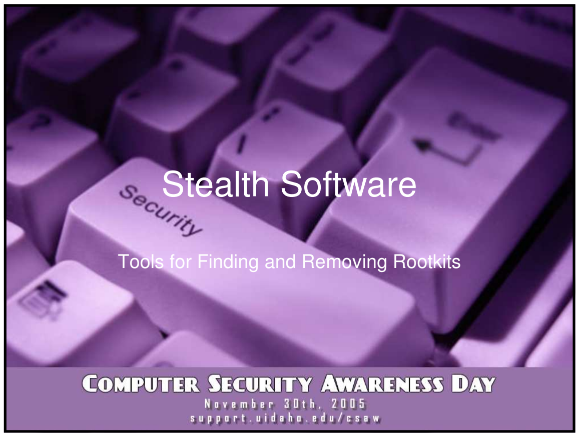
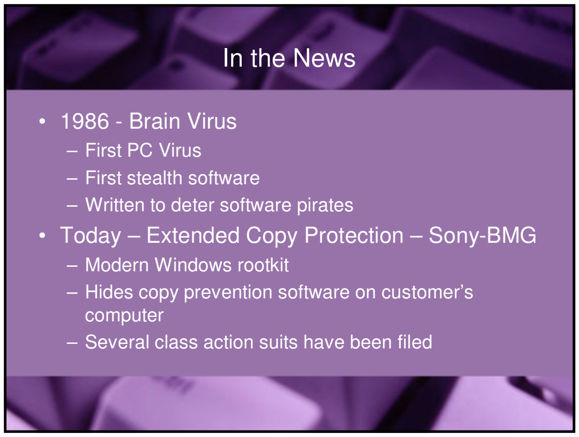


Stealth Software

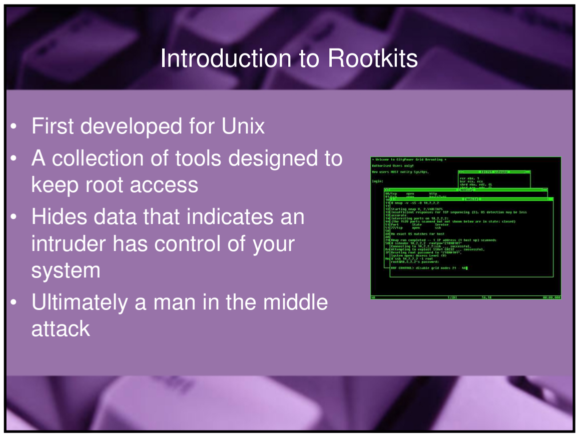
Tools for Finding and Removing Rootkits

COMPUTER SECURITY AWARENESS DAY
November 30th, 2005
support.uidaho.edu/csaw



In the News

- 1986 - Brain Virus
  - First PC Virus
  - First stealth software
  - Written to deter software pirates
- Today – Extended Copy Protection – Sony-BMG
  - Modern Windows rootkit
  - Hides copy prevention software on customer's computer
  - Several class action suits have been filed



Introduction to Rootkits

- First developed for Unix
- A collection of tools designed to keep root access
- Hides data that indicates an intruder has control of your system
- Ultimately a man in the middle attack

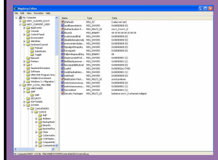
Hiding Techniques

- Hiding behind complexity
  - C:/Windows/
    - Windows hides directory by default to discourage casual viewing
    - C:/Windows/System/ has over 2000 files and 800 MB
  - Used by most “commercial” malware
    - Goals to infect the greatest number of novice users and probably make money from it

