Low Down and Dirty: Anti-forensic Rootkits

Presented by Darren Bilby
Ruxcon 2006

Agenda

- Anti-forensics Overview
- Digital Forensics Acquisition
- The Live Imaging Process
- How Live Forensics Tools Work
- DDefy Introduction
- NTFS Basics
- DDefy Disk Forensics Demonstration
- DDefy Challenges
- DDefy Memory Forensics Demonstration
- Better Methods for Live Imaging

This is Not…

- A demonstration of 0day rootkit techniques

This is …

- Showing flaws in current and proposed forensic techniques
- Showing how evidence could be manipulated and people wrongly convicted through bad forensic methodologies

Digital Anti-forensics

Anti-Forensics Methods

- **Data Contraception**
  - Prevent evidence data from existing somewhere that can be analyzed
  - E.g. Memory only malware, memory only exploitation

- **Data Hiding**
  - Put the data on disk but put it somewhere the forensic analyst is unlikely to look
  - E.g. Defilers toolkit, runefs,

Anti-Forensics Overview

- **Data Destruction**
  - Destroy any evidence before someone gets a chance to find it
  - E.g. Disk wiping, wipe, srm, evidence eliminator, necrofile

- **Data Misdirection**
  - Provide the forensic analyst false data that is indistinguishable from the real thing
  - No public examples… until now.

Digital Forensics Acquisition

- **Need to gather an evidential copy of a system**
- **The Aim**
  - Gather the "best" evidence available
- **Gather volatile information**
  - memory, process list, network connections, open files…
- **Power off machine and image disk**

Digital Forensics Acquisition

- **What really happens…**

- **Two Competing Aims**
  - Gather the "best" evidence available
  - Allow the system to continue operation in an unhindered manner

- **Results in "Live Imaging"**
  - Taking a copy of a system while that system is still functioning in a live environment

Reasons for “Live Imaging”

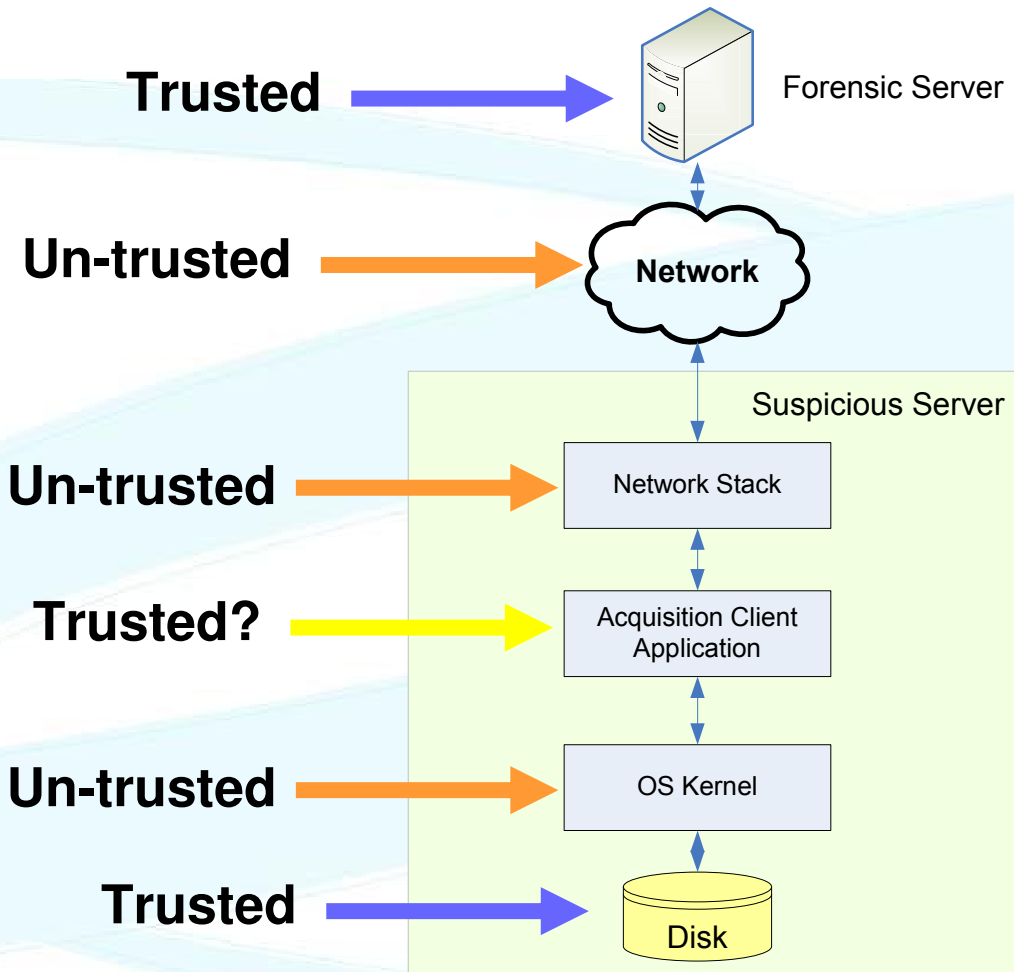
- **Business critical systems that cannot be shut down**
- **Shutting down systems may create legal liability for examiners through:**
  - damaging equipment
  - unintentional data loss
  - hampering operations
- **Judge instructs that evidence gathering must be conducted using the least intrusive methods available**
- **Encrypted volumes**

Digital Forensics Acquisition

Live imaging is now common practice

- **Tools**
  - Helix (dd/netcat)
  - Prodiscover IR
  - Encase EEE/FIM
  - FTK
  - Smart
  - …

