Kernel Malware

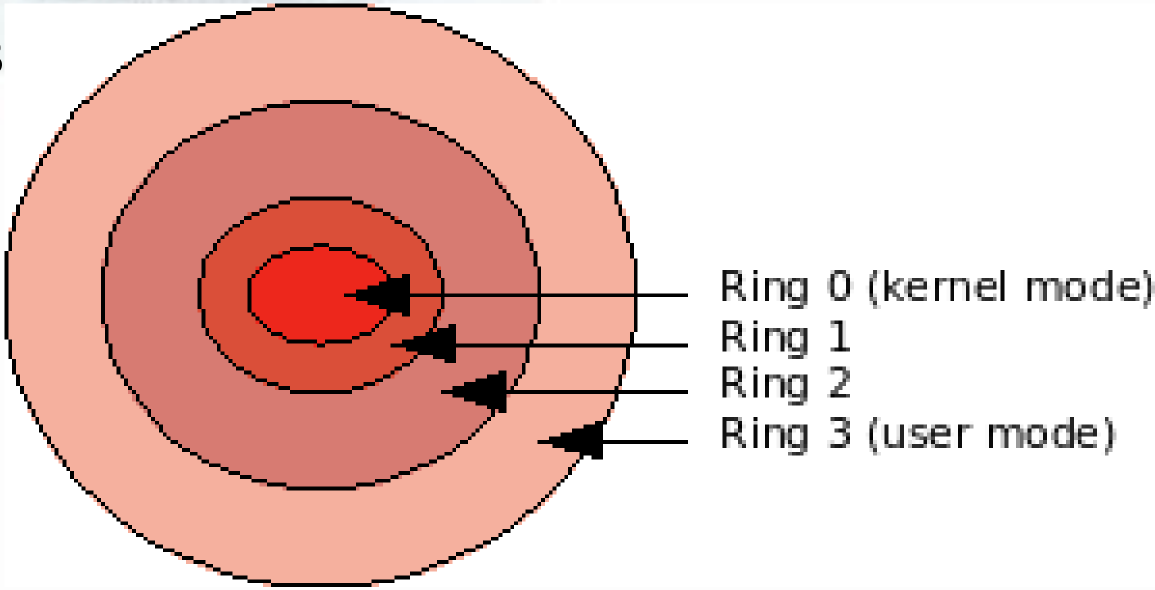
Zhuoqun Cheng

Feb 2013

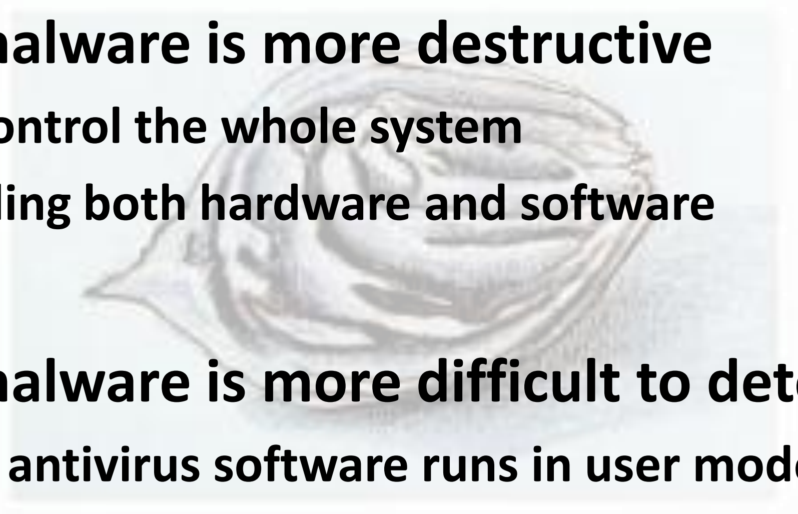
Kernel Mode vs. User Mode

- x86 provides 4 privilege levels
  - Ring 0 – kernel mode for kernel (highest)
  - Ring 1,2 - not used
  - Ring 3 - user mode for applications (lowest)