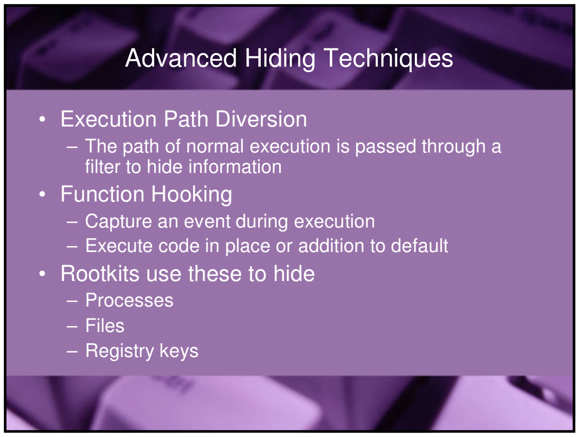
Hiding Techniques

- Filesystem tricks
  - Use system characters
    - Name folders or files ‘.’, ‘..’, ‘_’, ‘__’
    - Use similar characters
      - ‘l’ vs ‘1’ or ‘O’ vs ‘0’
      - Run32dl1.dll Run32dl1.dll
  - Utilize file attributes
    - Hidden, system, archive attributes
    - Novice users will not be able to see target files

Hiding Techniques

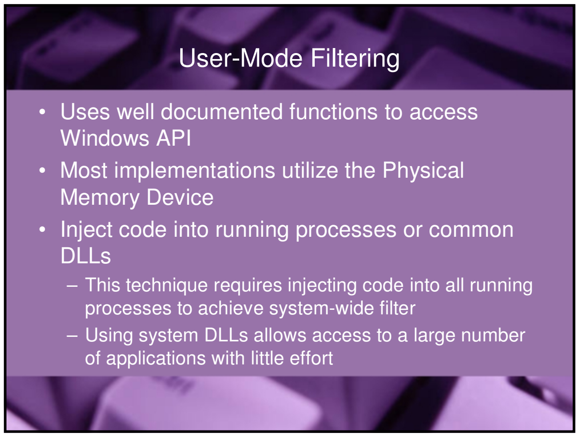


- Windows Registry
  - Database to record relationship between hardware, memory, application data
  - The vast size of the Registry makes it simple to hide information from even the most advanced user
    - Passwords
    - Binary data (applications, images, i.e.)
    - Start-up applications and services



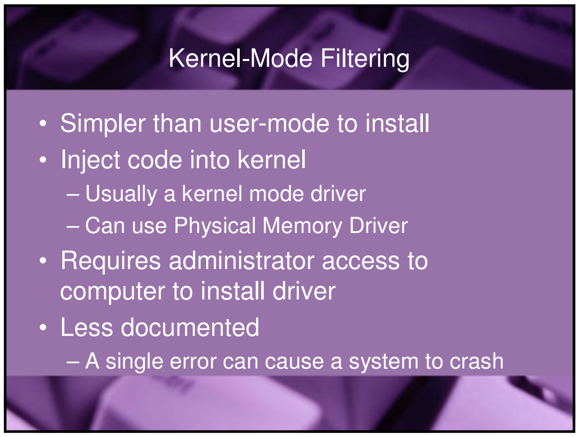
Advanced Hiding Techniques

- Execution Path Diversion
  - The path of normal execution is passed through a filter to hide information
- Function Hooking
  - Capture an event during execution
  - Execute code in place or addition to default
- Rootkits use these to hide
  - Processes
  - Files
  - Registry keys



User-Mode Filtering

- Uses well documented functions to access Windows API
- Most implementations utilize the Physical Memory Device
- Inject code into running processes or common DLLs
  - This technique requires injecting code into all running processes to achieve system-wide filter
  - Using system DLLs allows access to a large number of applications with little effort



Kernel-Mode Filtering

- Simpler than user-mode to install
- Inject code into kernel
  - Usually a kernel mode driver
  - Can use Physical Memory Driver
- Requires administrator access to computer to install driver
- Less documented
  - A single error can cause a system to crash

Physical Memory Device

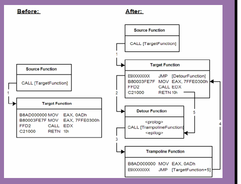
- A device driver to allow applications to write directly to memory
- Both Kernel-Mode and User-Mode rootkits utilize this device to inject code into running processes
- In recent service packs Microsoft has denied access to the device from User-Mode

