

The Computer Forensics Challenge and Anti-Forensics Techniques

EkoParty Security Conference 2007

Domingo Montanaro

<conferences@montanaro.org>

Buenos Aires, Diciembre 1, 2007

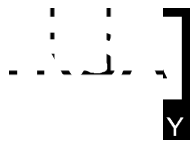
Agenda

Defeating forensics analysis

- **Subverting clones/imaging processes**
- **Backdoors/Rootkits/Whatever**
- **Etc ;D**

Data Remanence -> Magnetic Media

- **From erased data (covering some filesystems)**
- **From overwritten data**
- **From destroyed media**



Agenda

Defeating forensics analysis

- **Subverting clones/imaging processes**
- **Backdoors/Rootkits/Whatever**
- **Etc ;D**

Data Remanence -> Magnetic Media

- **From erased data (covering some filesystems)**
- **From overwritten data**
- **From destroyed media**

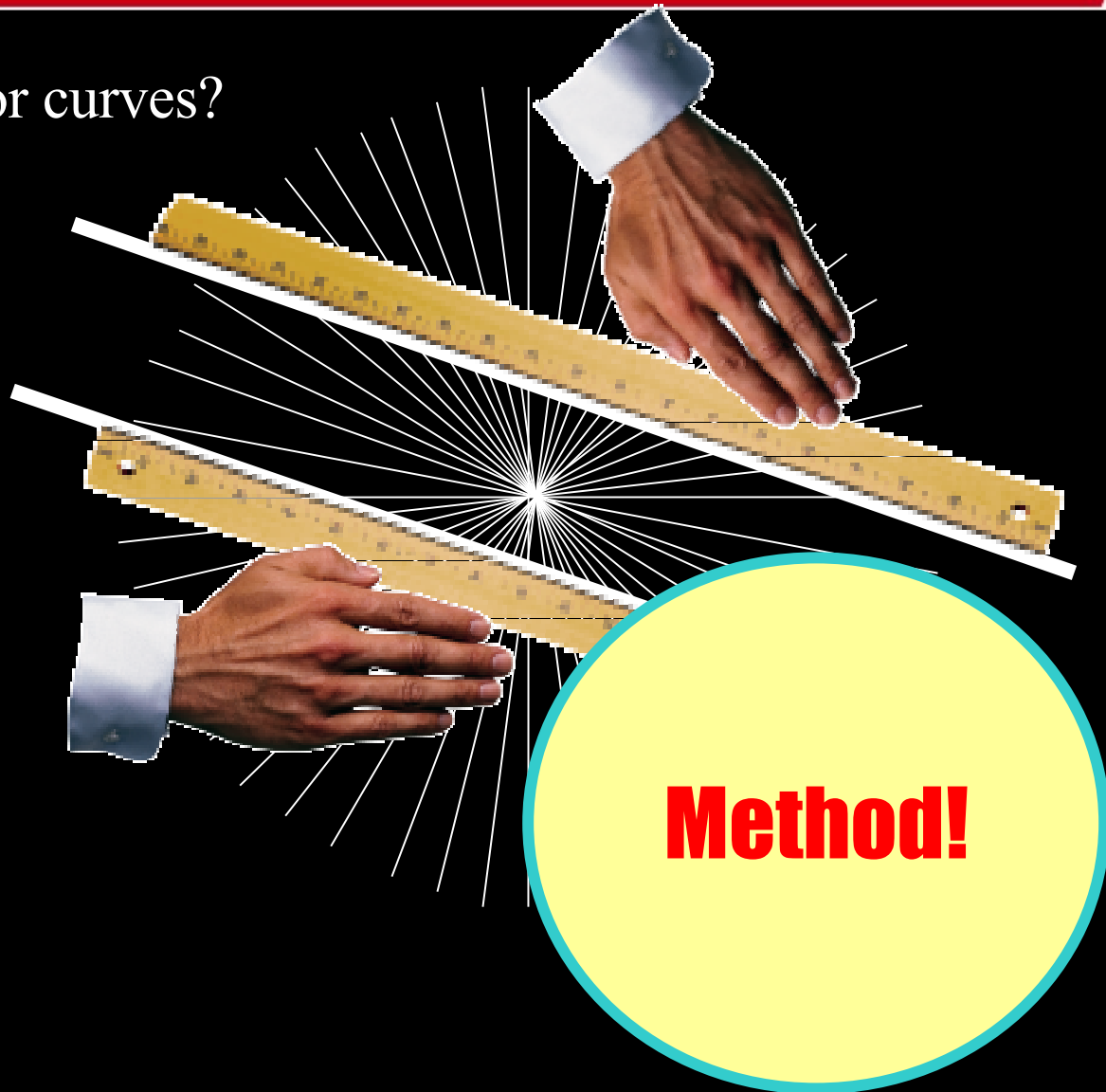
Being prepared to the incident

- Turn off or keep turned on the hw? It **Depends**
- RAM Clone ? **Always**
- Using the SO or hw specialized with DMA support?
- Take the HD out or clone? **Clone**
- Physical Manipulation of evidences? For Sure – Special equipment
- Hard Locks ? You kidding me, right?



Methodology

Straight Lines or curves?

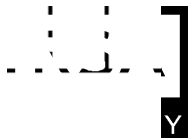


Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**



ADS – Alternate Data Streams

```
C:\ads>echo "Conteudo Normal" > teste.txt
```

```
C:\ads>echo "Conteudo Escondido" > teste.txt:escondido.txt
```

```
C:\ads>dir /a
```

```
Pasta de C:\ads
```

```
22/11/2004  00:59          <DIR>
```

```
22/11/2004  00:59          <DIR>
```

```
22/11/2004  00:59                20 teste.txt
```

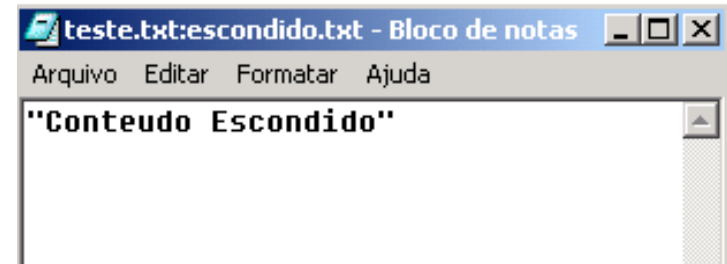
```
1 arquivo(s)                20 bytes
```

```
2 pasta(s)  1.696.808.960 bytes disponíveis
```

```
C:\ads>type teste.txt
```

```
"Conteudo Normal"
```

```
C:\ads>notepad teste.txt:escondido.txt
```

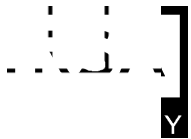


Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**



Hash Collision

```
black@bishop:~/quebra_md5$ ls
```

```
1.asc 1.bin 2.asc 2.bin resultado.txt
```

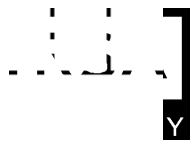
```
black@bishop:~/quebra_md5$ cmp 1.bin 2.bin
```

```
1.bin 2.bin differ: char 20, line 1
```

```
black@bishop:~/quebra_md5$ md5sum 1.bin 2.bin
```

```
79054025255fb1a26e4bc422aef54eb4 1.bin
```

```
79054025255fb1a26e4bc422aef54eb4 2.bin
```



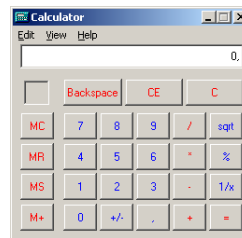
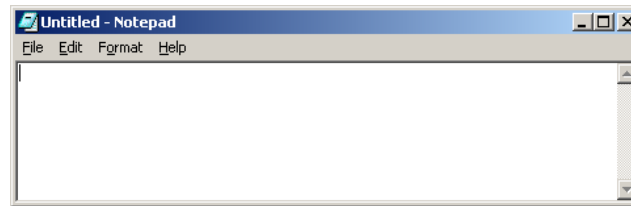
Hash Collision