The Live Imaging Process



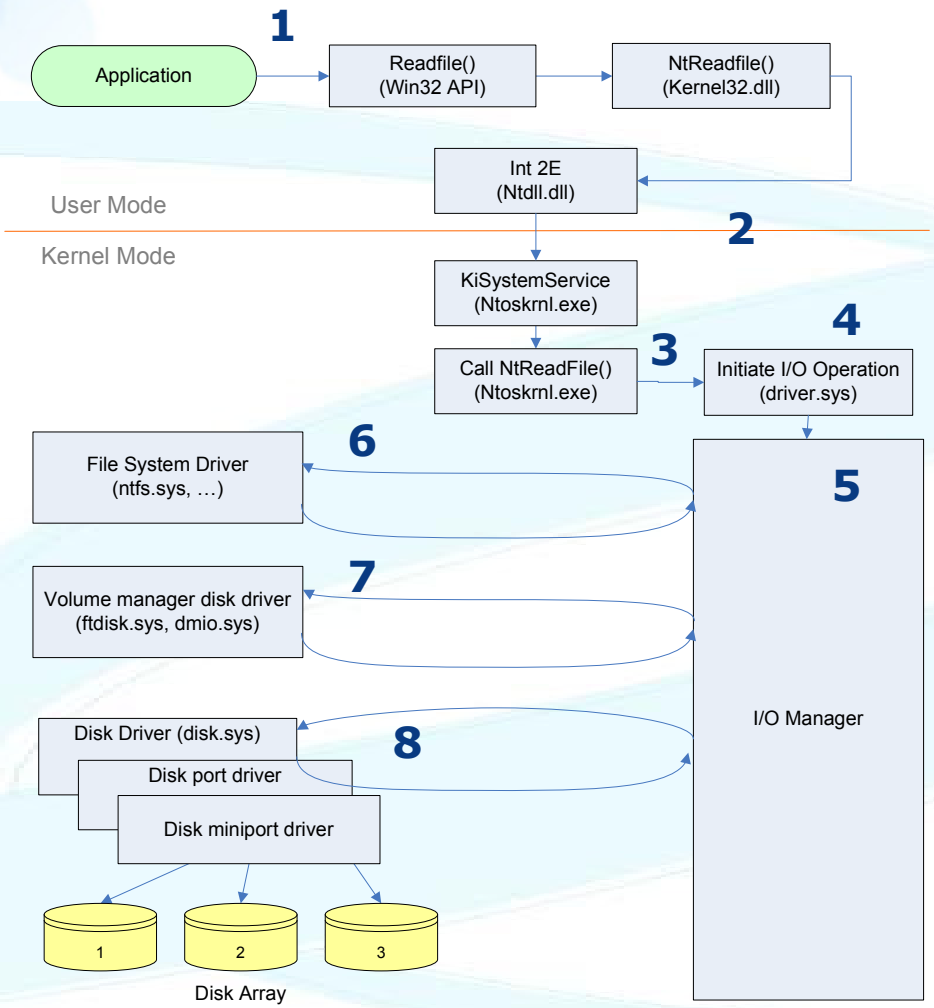
So this is common practice, accepted as legitimate by most courts of law, supported by many big name forensic vendors, it must be foolproof right?

uhhh... ok

Live imaging…

… is like turning up to a homicide at the docks and asking the mafia to collect your evidence and take it back to the police station for you.

What Happens When You Read a File?

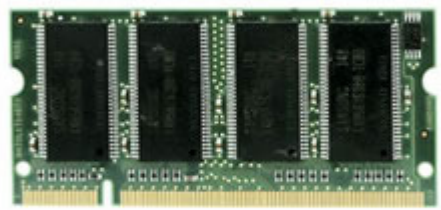


1. Readfile() called on File1.txt offset 0
2. Transition to Kernel mode
3. NtReadFile() processed
4. I/O Subsystem called
5. I/O Request Packet (IRP) generated
6. Data at File1.txt offset 0 requested from ntfs.sys
7. Data at D: offset 2138231 requested from dmio.sys
8. Data at disk 2 offset 139488571 requested from disk.sys

How do the Live Forensics Tools Work?

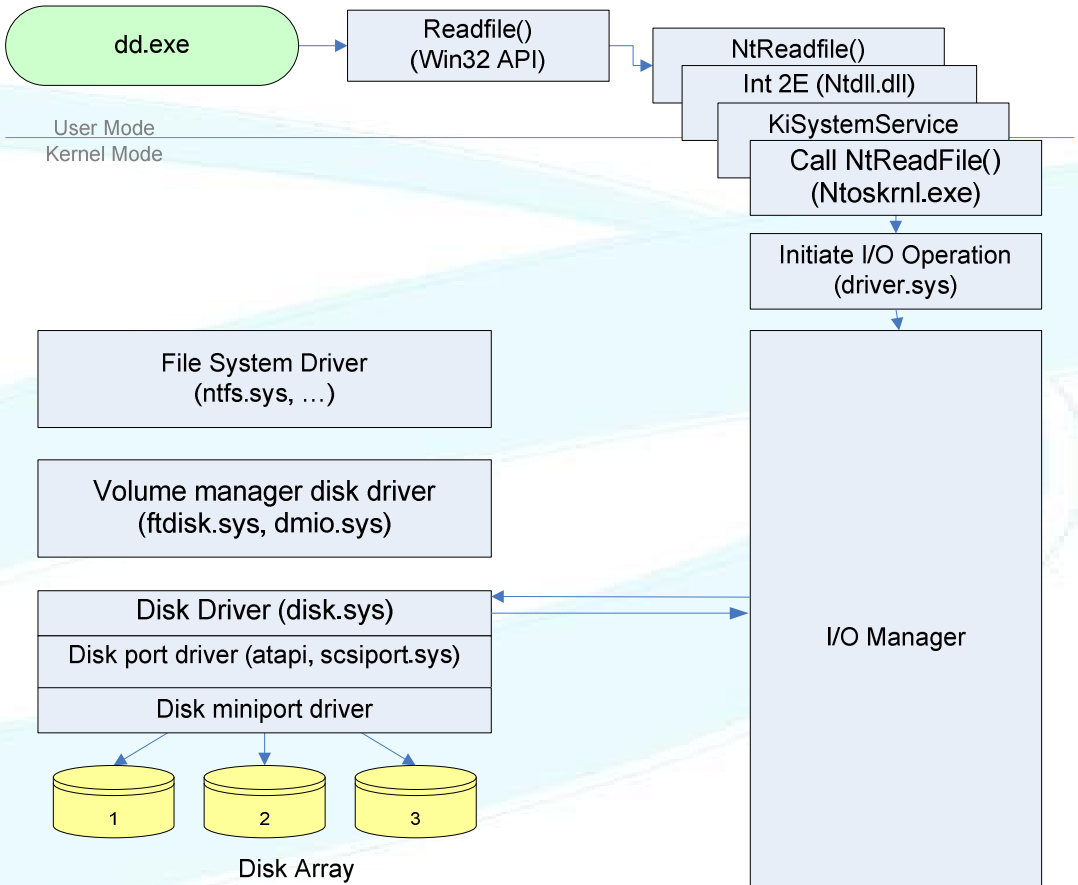
Read special device object files

- **\\.\PhysicalMemory**
  - Contains what is in physical memory
  - Only sees memory it can access safely
  - Changes as it is read



- **\\.\PhysicalDriveX**
  - The raw data on the disk
  - Ignores software cache
  - Could change as it is read

How do the Live Forensics Tools Work?