Inline Hooking

- Most widely used
- Code is inserted into a running process
- Technique seen only in user-mode root kits
  - Kernel-mode inline hooking not well documented
  - User-mode and other techniques have been effective enough
  - Will probably change in the future

Inline Hooking

- Detour Functions
  - Patched into running code
  - Preprocessing
  - Calls “trampoline” function
    - Runs unpatched code
    - Returns control to detour function
  - Post Processing



Before:

Source Function
```
CALL [TargetFunction]
```

Target Function
```
B8AD000000 MOV   EAX, 0ADh
B80003FE7F MOV   EAX, 7FFE0300h
FFD2       CALL  EDX
C21000     RETN  10h
```

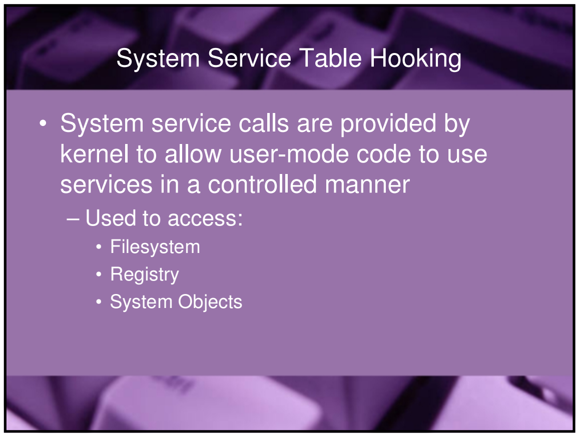
After:

Source Function
```
CALL [TargetFunction]
```

Target Function
```
E9XXXXXXXX   JMP   [DetourFunction]
B80003FE7F   MOV   EAX, 7FFE0300h
FFD2         CALL  EDX
C21000       RETN  10h
```

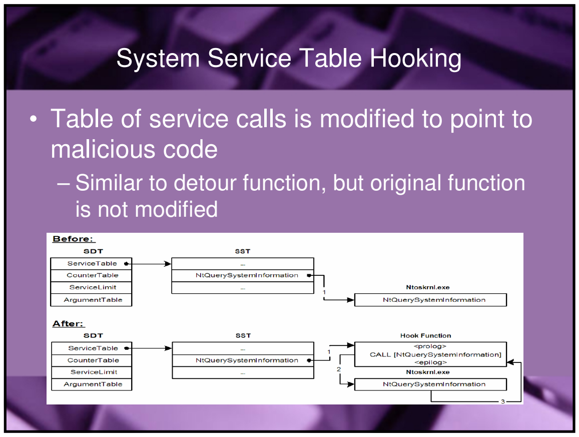
Detour Function
```
     <prolog>
CALL [TrampolineFunction]
     <epilog>
```

Trampoline Function
```
B8AD000000   MOV   EAX, 0ADh
E9XXXXXXXX   JMP   [TargetFunction+5]
```



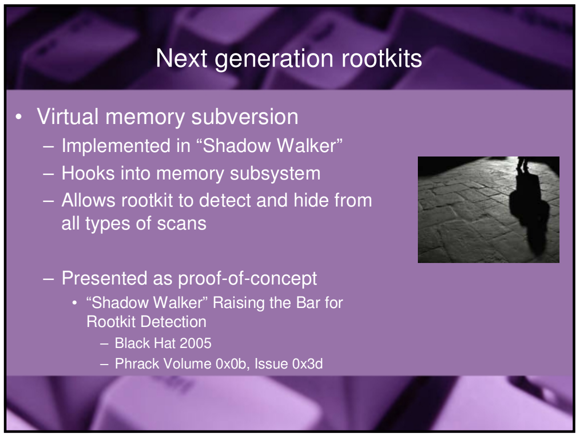
System Service Table Hooking

- System service calls are provided by kernel to allow user-mode code to use services in a controlled manner
  - Used to access:
    - Filesystem
    - Registry
    - System Objects