Not indicated to use only MD5 nowadays

From: Gerardo Richarte - CORE SDI
MD5 to be considered harmful today

```
C:\WINDOWS\system32\cmd.exe - jane_0
P:\Estudos\hashes>jane_0
C:\DOCUME~1\MONTAN~1\CONFIG~1\Temp\SHA9C.tmp
```

```
C:\WINDOWS\system32\cmd.exe - jane_1
P:\Estudos\hashes>jane_1
C:\DOCUME~1\MONTAN~1\CONFIG~1\Temp\SHA9D.tmp
```



```
192.168.0.1 - PuTTY
thanathoes:~/quebra_md5# md5sum *.exe
a0124944226db6b68588ff8f30cbde87 jane_0.exe
a0124944226db6b68588ff8f30cbde87 jane_1.exe
thanathoes:~/quebra_md5#
```

Same MD5

```
192.168.0.1 - PuTTY
thanathoes:~/quebra_md5# cksfv *.exe
; Generated by cksfv v1.3.5 on 2007-02-15 at 18:15.07
; Originally Written by Bryan Call <bc@fodder.org>
; New versions maintained by Heikki Orsila <heikki.orsila@iki.fi>
; New versions can be obtained from http://www.iki.fi/shd/foss/cksfv/
;
; 558080 15:55.21 2005-12-02 jane_0.exe
; 558080 15:55.21 2005-12-02 jane_1.exe
jane_0.exe 4439F3A3
jane_1.exe 4439F3A3
thanathoes:~/quebra_md5#
```

Same CRC

Hash Collision

Again, not good to use only MD5

<http://www.doxpara.com/research/md5/confoo.pl>

confoo \$VERSION: Web Conflation Attack Using Colliding MD5 Vectors and Javascript

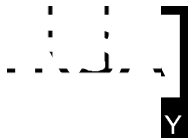
Author: Dan Kaminsky(dan\@doxpara.com)

Example: ./confoo www.lockheedmartin.com active.boeing.com/sitemap.cfm

Attack Vectors!

<http://www.doxpara.com/stripwire-1.1.tar.gz>

Stripwire emits two binary packages. They both contain an arbitrary payload, but the payload is encrypted with AES. Only one of the packages ("Fire") is decryptable and thus dangerous; the other ("Ice") shields its data behind AES. Both files share the same MD5 hash.

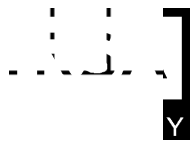


Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**



Simplistic Image Steganography

- Image files follow their layout standards, as of any other kind of file
- Each standard has it's own data hiding capabilities (GIF, BMP, TIFF, etc) – of course, not the original purpose

Ex: GIF89a

- Con: Not many tools to analyze file's layout, comparing it to a standard layout and a base of layout possibilities (out-of-range values in some fields)

And we are not even talking about the graphic part, which implies on techniques such as Color Reduction, LSB (Least Significant Bit) – noise, etc.



Dumbest stego method ;))

Nome	Tamanho	Tipo	Data de modificação
logo_h2hc	8 KB	Imagem no formato...	15/2/2006 18:44
trecho	585 KB	Winamp media file	15/2/2006 18:54

Two simple files

```
F:\Estudos\StegTest>copy logo_h2hc.gif /b + trecho.mp3
logo_h2hc.gif
trecho.mp3
1 arquivo(s) copiado(s).
F:\Estudos\StegTest>
```

Simply copy command

Nome	Tamanho	Tipo	Data de modificação
logo_h2hc	592 KB	Imagem no formato...	16/2/2007 15:05
trecho	585 KB	Winamp media file	15/2/2006 18:54

The 2 files continue, but notice the size of "logo_h2hc.gif"



Opening the file on the standard Image Visualization app, it comes up what was expected



Dragging and dropping the same GIF file on a winamp's window, we have 37 seconds of sound.