How do the Live Forensics Tools Work?

- **All tested live forensics tools utilize this method**
  - DD (GM Garner)
  - FTK Imager
  - Prodiscover IR

- **Example live imaging command**

  psexec \\target –u Administrator –p password dd.exe if=\\.\PhysicalDrive0 of=\\mymachine\share$\target.drive0.dd bs=8k conv=noerror

How do the Live Forensics Tools Work?

Aim: Get an accurate low level copy of disk and memory.

Good Things about Live Forensics Tools:

- Bypass standard rootkit techniques so a hacker can't hide their files
- Bypass major parts of the Windows driver stack including the File System Driver and Volume Manager Driver

DDefy Anti-forensic Rootkit

DDefy: The Idea

- Live forensics tools trust a bad kernel to give them clean data
- Lets create a rootkit that hooks below the forensic tools
- Provide the forensic tool a valid but "cleansed" version of the system
- Implement as low as we can get while still being practical
- Implement instead of claiming theory

DDefy: The Challenge

From Live forensics tool documentation:

Q. Can't a rootkit be written to avoid detection?

A. While it is **theoretically possible** to create a rootkit to alter the lower level disk sector read command, **it would be extremely difficult and would require significant information about the specific machine being rooted**. Any attempt to determine which sectors contain the data the rootkit is trying to hide would need to keep track of virtually the complete disk data structure on the system to keep the normal operation of the system from overwriting the files. If the cracker tried to mark a section of disk "bad" to prevent the system from altering the hidden files, it would no longer be available to read using the rooted disk sector commands. **We believe it is impractical to create this low level rootkit.**

DDefy: Implementation

- **The Aim: When someone does live forensics on my system they should get a valid image, but not my sensitive data.**

- **Written on the power of short blacks and jack daniels**

- **Proof of concept for 2K/XP/2k3**
- **Kernel mode Rootkit**

- **Upper Disk Filter Driver**
- **Intercepts IRP_MJ_READ I/O Request Packets sent to the disk and modifies the return data**

DDefy Disk Forensics Demo

Capture Demo

Analysis Demo

DDefy: Results

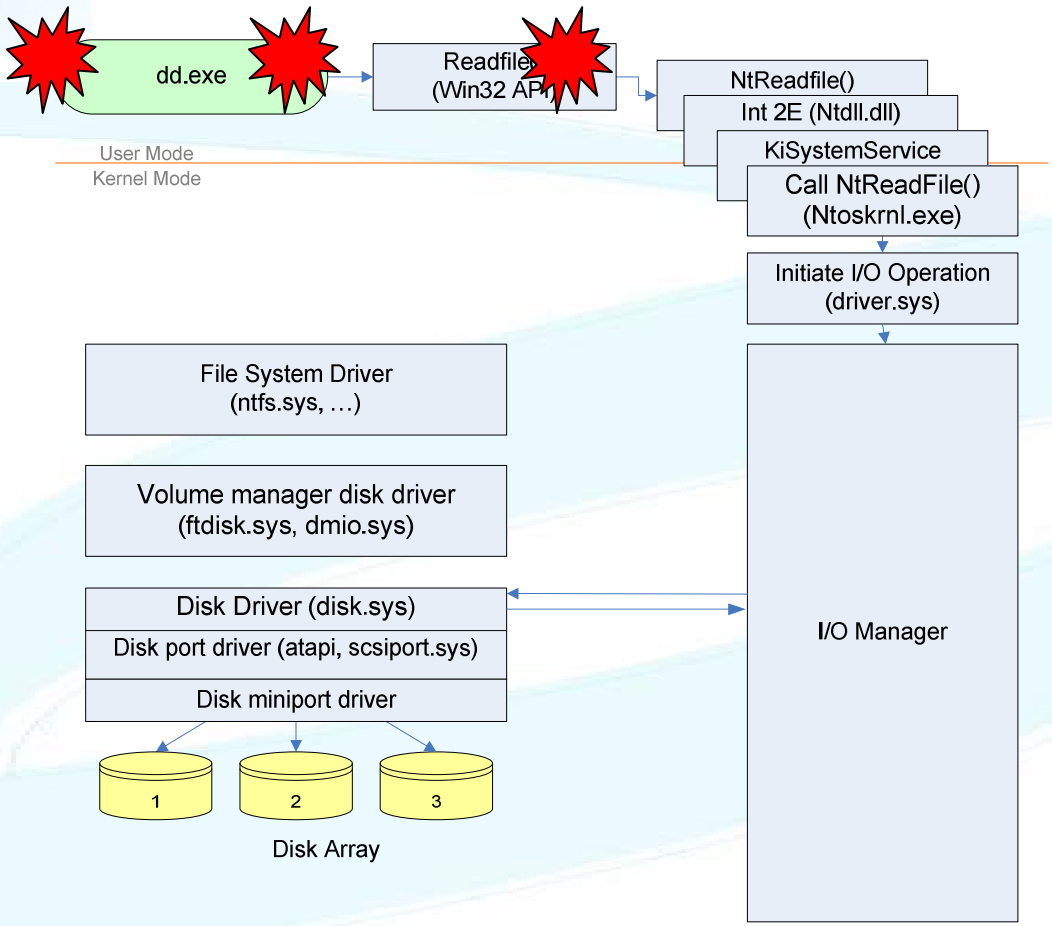
- **Any data that is stored on the physical disk can be hidden from a live forensics or detection tool**
- **There is no way to completely prevent this**
- **Live forensic imaging is still a useful tool**
  - but it needs to be used with full knowledge of the limitations
- **Image the disk offline whenever possible**

DDefy: Challenges

1. Where do we intercept? (hook)
2. How do we understand the file system without a file system?
3. How do we ensure we give our investigator a valid image?
4. How do we avoid detection?

