

Die Kunst der Datenverschleierung

Forensische Analysen: Gegenmaßnahmen unter Unix Dateisystemen



Marc McGuinness <marc@mcguinness.de>

Unix Friends and User Group (UnFUG) Furtwangen

17.11.2004

Inhalt

1. Einleitung
2. Unix Dateisysteme
3. Forensische Analysen
4. Schutz vor forensischen Analysen
5. Fragen & Antworten

Einleitung

◆ Wer bin ich?

- ◆ Marc McGuinness

◆ Was mache ich sonst so?

- ◆ Computer Networking
- ◆ Krankenhäuser in IT-Sicherheitsfragen beraten
- ◆ Institut für Business Consulting e.V.
- ◆ Lincolnshire Linux User Group

◆ Warum Datenverschleierung?

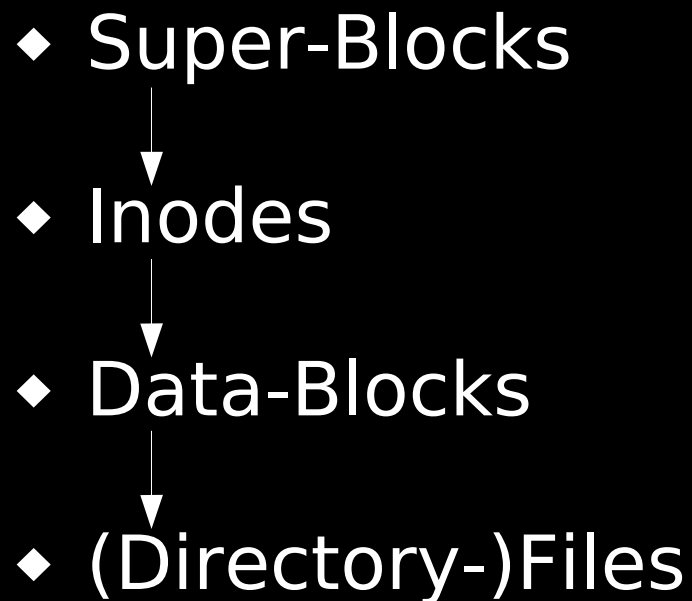
- ◆ Sicherheit ist ein Rüstungswettlauf
- ◆ Häufigerer Einsatz von forensischer Analyse
- ◆ Häufigerer Einsatz von Schutzmaßnahmen

Inhalt

1. ~~Einleitung~~
2. Unix Dateisysteme
3. Forensische Analysen
4. Schutz vor forensischen Analysen
5. Fragen & Antworten

Unix Dateisysteme

◆ Bestandteile eines Unix Dateisystems (1)



Dateisysteme Übersicht

2 Hauptbestandteile in jedem Dateisystem

1) Dateien

- ◆ Meta-Daten
 - ◆ Permissions, Owner, Size, usw.
- ◆ Daten
 - ◆ Blöcke gefüllt mit Bytefolgen

2) Meta-Dateien

- ◆ Verwaltung von Daten als Dateien für Menschen

Dateisystem

◆ Superblock

- ◆ Beschreibt das Dateisystem
- ◆ Ort des Superblocks ist bekannt

◆ Datenblock

- ◆ Speichert *ähm*, tja... Daten!
- ◆ Ein Block ist die kleinste atomare FS-Einheit
- ◆ Mehrere Festplattensektoren je Block

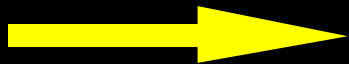
Unix Dateisysteme: inodes

- ◆ Inodes sind Dateien!
- ◆ Und nochmal: Sie speichern Meta-Daten!
 - ◆ Zeitstempel, Permissions, Größe, Reference Counts
- ◆ Speichert Referenzen auf Datenblöcke in einer Liste