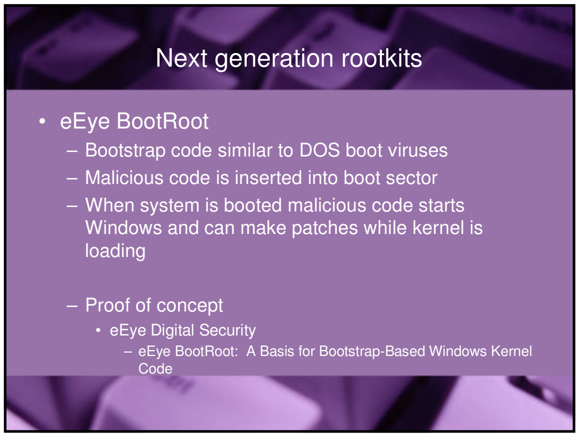
System Service Table Hooking

- Table of service calls is modified to point to malicious code
  - Similar to detour function, but original function is not modified



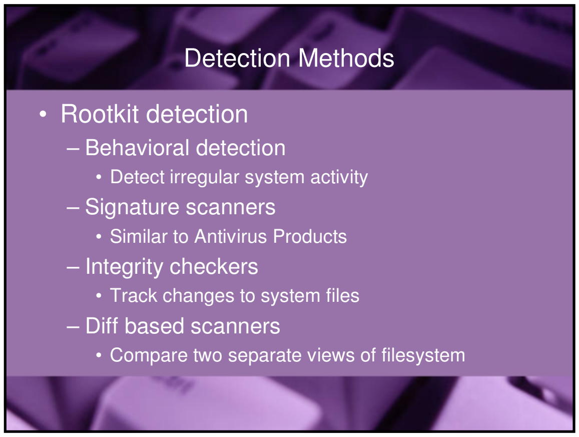
Next generation rootkits

- Virtual memory subversion
  - Implemented in “Shadow Walker”
  - Hooks into memory subsystem
  - Allows rootkit to detect and hide from all types of scans
  - Presented as proof-of-concept
    - “Shadow Walker” Raising the Bar for Rootkit Detection
      - Black Hat 2005
      - Phrack Volume 0x0b, Issue 0x3d



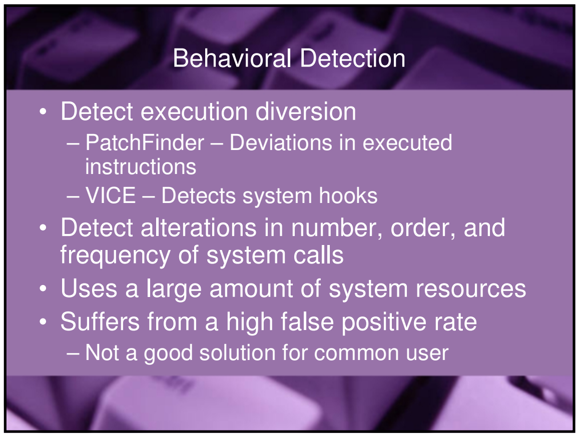
Next generation rootkits

- eYe BootRoot
  - Bootstrap code similar to DOS boot viruses
  - Malicious code is inserted into boot sector
  - When system is booted malicious code starts Windows and can make patches while kernel is loading
  - Proof of concept
    - eEye Digital Security
      - eEye BootRoot: A Basis for Bootstrap-Based Windows Kernel Code