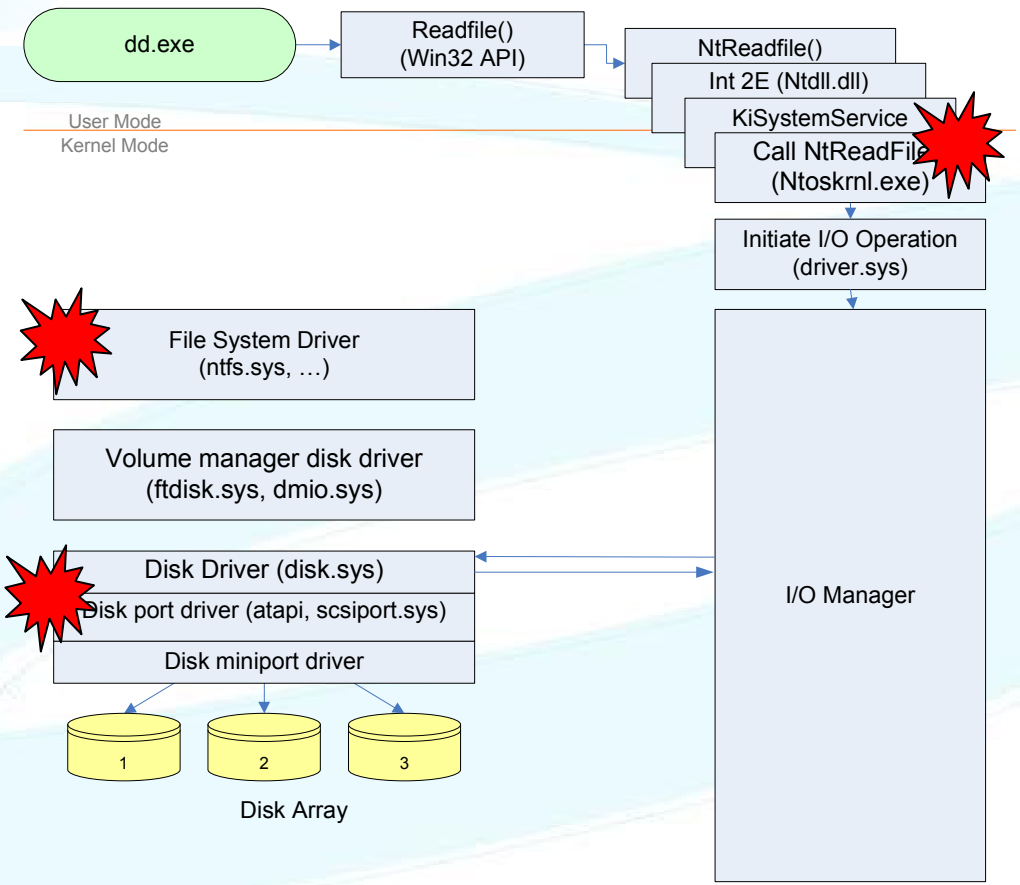
Where do we Intercept?

Public Userland (Ring 3) Rootkits



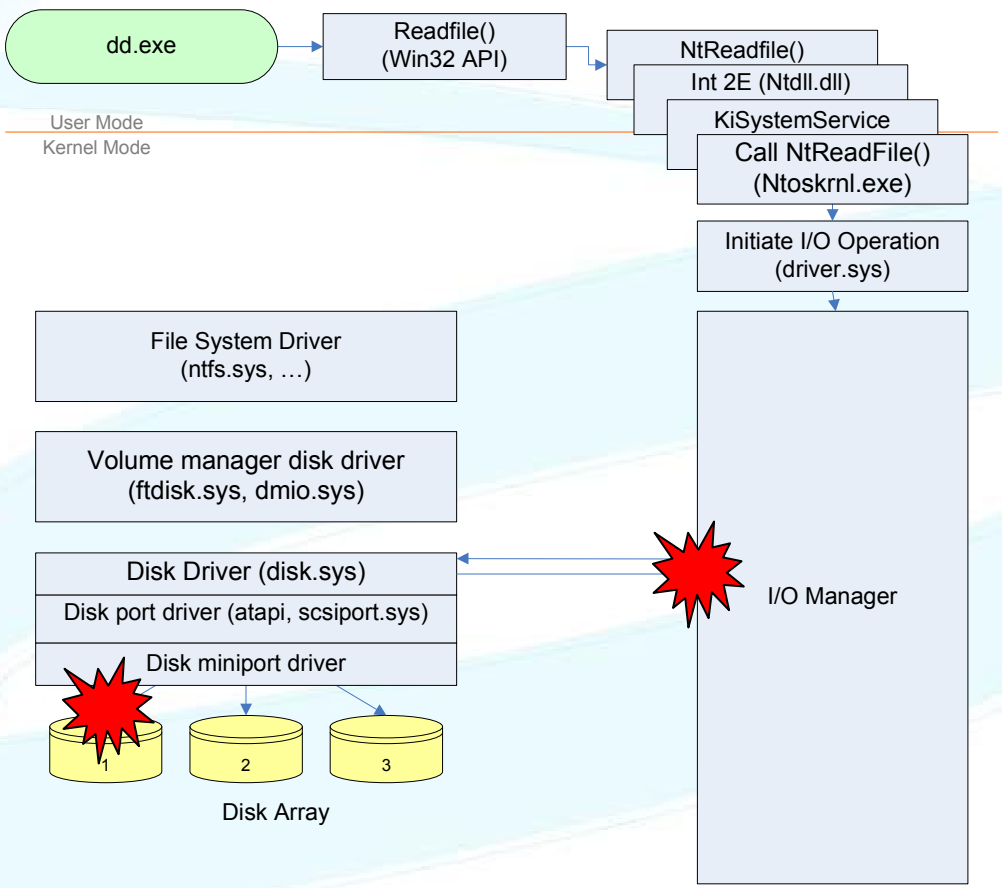
- **Binary replacement eg modified Exe or Dll**
- **Binary modification in memory eg He4Hook**
- **User land hooking eg Hacker Defender**
  - IAT hooking

Kernel (Ring 0) Rootkits



- **Kernel SDT Hooking**
  E.g. NtRootkit
- **Driver replacement**
  E.g. replace ntfs.sys with ntfss.sys
- **Direct Kernel Object Manipulation – DKOM**
  E.g. Fu, FuTo