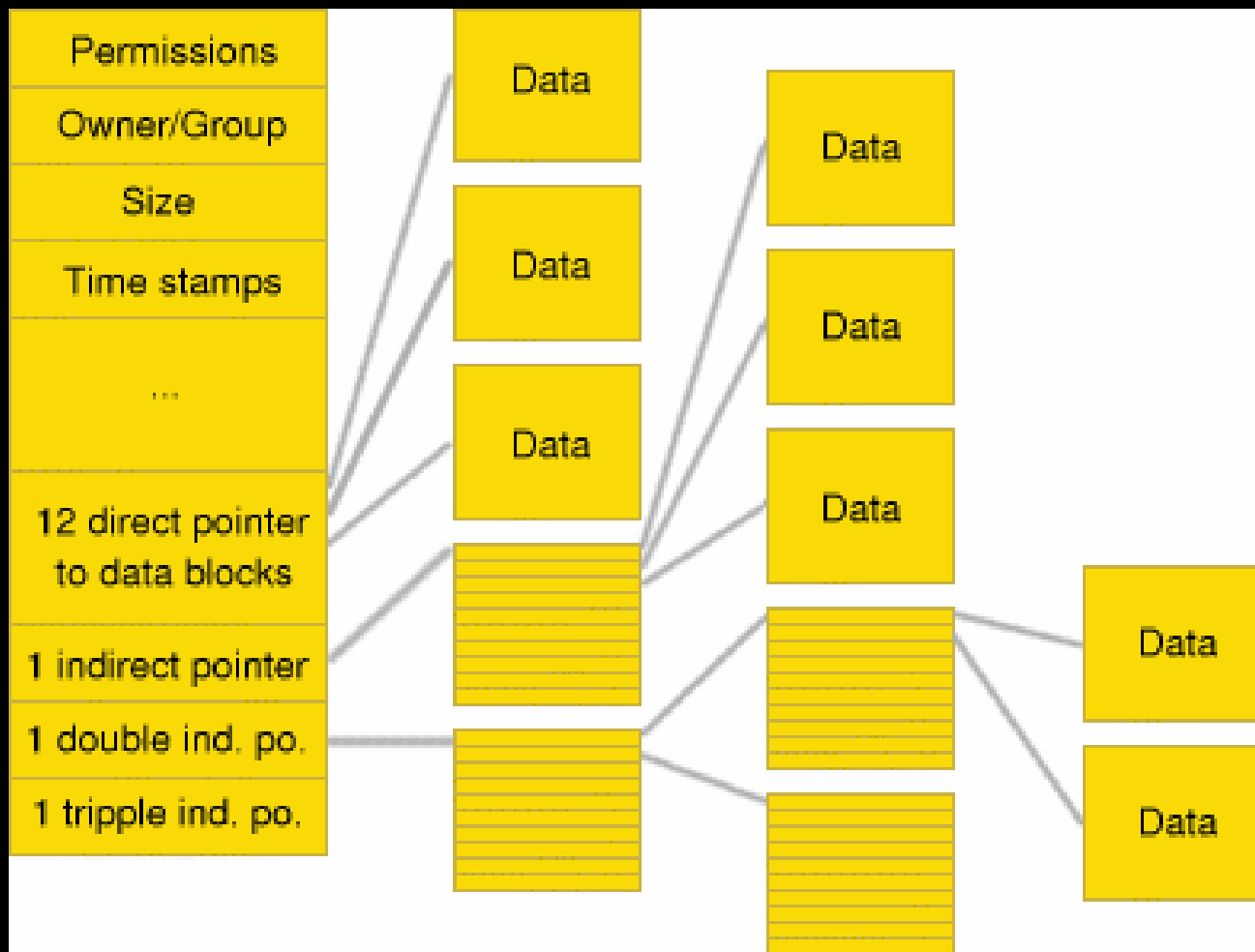
- ◆ block pointer

```
./linux-2.6.9/include/linux/ext2_fs.h:
```

```
211 struct ext2_inode {
212     __le16 i_mode; /* File mode */
213     __le16 i_uid; /* Low 16 bits of Owner Uid */
214     __le32 i_size; /* Size in bytes */
215     __le32 i_atime; /* Access time */
216     __le32 i_ctime; /* Creation time */
217     __le32 i_mtime; /* Modification time */
218     __le32 i_dtime; /* Deletion Time */
    ...
    ...
234     __le32 i_block[EXT2_N_BLOCKS]; /* Pointers to blocks */
```