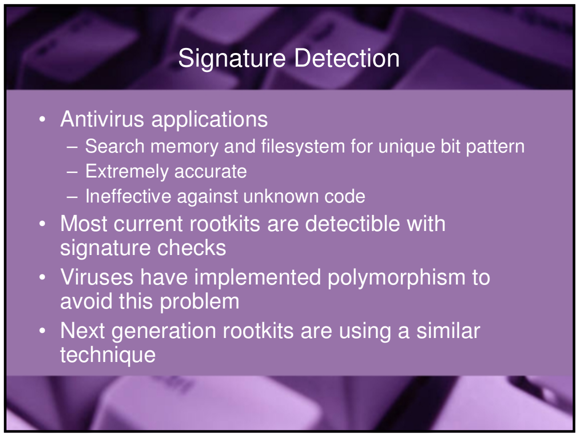
Detection Methods

- Rootkit detection
  - Behavioral detection
    - Detect irregular system activity
  - Signature scanners
    - Similar to Antivirus Products
  - Integrity checkers
    - Track changes to system files
  - Diff based scanners
    - Compare two separate views of filesystem



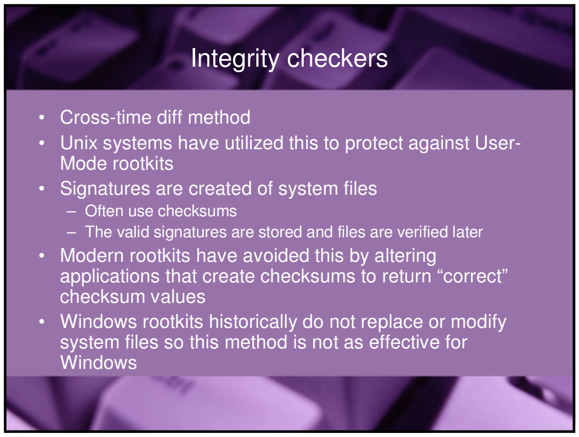
Behavioral Detection

- Detect execution diversion
  - PatchFinder – Deviations in executed instructions
  - VICE – Detects system hooks
- Detect alterations in number, order, and frequency of system calls
- Uses a large amount of system resources
- Suffers from a high false positive rate
  - Not a good solution for common user



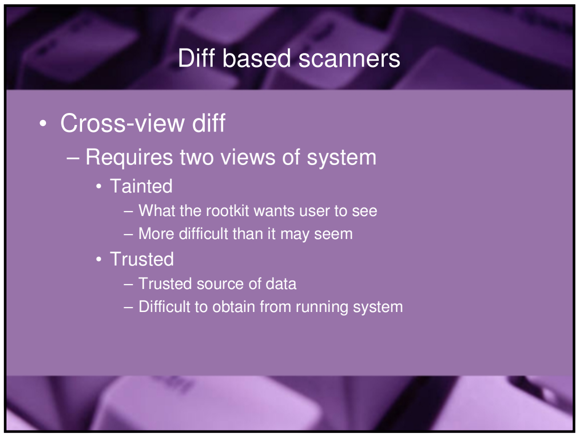
Signature Detection

- Antivirus applications
  - Search memory and filesystem for unique bit pattern
  - Extremely accurate
  - Ineffective against unknown code
- Most current rootkits are detectible with signature checks
- Viruses have implemented polymorphism to avoid this problem
- Next generation rootkits are using a similar technique



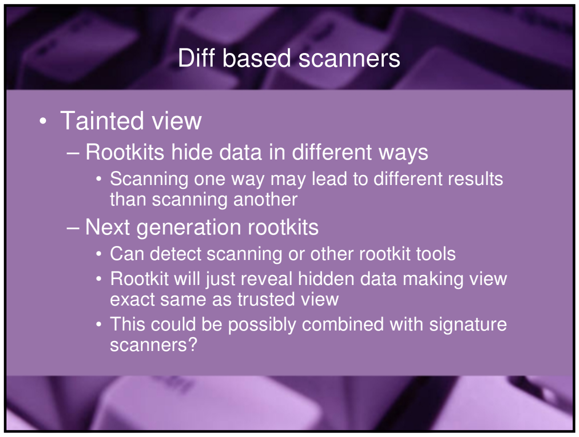
Integrity checkers

- Cross-time diff method
- Unix systems have utilized this to protect against User-Mode rootkits
- Signatures are created of system files
  - Often use checksums
  - The valid signatures are stored and files are verified later
- Modern rootkits have avoided this by altering applications that create checksums to return “correct” checksum values
- Windows rootkits historically do not replace or modify system files so this method is not as effective for Windows