Kernel (Ring 0) Rootkits

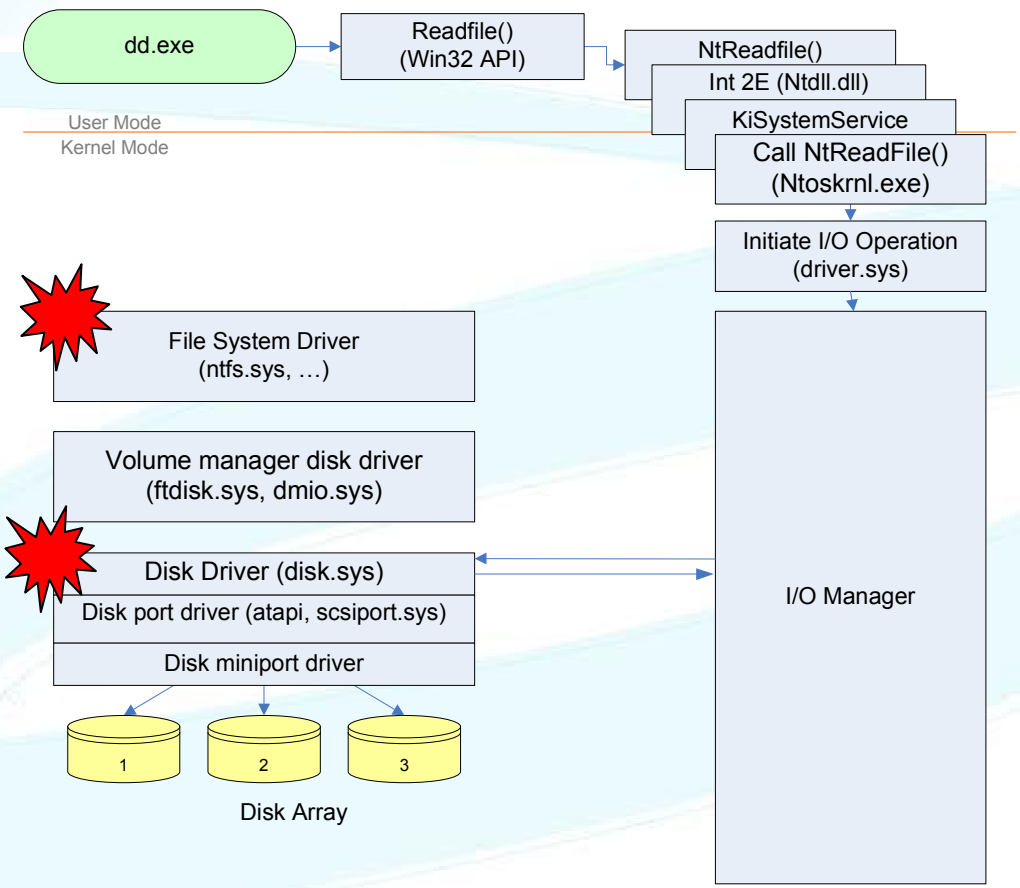


- **IO Request Packet (IRP) Hooking**
  - IRP Dispatch Table

  E.g. He4Hook (some versions)

- **Disk Firmware**
- **BIOS?**

Kernel (Ring 0) Rootkits



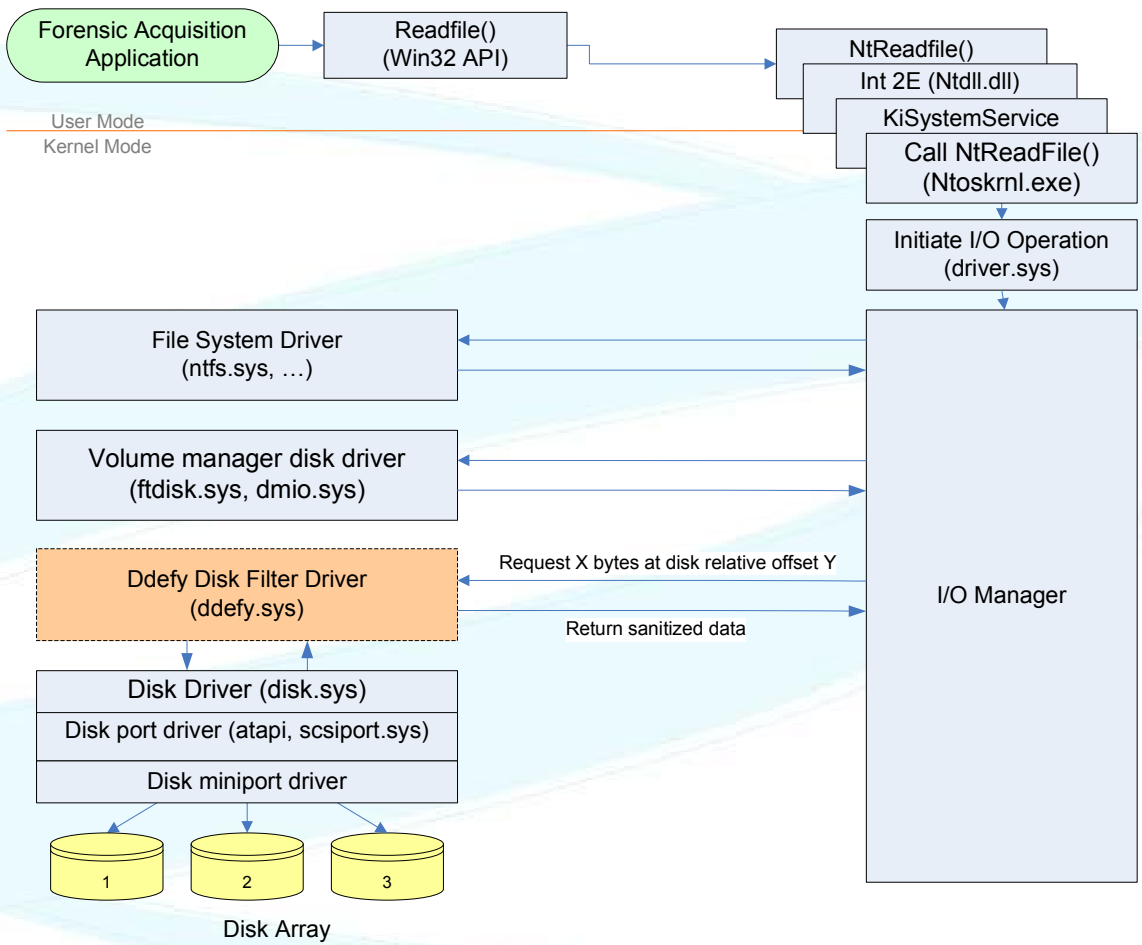
- **Filter Drivers**
  - The official Microsoft method
- **Types**
  - File system filter
  - Volume filter
  - Disk Filter
  - Bus Filter

E.g. Clandestine File System Driver (CFSD)

DDefy: Where Do We Intercept?