Inode Struktur



Verzeichnisdateien

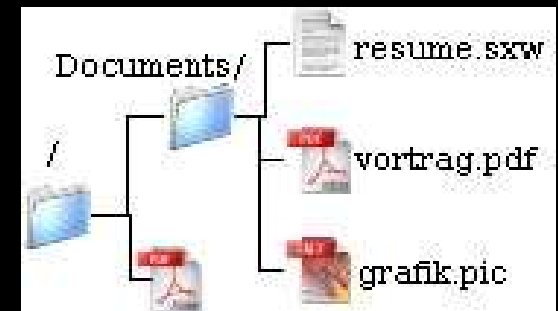
- ◆ Erstellen die Dateisystemhierarchie
- ◆ Bilden Verzeichniseinträgen auf inodes ab

```
./linux-2.6.9/include/linux/ext2_fs.h:  
519 struct ext2_dir_entry_2 {  
520     __le32 inode; /* Inode number */  
521     __le16 rec_len; /* Directory entry length */  
522     __u8 name_len; /* Name length */  
523     __u8 file_type;  
524     char name[EXT2_NAME_LEN]; /* File name */  
525 };
```

```
./linux-2.6.9/fs/ext2/dir.c:  
28 typedef struct ext2_dir_entry_2 ext2_dirent;
```

Documents/ inode #511

Datei	inode #
grafik.pic	33
resume.sxw	692
vortrag.pdf	453



Zusammenfassung

- ◆ **Super Block**
 - ◆ *Beschreibt das Dateisystem*

- ◆ **Inode**
 - ◆ Beschreibt Dateien

- ◆ **Data Block**

- ◆ **(Directory-)Files**
 - ◆ DNS des Dateisystems

Inhalt

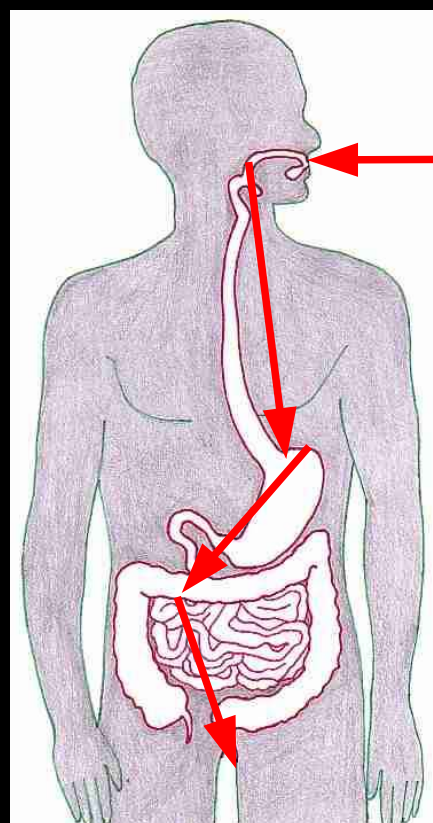
1. ~~Einleitung~~
2. ~~Unix Dateisysteme~~
3. Forensische Analysen
4. Schutz vor forensischen Analysen
5. Fragen & Antworten

Forensische Analysen

- ◆ **Einleitung**
- ◆ **Datenrettung**
- ◆ **Datenbestimmung**
- ◆ **Datenanalyse**

Einleitung

Ablauf einer forensischen Analyse



Bitstreams

Dateisysteme

Dateien

Beweise

Datenrettung

- ◆ Bitstream in ein Dateisystem übersetzen, danach
 - ◆ The Coroner's Toolkit (2) (für Unix)
 - ◆ *Stellt gelöschte Dateien wieder her*
 - ◆ The Coroner's Toolkit: Utils
 - ◆ *Untersucht gelöschte Verzeichniseinträge*
- ◆ Wenn das Dateisystem bekannt ist
 - ◆ werden gelöschte Daten gelesen

Datenbestimmung

- ◆ Dateisystem in Beweismittel umwandeln: Dateien (Bitstreamteile)
- ◆ Dateiinhalt erfordert ein Verständnis der Datenformate!
 - ◆ *Emails*
 - ◆ *Grafiken*
 - ◆ *Textdateien*
 - ◆ *... USW. ...*

Datenanalyse

- ◆ *Beweise aus den gewonnenen Daten extrahieren*
 - ◆ *Bsp.: JPEG-Dateien mit illegalem Inhalt*
 - ◆ *Bsp.: Logdateien mit Zugriffsinformationen*
- ◆ *Das Mittel der Wahl: Suche nach Schlüsselwörtern*