- Higher level can control lower levels and access more hardware resources

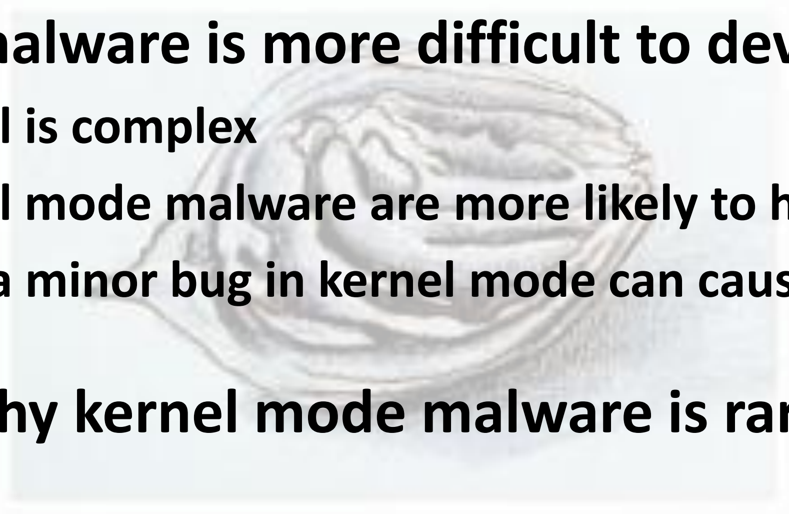


Kernel Malware vs. User Malware



- Kernel malware is more destructive
  - Can control the whole system
  - including both hardware and software
- Kernel malware is more difficult to detect or remove
  - Many antivirus software runs in user mode
  - lower privilege than malware
  - cannot scan or modify malware in kernel mode

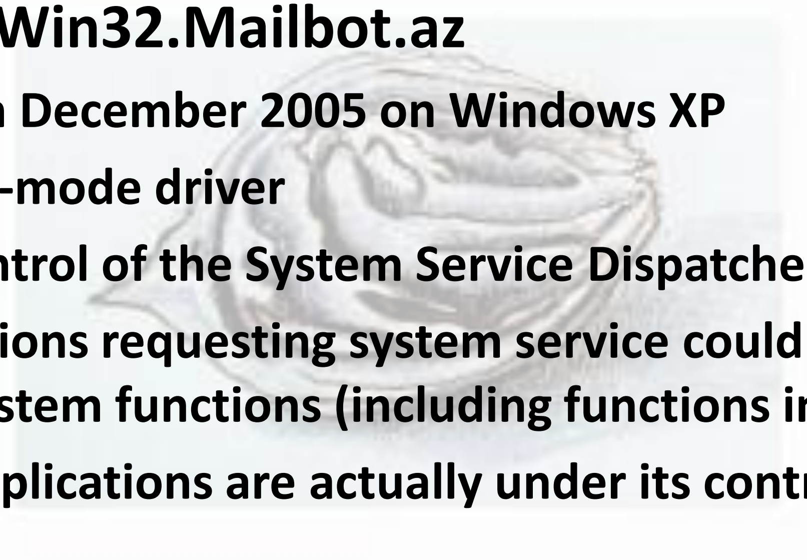
Kernel Malware vs. User Malware



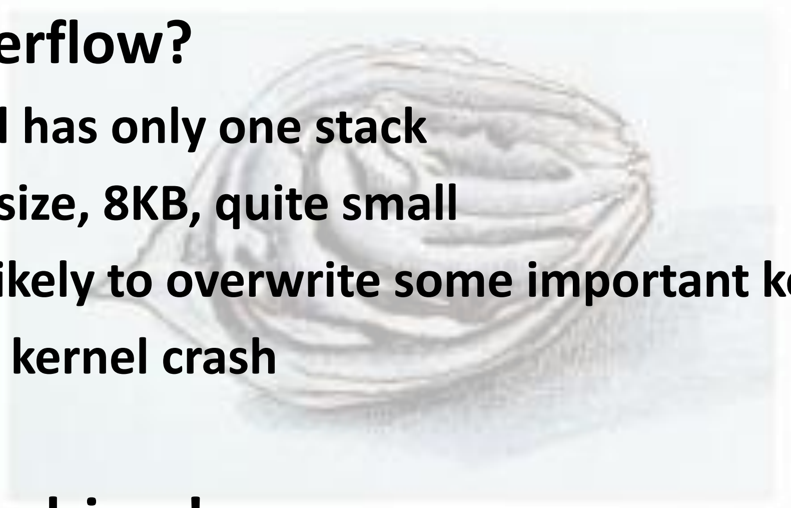
- Kernel malware is more difficult to develop
  - Kernel is complex
  - Kernel mode malware are more likely to have bugs
  - Even a minor bug in kernel mode can cause kernel crash
- That’s why kernel mode malware is rare

An Example

- SpamTool.Win32.Mailbot.az
  - Found in December 2005 on Windows XP
  - A kernel-mode driver
  - Took control of the System Service Dispatcher (SSD)
  - Applications requesting system service could be redirected to other system functions (including functions in malware)
  - So all applications are actually under its control



How to exploit kernel?