- **Wherever we want**
- **The Windows driver model leaves us a lot of scope**
- **Personally I like disk filter drivers or IRP hooks because**
  - a) They are simple to implement
  - b) They're not detected by most rootkit detectors

DDefy: Where We Hook



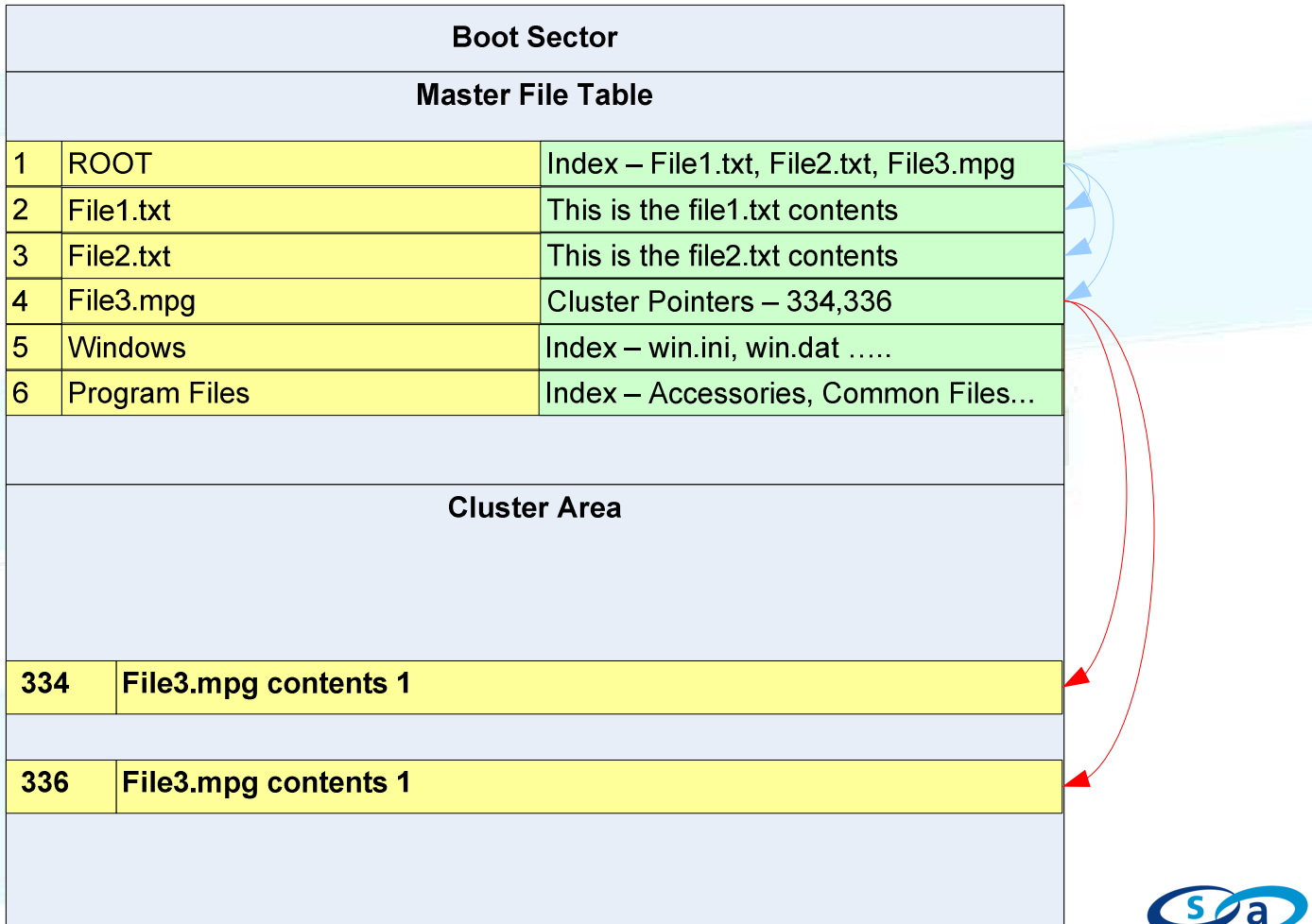
How Do We Understand the File System?

- NTFS is my target
- Easy thanks to recent publications (File System Forensics – B. Carrier)

- Ask the operating system to give us most of the information anyway

- We could do this ourselves, but I assume ntfs.sys is a better NTFS interpreter than I am

How Do We Understand the File System?



How Do We Understand the File System?

- **To completely hide a file from \\.\PhysicalDrive0 with DDefy we have to fake:**
  - Directory Entry (Index)
  - Master File Table (MFT) Entry
  - MFT Data (for small files)
  - Data in Clusters (for larger files)

Avoiding Detection

- This is all irrelevant if the forensic analyst finds "hacker.exe" running in memory
- So how does the forensics guy get his process listing? Network connections etc?
- \\.\PhysicalMemory
- Can we clean this up with some kernel modifications?