Forensische Analyse: Zusammenfassung

- ◆ Setzt voraus, dass das Dateisystem ein Logbuch über Systemaktivität ist
- ◆ Datenrettung
- ◆ Datenbestimmung
- ◆ Datenanalyse

Inhalt

- ~~1. Einleitung~~
- ~~2. Unix Dateisysteme~~
- ~~3. Forensische Analysen~~
4. Schutz vor forensischen Analysen
5. Fragen & Antworten

Schutz vor forensischen Analysen

- ◆ Daten sind Beweise!
- ◆ Grundregeln zum Schutz vor Forensik

1) *Datenvernichtung*

2) *Datenverschleierung*

3) *Datenvermeidung*

„Anzahl und Qualität der Beweismittel einschränken!“

Datenvernichtung

- ◆ Überbleibsel gelöschter Dateien
 - ◆ *inodes*
 - ◆ *Verzeichniseinträge*
 - ◆ *Datenblöcke*

- ◆ Dateisystemaktivität
 - ◆ *inode Zeitstempel*

Unser Toolkit (3)

- ◆ Necrofile (für Windows)
 - ◆ *Gelöschte inodes säubern*

- ◆ Klismafile
 - ◆ *Verzeichniseinträge säubern*
 - ◆ *Teil des TDT (The Defiler's Toolkit), aber fast unmöglich zu bekommen – siehe (3)*

Datenverschleierung

- ◆ Voraussetzungen
- ◆ Vorgehensweise
- ◆ Implementierung

Voraussetzungen

- ◆ Versteck
- ◆ Ausser Reichweite forensischer Tools
 - ◆ kurzfristig – langfristig unsicher
- ◆ Zuverlässig
 - ◆ Daten dürfen nicht verloren gehen
- ◆ Sicher
 - ◆ Datenzugriff nur mit den richtigen Tools möglich
 - ◆ Verschlüsselt

Vorgehensweise

***„Ich bin hier, um mit Euch über
FISTing zu reden!“***

Filesystem Insertion & Subversion Technique (FIST)

- ◆ Daten „einführen“, wo sie eigentlich nicht hingehören
- ◆ Datenspeicherung in Meta-Dateien
 - ◆ z.B. Journals, Verzeichnisse, usw.
- ◆ Meta-Daten zu manipulieren ist gefährlich!!!
 - ◆ passt auf fsck auf!
- ◆ Was können wir FISTen?

Implementierungen

- ◆ Rune FS
 - ◆ Speichert Daten im „Bad Blocks“ inode
- ◆ Waffen FS
 - ◆ Speichert Daten im ext3 Journal
- ◆ KY FS
 - ◆ Speichert Daten in Verzeichnisdateien
- ◆ Data Mule FS
 - ◆ Speichert Daten im für inodes reservierten Speicher

Rune FS

- ◆ Verwendet „Bad Blocks“-inode 1 (< „/“-inode 2)
- ◆ Nutzt fehlerhafte Implementierung in TCT aus
- ◆ Speichert bis zu 4 Gbyte Daten
- ◆ Ganz gewöhnliche inode

Waffen FS

- ◆ Fügt ext2 ein ext3 Journal hinzu
- ◆ Nutzt Lücken in e2fsck und forensischer Tools aus
 - ◆ e2fsck unterstützt ext2 und ext3
 - ◆ e2fsck errät um welches Dateisystem es sich handelt
- ◆ Speichert 32 Mbyte Daten (Größe eines Journals)
- ◆ Gewöhnliche Datei mit einem gefälschten Journalheader

KY FS (nicht verfügbar)

- ◆ Verwendet Verzeichnisskelette (gelöschte Verzeichnisse)
 - ◆ Für den kernel und fsck: inode == 0
 - ◆ Für forensische Tools: namelen == 0
- ◆ Nutzt Schwachstellen im Kernel, e2fsck und forensischer Tools aus
- ◆ Speicherplatz lediglich durch Festplatte begrenzt

Kill Your FileSystem

Daten Mule FS (nicht verfügbar)

- ◆ Speicherung innerhalb der Dateisystemstruktur
 - ◆ Reservierter Speicherplatz
 - ◆ Padding
- ◆ Ausser Reichweite für den Kernel und fsck
- ◆ Ignoriert durch forensische Tools