- Stack overflow?
  - Kernel has only one stack
  - Fixed size, 8KB, quite small
  - Very likely to overwrite some important kernel data
  - Cause kernel crash

- Loadable driver!
  - Drivers run in kernel mode
  - Windows allows drivers to be loaded at runtime
  - Develop malware as drivers and ask kernel to load it

Mitigation

- Drivers must be signed since Windows Vista
- Check before driver is loaded
- Unsigned driver cannot be loaded into kernel

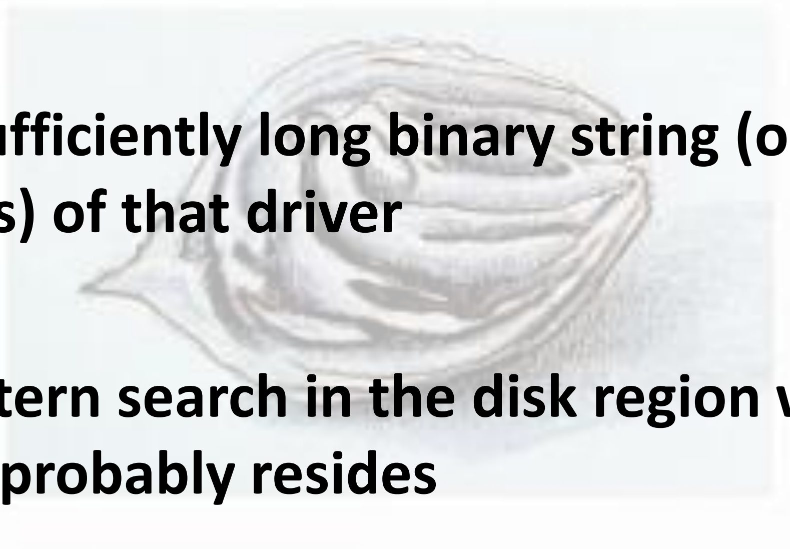
One possible bypass

- Loaded driver (signed and checked) will be swapped out from memory to *Pagefile* in disk when short of memory
- Modify *Pagefile* and insert our shellcode
- Call that driver
- Swapped in and get executed

First how to force the specific driver to be swapped out?

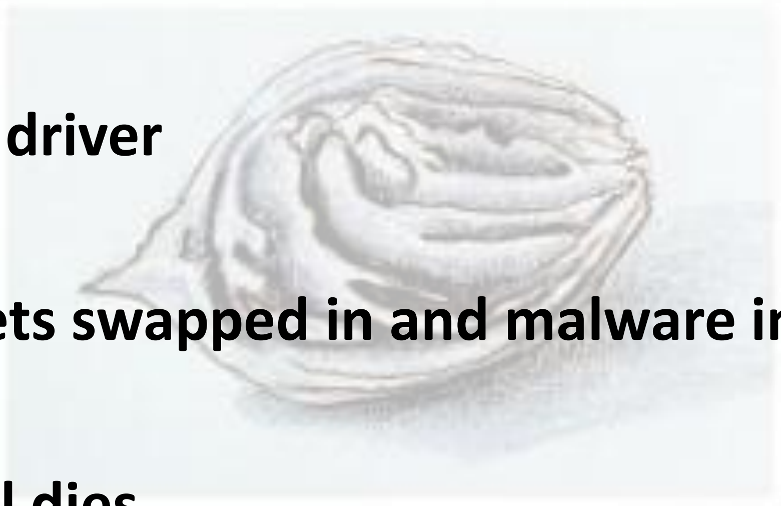
- Allocate huge amount of memory for a process to use up physical memory
- Some rarely used drivers are always swapped into disk

Second how to locate and modify that driver?