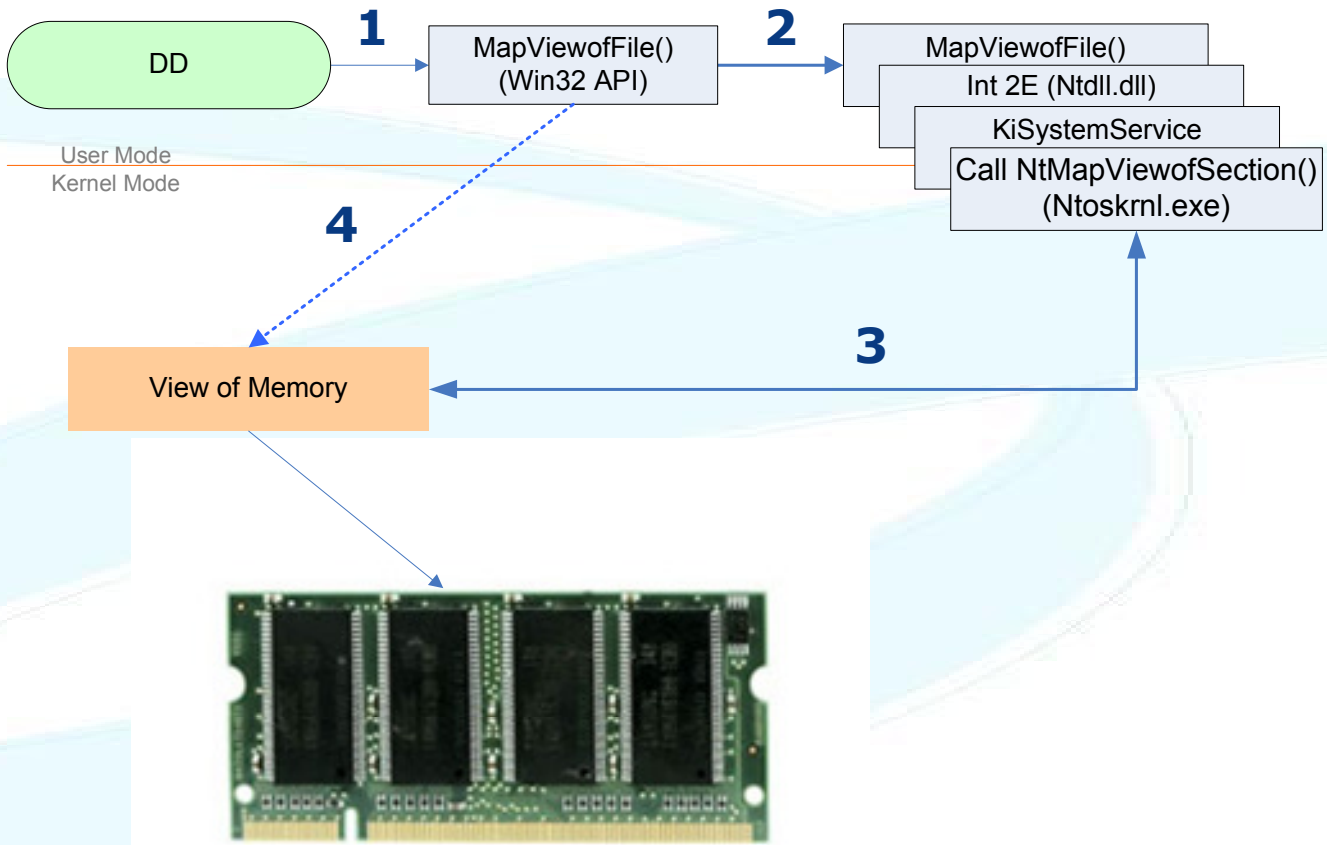
DDefy: Memory Interception

- **Standard rootkit SSDT hook on ZwMapViewofSection for all accesses to \\.\PhysicalMemory**

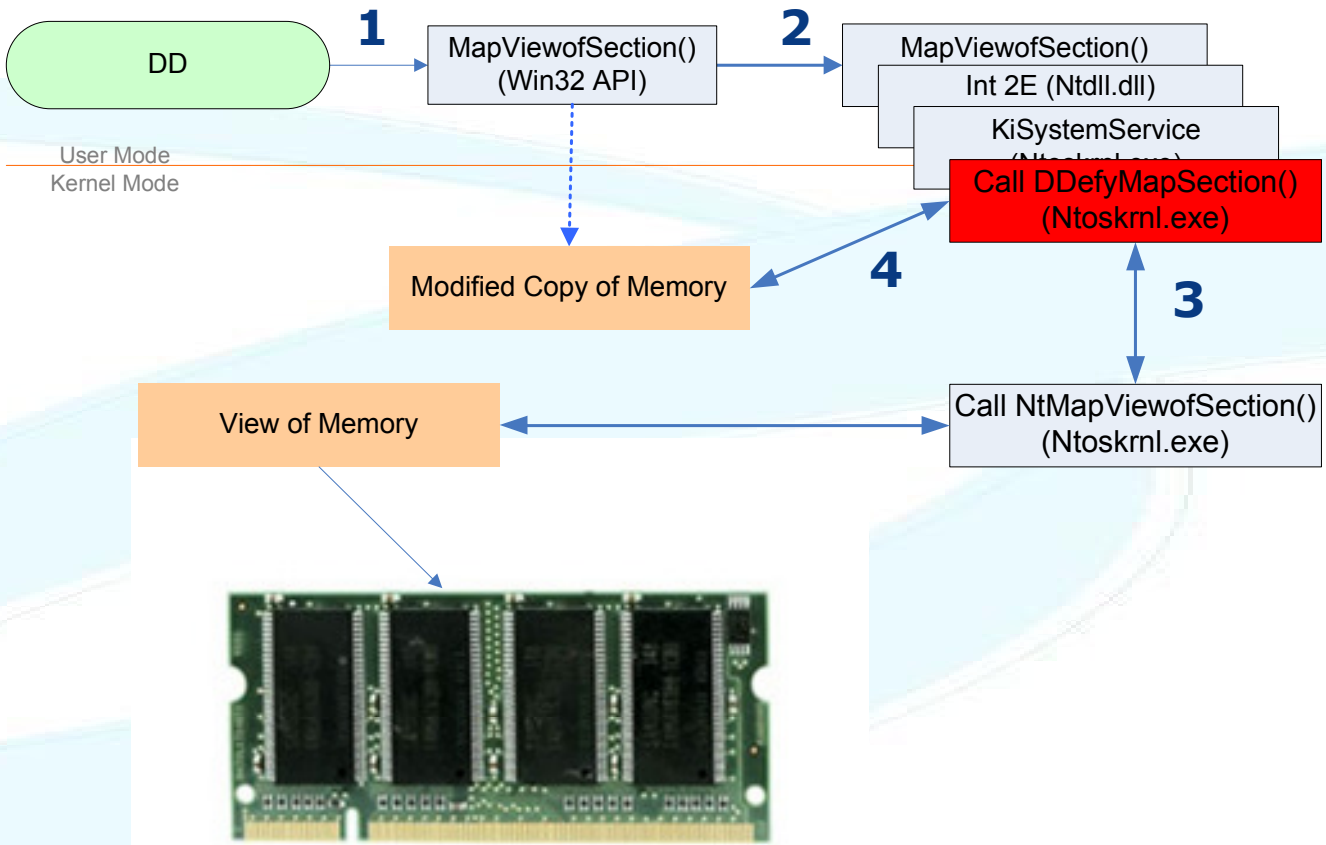
- **Modifies process and thread list**
- **Cleanses process contents**