Daten Mule FS - Speicherplatz

- ◆ Super Block: 759 Byte
- ◆ Gruppendeskriptor: 14 Byte
- ◆ Inode: 10 Byte

- ◆ **1 Gbyte ext2** Dateisystem (default 4K Blocks)
 - ◆ 8 Gruppen
 - ◆ Super Blocks: 4 (3036 Byte)
 - ◆ Gruppendeskriptoren: 64 (896 Byte)
 - ◆ Inodes: 122112 (1221120 Byte)
 - ◆ Gesamt: 1225052 Byte = ~1196Kbyte = **~1Mbyte**

Datenvermeidung

***„Was ist das Besondere daran
Daten nicht zu erstellen?“***

Datenvermeidung

- ◆ Lieber Daten garnicht erst erstellen, als sie zu löschen
- ◆ Anzahl Beweismittel so gering wie möglich halten
 - ◆ Daten nicht ins Dateisystem gelangen lassen
 - ◆ Intra Userland Devices zur Kommunikation mit dem Kernel verwenden
- ◆ Qualität der Beweismittel senken
 - ◆ Durch den Einsatz von Standardtools (löschen, überschreiben, usw.)

Gute Rootkits

- ◆ Patchen im Arbeitsspeicher
 - ◆ kernel
 - ◆ sshd
 - ◆ Apache

- ◆ Nutzt vorhandene Standardtools, nicht selbstprogrammierte Programme

Vorsorgemaßnahmen

- ◆ IUDs erlauben den Zugriff auf Register und Speicheradressen
 - ◆ Intra Userland Device (gdb – GNU debugger) (4)
 - ◆ Inter Userland Device

- ◆ Prozess-Puppenspieler
 - ◆ Prozesse durch Proxies kontrollieren

gdb als IUD

- ◆ „syscall proxying“
- ◆ libgdbrpc
 - ◆ Führt Syscalls in einem Slave Prozess aus
 - ◆ Erlaubt Speicher- und Registerzugriff
 - ◆ mmap, mprotect, copy_to(), copy_from()
 - ◆ Textbasiert

Datenvermeidung: rexec v1 (5)

- ◆ Remote Ausführung von Code ohne Dateien auf der Festplatte zu erstellen
 - ◆ Verwendet gdb als IUD
 - ◆ Remote ein Prozessabbild erstellen
 - ◆ Prozesse wie ein Puppenspieler bedienen
- ◆ Hiermit tricksen wir Honeypots aus
 - ◆ Keine Daten – Keine Beweise

Userland Exec

- ◆ Ein Prozessabbild vom Buffer erstellen
 - ◆ `ul_exec(void *elf_buf, int argc, char **argv)`
- ◆ Ohne Festplattenzugriff
- ◆ Shared Object (Bibliothek)
- ◆ Wurde entwickelt 2004

ftrans (nicht verfügbar)

- ◆ Wurde entwickelt 2004
- ◆ Verwendet einen proprietären IUD Server und `ul_exec`
- ◆ grober Client
- ◆ Sichere Übertragung von Binaries mittels SSL
- ◆ Es handelt sich hierbei um eine „Anti-Honeypot“ Entwicklung

Datenvermeidung: rexec v2 (5)

- ◆ Verwendet libgdbrpc als IUD
- ◆ Lädt ein ELF Binary auf den Server
- ◆ Verwendet ul_exec()
- ◆ Entwickelt Juli 2004

Datenvermeidung: xsh (nicht verfügbar)

- ◆ eXploit Shell
- ◆ Funktionen
 - ◆ rexec v2
 - ◆ AscII Upload
 - ◆ Scripting
 - ◆ Kommando Aliase

