth into their viruses, so that they have time to replicate before they are detected
5. A virus writer community exists that maintains viruses and produces later variants of those viruses.
6. There is, therefore, an ongoing battle between the virus and anti-virus communities.

Lessons from *Stoned*

7. Viruses must make changes to the system, so they <u>must</u> be detectable.
8. The undetectable super-virus is a myth!

Announcement

- Midterm 1 on March 1 (Wednesday)

MBR Infection Techniques

1. Save the bootstrap loader code elsewhere on disk, replace it with virus startup code, leave PT entries alone
2. Overwrite bootstrap loader code without saving it, leave PT entries alone
3. Change only the PT entries (to point to virus code), leave the bootstrap loader alone
4. Save entire MBR (loader and PT) to end of disk, replace with virus version

Overwriting the Loader

- *Azusa* virus in January, 1991, was one of the first to use this <u>destructive</u> technique
- Anti-virus software was unable to repair infected systems
- Solution: Anti-virus software was updated to include a generic boot-strap loader that could be written over the virus loader, restoring the system to its initial MBR state

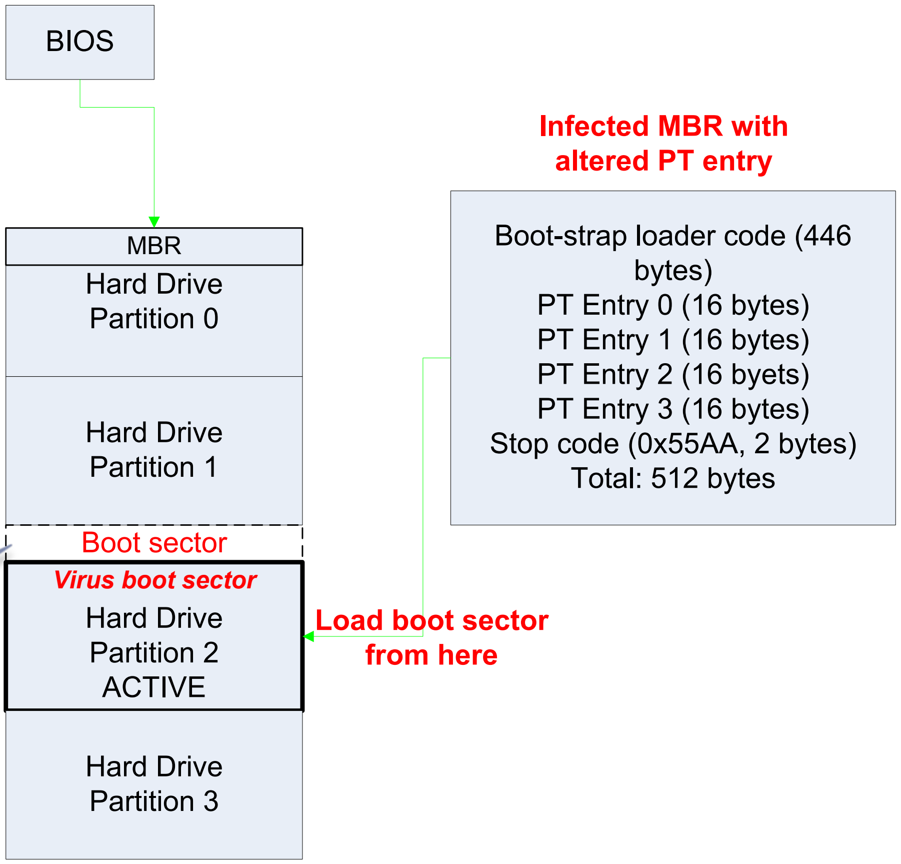
MBR Infection Techniques

1. Save the bootstrap loader code elsewhere on disk, replace it with virus startup code, leave PT entries alone
2. Overwrite bootstrap loader code without saving it, leave PT entries alone
3. Change only the PT entries (to point to virus code), leave the bootstrap loader alone
4. Save entire MBR (loader and PT) to end of disk, replace with virus version

Altering the Partition Table

- 1991 Russian virus *Starship* was among the first to make the partition table point to the virus rather than the boot sector, by changing only 3 bytes in the PT:

Original boot sector is now in an unreachable dead spot on the disk.



Detecting an Altered PT

- Anti-virus software can analyze boot sectors to detect virus operations within it (e.g. hooking an interrupt)
- The bypassed boot sector can be detected by noticing that the PT entries leave an unreachable region on the disk
- The unreachable region can be analyzed to confirm that it is a valid boot sector

MBR Infection Techniques

1. Save the bootstrap loader code elsewhere on disk, replace it with virus startup code, leave PT entries alone
2. Overwrite bootstrap loader code without saving it, leave PT entries alone
3. Change only the PT entries (to point to virus code), leave the bootstrap loader alone
4. Save entire MBR (loader and PT) to end of disk, replace with virus version