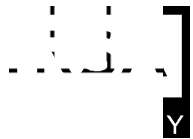


Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**

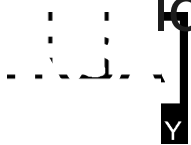


Slack Space

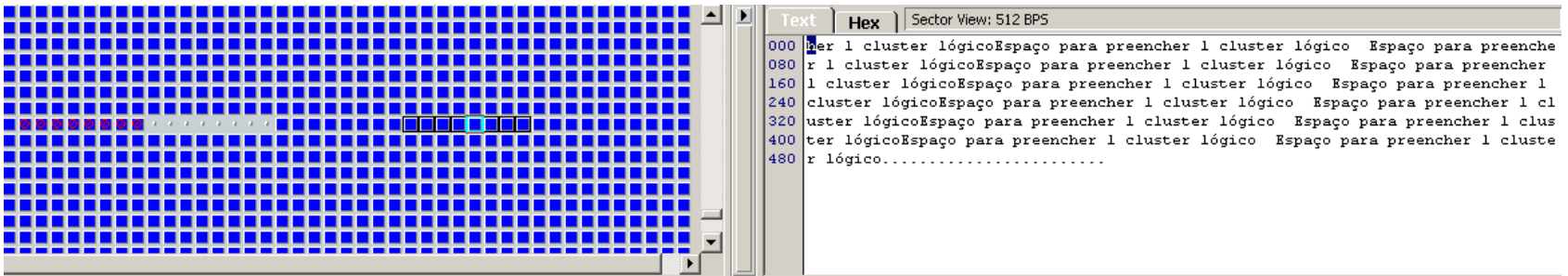
Non-addressable space in the MFT than can be written by specific tools (RAW)

- NTFS uses logical cluster of 4kb
- Files less than 4kb use 4kb (outside MFT)
- Tools can build a own MFT and address directly on the disk its own blocks to use as a container for the backdoor (and can mark it as bad block to the filesystem, so it would not be overwritten)
- Combining this to crypto/steganographic technics should make the forensics job much harder (and most of times when it's well done, efforts will be lost)

Update: Tool: Slacker from the Metasploit project

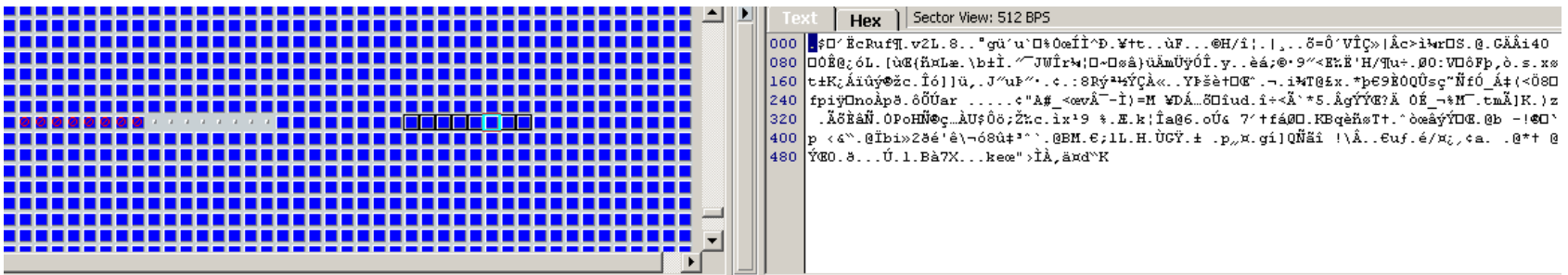


Slack Space



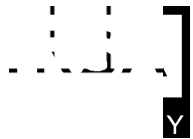
Sector View: 512 BPS

```
000 r 1 cluster lógicoEspaço para preencher 1 cluster lógico Espaço para preenche
080 r 1 cluster lógicoEspaço para preencher 1 cluster lógico Espaço para preencher
160 l cluster lógicoEspaço para preencher 1 cluster lógico Espaço para preencher 1
240 cluster lógicoEspaço para preencher 1 cluster lógico Espaço para preencher 1 cl
320 uster lógicoEspaço para preencher 1 cluster lógico Espaço para preencher 1 clus
400 ter lógicoEspaço para preencher 1 cluster lógico Espaço para preencher 1 cluste
480 r lógico.....
```