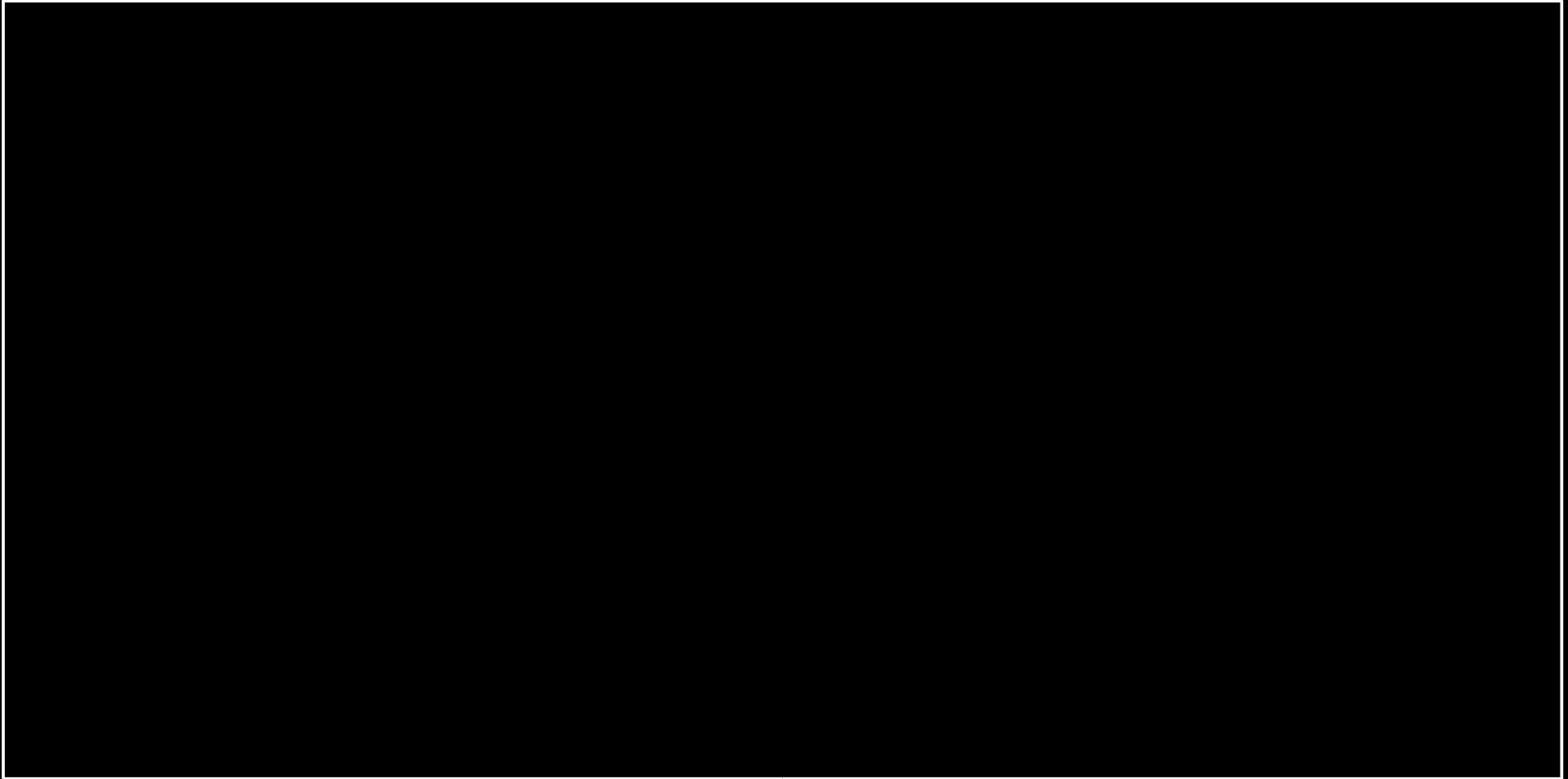
Zusammenfassung

- ◆ Einführung in ext2 und ext3 gegeben
- ◆ Übersicht über Computer Forensik gezeigt
- ◆ Grundlagen über Gegenmaßnahmen besprochen
- ◆ Simple Maßnahmen im Detail vorgestellt
- ◆ Eure Unix-Rechner ge0wned!!!

Inhalt

- ~~1. Einleitung~~
- ~~2. Unix Dateisysteme~~
- ~~3. Forensische Analysen~~
- ~~4. Schutz vor forensischen Analysen~~
5. Fragen & Antworten

Fragen?



Referenzen

- (1) <http://www.lug-eggenfelden.org/linuxfibel/filesys.htm>
- (2) <http://www.porcupine.org/forensics/tct.html>
- (3) <http://www.phrack.org/show.php?p=59&a=6>
- (4) <http://www.phrack.org/show.php?p=62&a=8>
- (5) <http://www.mksoftware.com/docs/man1/rexec.1.asp>