Saving the Entire MBR

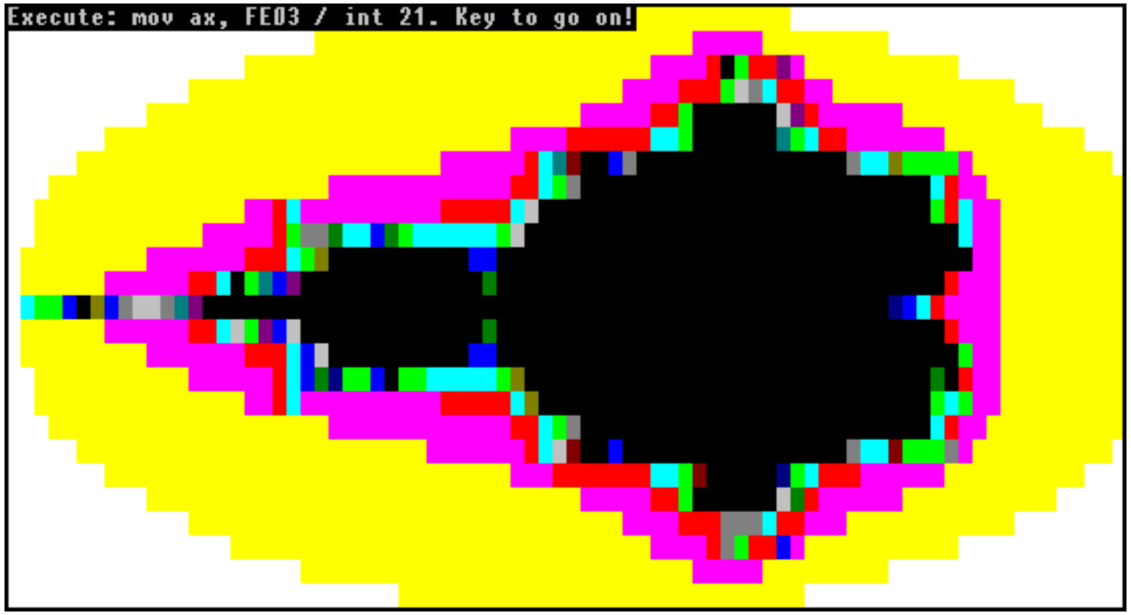
- *Tequila* virus is a 1991 example. It saved the entire MBR to the end of the hard disk, then altered the PT entries to make that area a dead spot that could not be overwritten.
- Also kept a copy of itself in the dead spot, safe from overwriting
- Seems to have saved the MBR to permit system repair (i.e. tried to be nondestructive)
- Hooked the disk interrupt, spread itself through floppies and *.EXE files

Tequila Stealth Techniques

- *Tequila* made many changes that should be visible to anti-virus software or even to system users:
  - *.EXE files got bigger because of infection, the files have been modified recently, etc.
- *Tequila* hooks the interrupt used to access file sizes.
  - When a DIR command runs, it reports the original file sizes, not the larger infected size, because the disk file size has been hooked
- *Tequila* keeps the file modified date from changing
- It lies dormant for four months after infection, so that users will not be able to identify which recent diskette must have carried the infection
- *Tequila* was also nicknamed *Stealth* at one time

Tequila Awakes After 4 Months

- After its four month dormant period, Tequila generates a crude, block-pixel random fractal picture
- It has had four months to spread, so it is no longer necessary to remain stealthy



```
Execute: mov ax, FE03 / int 21. Key to go on!
```

Detecting *Tequila*

- Despite its stealth, *Tequila* still leaves traces of its infection:

1. The total system memory has been reduced, because *Tequila*, along with *Stoned* and many other DOS viruses, allocated itself into a memory block
2. The PT entries have been modified
3. There is a dead spot at the end of the disk in which *Tequila* and the original MBR lie

Boot Sector Viruses

- A virus can modify the boot sector (first sector of active partition), leaving the MBR alone
- The advantage is that detection is not as easy as noticing a modified boot-strap loader or PT entries; no dead spots on disk, etc.
- Anti-virus code can still analyze the boot sector and detect code patterns that hook interrupts, etc.
- 1988 *Denzuko* virus is early example

  Also the first example of a competitive virus; it killed the *Brain* virus when it discovered they coexisted on a PC!

Competitive Virus Question

- What are the possible motivations for a virus writer to make a competitive virus?

Memory Resident Viruses

- We already saw that *Stoned* and *Tequila* allocated a block of memory for themselves, reduced system memory available to protect their area, and executed "in the background" while DOS continued to operate
- This permits the virus to monitor system activity, being active while hooked interrupts execute, detecting that a floppy is being written and infecting it, or detecting that a *.EXE file is being accessed from the hard disk and infecting it
- Can also detect anti-virus code execution and attack it! More on this later.
- By contrast, a **file-resident** virus only executes when you execute that file.

Cleaning Memory Resident Viruses

- Early anti-virus programs often repaired the MBR or boot sector and declared victory
- Memory-resident viruses were then designed to re-infect the MBR or boot sector after the anti-virus program finished
- Anti-virus code then evolved to scan memory, find the memory resident virus, zero it out, reallocate memory, and reboot
- Szor, Chapter 12, has the gory details of memory scanning and disinfection. Too much detail for this semester's class

Worth noting: Memory scanning and disinfection are not limited to the DOS era. These techniques are still required in current Windows anti-virus software.

Multi-partite Viruses

- Some viruses infect executable programs, so that whenever they are run, the infection can replicate
- Other viruses infect the MBR or boot sector, so that whenever the system boots, they become active
- A multi-partite virus, such as *Tequila*, does <u>both</u>.
  - It infects the MBR, hooks interrupts, makes itself memory resident, and also infects *.EXE files.
  - Infection can spread from the memory-resident virus OR from the *.EXE files to any floppy disks that are accessed by the system.

Disinfecting Multi-partite Viruses

- If anti-virus software repairs the MBR and cleans out the memory-resident virus, the system can still be infected when an infected *.EXE is invoked.
- Likewise, if all *.EXE files were scanned and cleaned but the MBR or memory-resident virus code were not discovered, the *.EXE files will get re-infected almost immediately.
- This is one example of why it is helpful for the anti-virus code to use patterns to detect exactly which virus is present, so that it knows all the system locations that are probably infected.