Sector View: 512 BPS

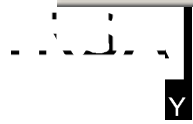
```
000 [;D'ÉcRuf7.v2L.8.. "gü'u"Q%0æíí^D.V+tc..ùF...@H/i!..l...8=ô'VÍÇ|Âc>i4rDS.@.GÃÄi40
080 Q0È@;ôL.[ûE(âMLe.\biî.~JWîr4;Q-Dsâ)ûâMÛyóí.y..éâ;@'9'<EhÈ'H/7tu+.00:V0ôFp,ð.s.xs
160 t±K;Áiúy@zç.Ió)]ü,..J'uf'..c.:8Ry*4YÇÀ<..YFÈè+QÇ'-.iMTEix.*pE9È0QÛsg"Ñfó_Á#(<08D
240 fpiyOnoâpâ.ôÜûar ...c"A# <ævÅ-i)=M WDÁ_S0iud.i+<Å'*5.ÂgY'YÇ?À OË_-+M'.tmÅ|K.)z
320 ..ÃõãÑ.0PoHÑç...ÀU;ôô;Zkc.ix'9 $.E.k;îa@6.oûâ 7'frâ0D.KBqêhst+.^ðæyYDÇ.@b -!@D'
400 p <æ\@Ibi>28é'ê\~ô8û+*'..@EM.Ç;1L.H.ÜCY.± .p,µ.gi|QÑâi !\Â..Euf.é/µç;.ca. .@*+ @
480 YEO.8...Ú.l.Bâ7X...keæ">IÀ,ând^K
```



Slack Space

19B751940	75 73 74 65 72 20 6C F3 67 69 63 6F 45 73 70 61	uster lógicoEspa
19B751950	E7 6F 20 70 61 72 61 20 70 72 65 65 6E 63 68 65	ço para preenche
19B751960	72 20 31 20 63 6C 75 73 74 65 72 20 6C F3 67 69	r 1 cluster lógi
19B751970	63 6F 0D 0A 45 73 70 61 E7 6F 20 70 61 72 61 20	co..Espaço para
19B751980	70 72 65 65 6E 63 68 65 72 20 31 20 63 6C 75 73	preencher 1 clus
19B751990	74 65 72 20 6C F3 67 69 63 6F 45 73 70 61 E7 6F	ter lógicoEspaço
19B7519A0	20 70 61 72 61 20 70 72 65 65 6E 63 68 65 72 20	para preencher
19B7519B0	31 20 63 6C 75 73 74 65 72 20 6C F3 67 69 63 6F	1 cluster lógico
19B7519C0	0D 0A 45 73 70 61 E7 6F 20 70 61 72 61 20 70 72	..Espaço para pr
19B7519D0	65 65 6E 63 68 65 72 20 31 20 63 6C 75 73 74 65	eencher 1 cluste
19B7519E0	72 20 6C F3 67 69 63 6F 00 00 00 00 00 00 00 00	r lógico.....
19B7519F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
19B751A00	10 24 8F 92 CB 63 52 75 66 B6 11 76 32 4C 01 38	.\$ 'ÉcRuf%.v2L.8
19B751A10	18 01 B0 67 FC B4 75 60 8D 25 D2 9C CD CC 5E D0	..*gü'u` %Ô íí^Ð
19B751A20	06 A5 86 74 06 12 F9 46 1B 02 17 A9 48 2F EE A6	.# t..ùF...@H/i
19B751A30	10 7C B8 05 18 F5 3D D4 B4 56 CE C7 BB 2A 2A 2Aõ=Ô'VÍÇ»***
19B751A40	2A 54 45 58 54 4F 20 45 53 43 4F 4E 44 49 44 4F	*TEXTO ESCONDIDO
19B751A50	2A 2A 2A 40 BF F3 4C 03 5B F9 8C 7B F1 A4 4C E6	***@íóL.[ù {ñ²Læ
19B751A60	08 5C 62 B1 CC 07 94 AF 4A 57 CE 72 BC A6 9D 7E	.\bñì. JWÍr% ~
19B751A70	7F F8 E2 7D FC C4 6D DC FF D3 CE 14 79 1E 18 E8	øá}üÄmÜyÓÍ.y..è
19B751A80	E1 3B A9 B7 39 94 3C 45 89 CB 27 48 2F B6 75 F7	á;@.9 <E É'H/¶u+
19B751A90	02 D8 4F 3A 56 8D F4 46 FE 2C F2 0C 73 1A 78 F8	.@0:V ôFp,ò.s.xø
19B751AA0	74 B1 4B BF C1 EF FB FD AE 9E 63 07 CE F3 5D 5D	t±KçÁiúy@[c.Íó]]
19B751AB0	FC 2C 0E 4A 94 75 DE 94 B7 0B A2 12 3A 38 52 FD	ü. .J uþ . . . :8Rý
19B751AC0	AA BD DD C7 C0 AB 1D 07 59 DE 9A E8 86 81 8C 88	è%YÇÀ« . Yp è
19B751AD0	1C AC 04 69 BE 54 40 A3 78 0B 2A FE 80 39 C8 D2	..i%T@fx.*þ 9ÉÒ