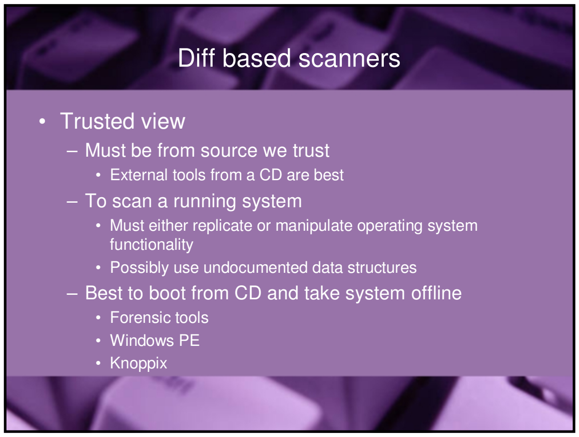
Diff based scanners

- Cross-view diff
  - Requires two views of system
    - Tainted
      - What the rootkit wants user to see
      - More difficult than it may seem
    - Trusted
      - Trusted source of data
      - Difficult to obtain from running system



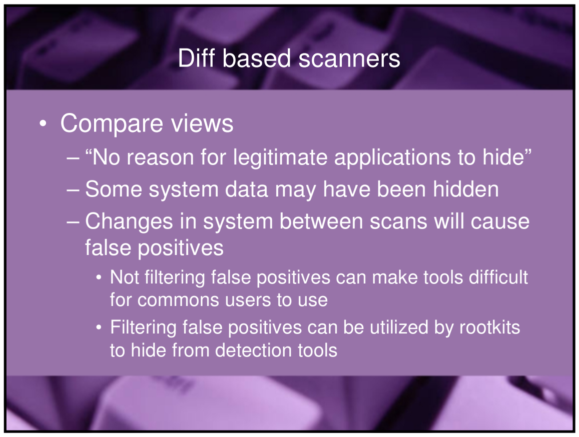
Diff based scanners

- Tainted view
  - Rootkits hide data in different ways
    - Scanning one way may lead to different results than scanning another
  - Next generation rootkits
    - Can detect scanning or other rootkit tools
    - Rootkit will just reveal hidden data making view exact same as trusted view
    - This could be possibly combined with signature scanners?



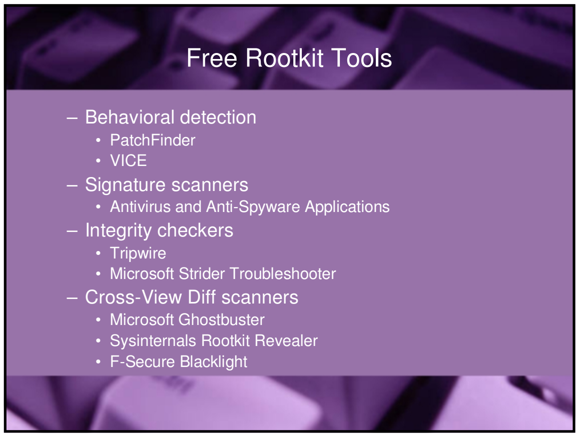
Diff based scanners

- Trusted view
  - Must be from source we trust
    - External tools from a CD are best
  - To scan a running system
    - Must either replicate or manipulate operating system functionality
    - Possibly use undocumented data structures
  - Best to boot from CD and take system offline
    - Forensic tools
    - Windows PE
    - Knoppix