- **Could be implemented using ShadowWalker technique (Sparks & Butler)**

DDefy: Copying Memory



DDefy: Manipulating Memory



DDefy: Manipulating Memory

```
NTSTATUS HookedZwMapViewOfSection(HANDLE  SectionHandle, HANDLE
    ProcessHandle, PVOID  *BaseAddress, …., PLARGE_INTEGER
    SectionOffset, …. )
{
 if ((SectionHandle == physical_memory_section_object)
          && (isHiddenMemory(SectionOffset, ProcessHandle)) {

          OldZwMapViewOfSection(*BaseAddress.....);
          RtlCopyMemory(buffer, BaseAddress);
          ZwUnmapViewOfSection(BaseAddress);
          MungeMemory(buffer, ProcessHandle);
          PhysicalAddress = MmGetPhysicalAddress(buffer);
          BaseAddress = NULL;
          OldZwMapViewOfSection(BaseAddress, PhysicalAddress, ...);
    } else {
          OldZwMapViewOfSection(BaseAddress, SectionOffset, ...);
    }
}
```

DDefy Memory Forensics Demo

C:\video\Mem Capture with DDefy.wmv

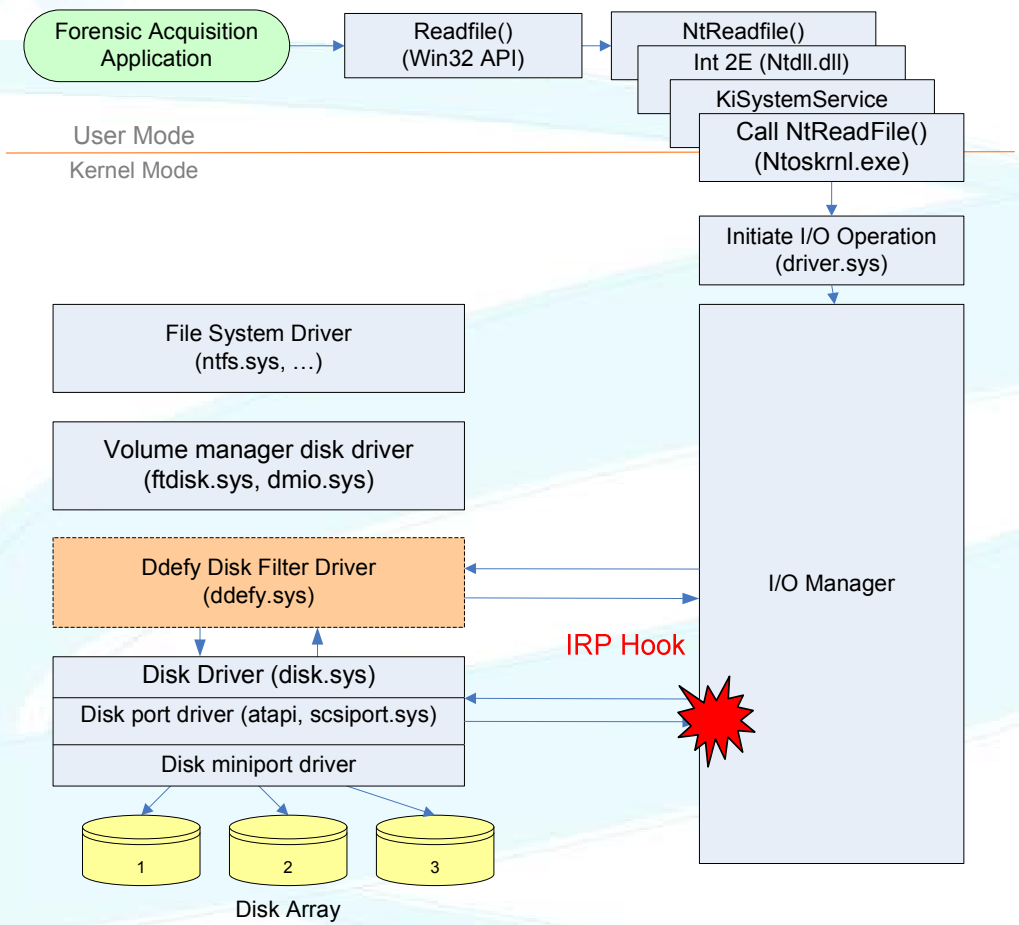
C:\video\Mem Analysis - Lame.wmv

DDefy Memory Interception

- **Detectable**
  - Ongoing rootkit detection arms race
- **Avoidable**
  - Zw Interception easily bypassed by multiple unpublished forensic memory techniques
  - Firewire works too ☺
- **Doesn't hide**
  - Memory mapped files
  - Memory paged to disk

Some solutions…

A Better Way of Acquiring Disk Data



- **Method**
  - Install IRP hook
  - Communicate directly with IRP hook
  - Encrypt communications
  - Confirm user land matches kernel land
- **Challenges**
  - Stability
  - OS Specific

A Better Way of Memory Imaging

- **Firewire (or any other bus)**
- **Utilise lower level functions**
- **Check the MapViewOfSection return against physical address**

What it Means

- **Live forensic imaging is a broken concept**
- **Data gathered via live imaging cannot be trusted**
- **Manipulation of evidence is both possible and probable**
- **However, live imaging is still useful if combined with the right knowledge**

Future Work

- **Defeating cross view rootkit detection tools in a generic way**

- **Implementation of an open source imaging tool to defeat these anti-forensic techniques**

Questions ?

http://www.security-assessment.com

darren.bilby@security-assessment.com

Resources

- **Windows System Internals 4th Edition–** D. Solomon, M. Russinovich
- **Rootkits –** G. Hoglund, J. Butler

- **Primary Windows Rootkit Resource**
  http://www.rootkit.com
- **Joanna Rutkowska – Stealth Malware Detection**
  http://www.invisiblethings.org
- **Windows Driver Development Resource**
  http://www.osronline.com