->Hidden Data

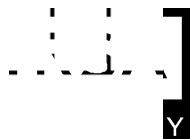


Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**



Use of redundant/Zero/Align spaces

Executables (ELF, Win32PE, etc) when compiled, depending on the compiler, most of the times need to have some space for alignment between subroutines.

Not a new idea in the IT field, since it's used by virii coders (injecting malware instructions into space used for alignment)

```
4AD051A5: C3 RETN ; end of subroutine
4AD051A6: 90 NOP ;
4AD051A7: 90 NOP ;
4AD051A8: 90 NOP ;
4AD051A9: 90 NOP ;
4AD051AA: 55 PUSH EBP ; begin of next subroutine
```

Alignment that can be used to store data
Can be 0x90, 0xCC or signature-based like GCC

On a 2GB “system” filesystem, it's possible to store nearly 1 MB on a “Second Filesystem” : inside the “system” filesystem, only using alignment spaces (including DLLs) – Need to remember that relative (short) Jumps are needed to return in the program normal flow.

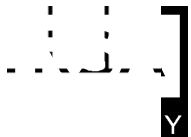
Going even deeper

So, every filetype has it's possibilities of storing “evil” data, not regarding compression formats.

Harmful to think on all this knowledge about hiding information (stego) in files to come in a toolkit.

Scenario:

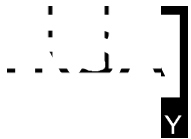
LibStego – Supports data hiding on several file formats, applying the parsing tons of these formats from wotsit.org



libStego

Supports 3 modes of operation

- 1) Growing up files – Ex: comments on graphic files (as showed before)**
- 2) Use redundant space on Multimedia formats (GIF, JPEG, AVI, MOV, etc), OLE formats (doc, xls, ppt, etc – not talking about compression here too) and others (DWG, CDR, etc)**
- 3) Use of alignment space on executable files (PE, ELF, etc)**



The libStego Project - Examples

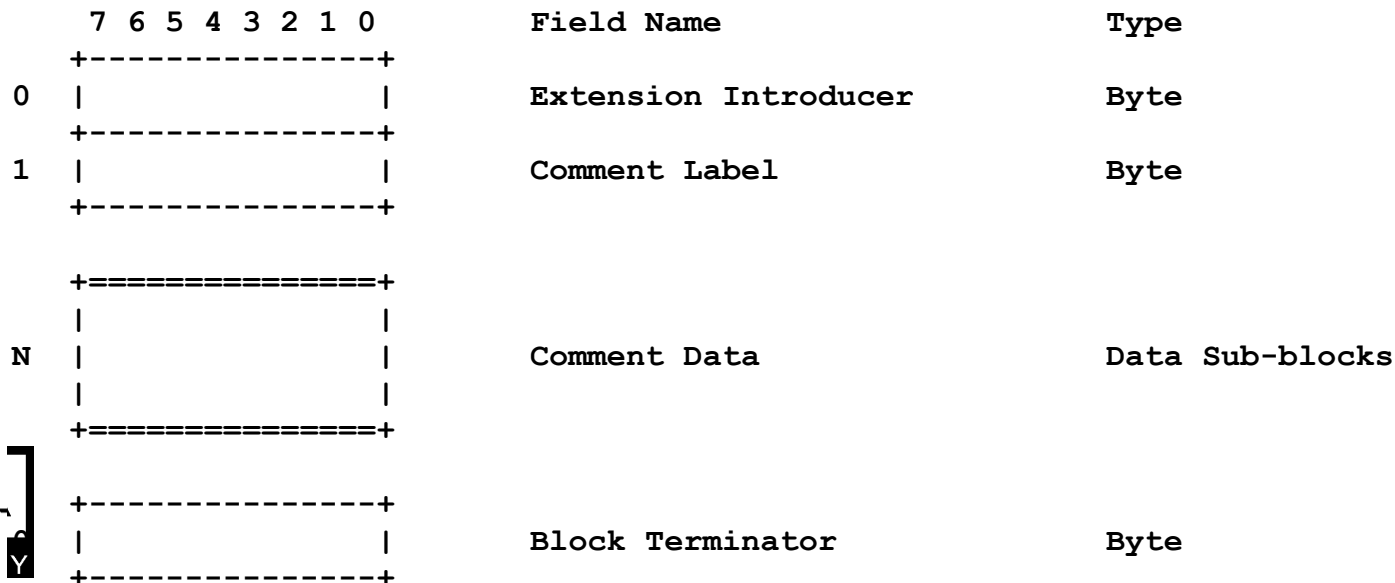
Field “Comment Extension” in GIF89a from CompuServe Graphics Interchange Format