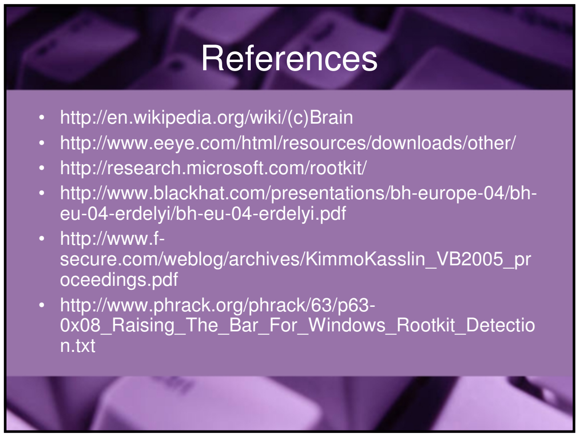
Diff based scanners

- Compare views
  - “No reason for legitimate applications to hide”
  - Some system data may have been hidden
  - Changes in system between scans will cause false positives
    - Not filtering false positives can make tools difficult for commons users to use
    - Filtering false positives can be utilized by rootkits to hide from detection tools



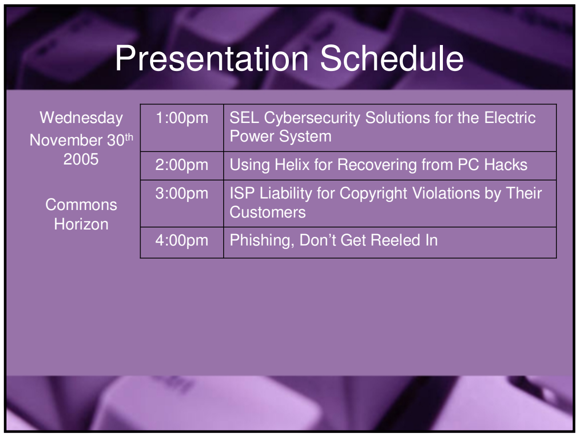
Free Rootkit Tools

- Behavioral detection
  - PatchFinder
  - VICE
- Signature scanners
  - Antivirus and Anti-Spyware Applications
- Integrity checkers
  - Tripwire
  - Microsoft Strider Troubleshooter
- Cross-View Diff scanners
  - Microsoft Ghostbuster
  - Sysinternals Rootkit Revealer
  - F-Secure Blacklight



References

- http://en.wikipedia.org/wiki/(c)Brain
- http://www.eeye.com/html/resources/downloads/other/
- http://research.microsoft.com/rootkit/
- http://www.blackhat.com/presentations/bh-europe-04/bh-eu-04-erdelyi/bh-eu-04-erdelyi.pdf
- http://www.f-secure.com/weblog/archives/KimmoKasslin_VB2005_proceedings.pdf
- http://www.phrack.org/phrack/63/p63-0x08_Raising_The_Bar_For_Windows_Rootkit_Detection.txt

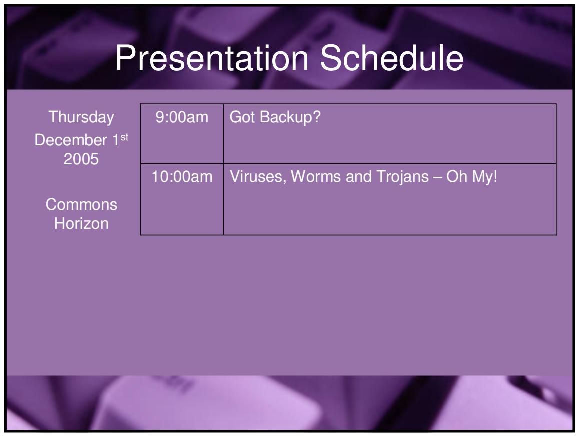


Presentation Schedule

Wednesday
November 30th
2005

Commons
Horizon

1:00pm	SEL Cybersecurity Solutions for the Electric Power System
2:00pm	Using Helix for Recovering from PC Hacks
3:00pm	ISP Liability for Copyright Violations by Their Customers
4:00pm	Phishing, Don't Get Reeled In



Presentation Schedule

Thursday
December 1st
2005

Commons
Horizon

9:00am	Got Backup?
10:00am	Viruses, Worms and Trojans – Oh My!