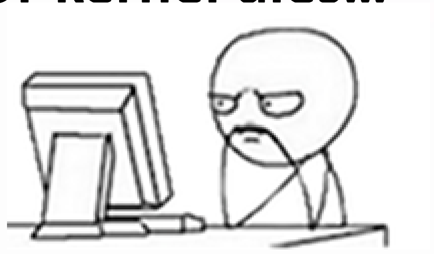


- Take a sufficiently long binary string (one of its functions) of that driver
- Do a pattern search in the disk region where *Pagefile* probably resides
- Replace it with our shellcode (extremely difficult to create useful shellcode)

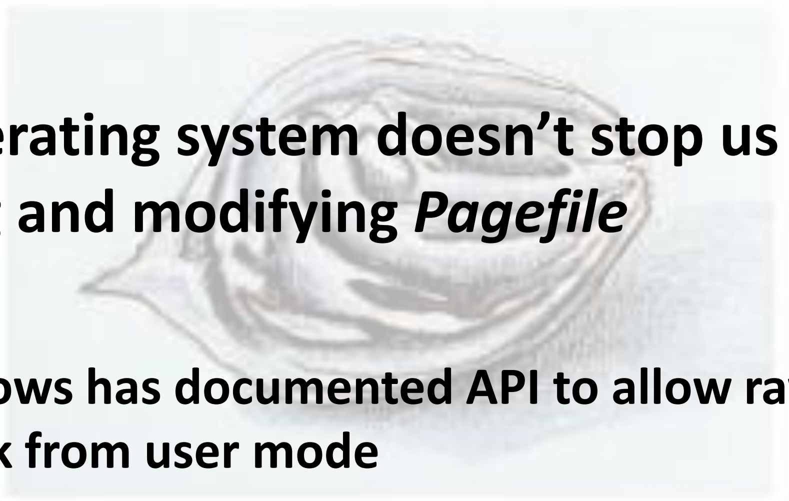
Final step



- Call that driver
- Driver gets swapped in and malware injected!
- Or kernel dies…

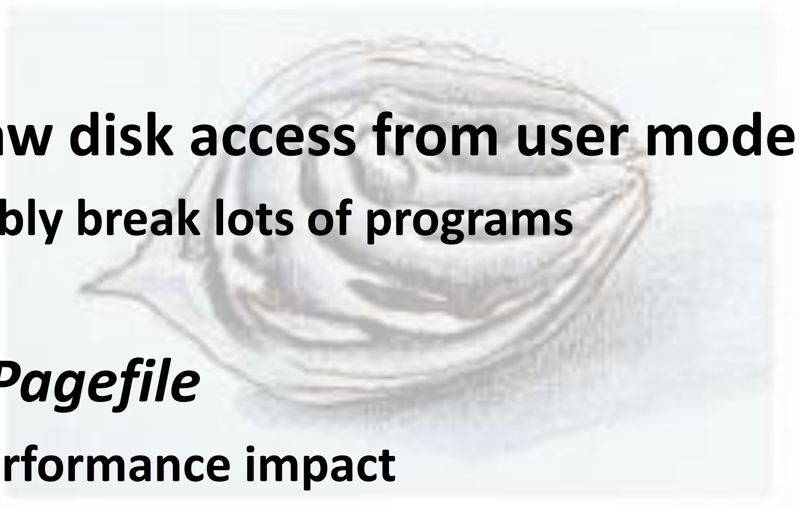


Wait…



- **Why operating system doesn't stop us from scanning and modifying *Pagefile***
    - **Windows has documented API to allow raw access to disk from user mode**
    - **We can read and write disk sectors which are occupied by the *Pagefile***
    - **While kernel has no idea what file we are modifying since we don't go through file system**

Possible mitigations



- Forbid raw disk access from user mode
  - probably break lots of programs
- Encrypt *Pagefile*
  - Big performance impact
- Disable kernel memory swapping
  - Possible. But users lose this useful feature

Thank you!
Questions?
Reference

- **Kernel Malware: The Attack from Within**
  - **Kimmo Kasslin, Kuala Lumpur**
- **Subverting Vista Kernel for Fun and Profit**
  - **Joanna Rutkowska**
- **Wiki: Rootkit**
  - **http://en.wikipedia.org/wiki/Rootkit**