24. Comment Extension.

a. **Description.** The Comment Extension contains textual information which is not part of the actual graphics in the GIF Data Stream. It is suitable for including comments about the graphics, credits, descriptions or any other type of non-control and non-graphic data. The Comment Extension may be ignored by the decoder, or it may be saved for later processing; under no circumstances should a Comment Extension disrupt or interfere with the processing of the Data Stream. This block is **OPTIONAL**; any number of them may appear in the Data Stream.

b. Required Version. 89a.

c. Syntax.



The libStego Project - Examples

Comments Chunk in Wave File Format

Comments Chunk Format

```
#define CommentID 'COMT' /* chunkID for Comments Chunk */

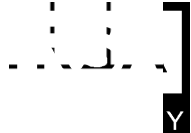
typedef struct {
    ID            chunkID;
    long          chunkSize;

    unsigned short numComments;
    char          comments[];
}CommentsChunk;
```

The ID is always COMT. chunkSize is the number of bytes in the chunk, not counting the 8 bytes used by ID and Size fields.

The numComments field contains the number of Comment structures in the chunk. This is followed by the Comment structures, one after the other. Comment structures are always even numbers of bytes in length, so there is no padding needed between structures.

The Comments Chunk is optional. No more than 1 Comments Chunk may appear in one FORM AIFF.



The libStego Project - Examples

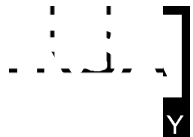
Field “JFIF extensions” in JPEG File Interchange Format Version 1.02

APP0 marker used to specify JFIF extensions

Additional APP0 marker segment(s) can optionally be used to specify JFIF extensions. If used, these segment(s) must immediately follow the JFIF APP0 marker. **Decoders should skip any unsupported JFIF extension segments and continue decoding.**

Immediately following the JFIF APP0 marker segment may be a JFIF extension APP0 marker. This JFIF extension APP0 marker segment may only be present for JFIF versions 1.02 and above. The syntax of the JFIF extension APP0 marker segment is:

```
X'FF', APP0, length, identifier, extension_code, extension_data
  length      (2 bytes)      Total APP0 field byte count, including the byte
                           count value (2 bytes), but excluding the APP0
                           marker itself
  identifier   (5 bytes)      = X'4A', X'46', X'58', X'58', X'00'
                           This zero terminated string ("JFXX") uniquely
                           identifies this APP0 marker. This string shall
                           have zero parity (bit 7=0).
  extension_code (1 byte)      = Code which identifies the extension. In this
                           version, the following extensions are defined:
                           = X'10'   Thumbnail coded using JPEG
                           = X'11'   Thumbnail stored using 1 byte/pixel
                           = X'13'   Thumbnail stored using 3 bytes/pixel
  extension_data (variable)    = The specification of the remainder of the JFIF
                           extension APP0 marker segment varies with the
                           extension. See below for a specification of
                           extension_data for each extension.
```



The libStego Project - Examples

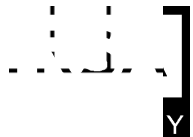
Comments on PDF files

From the “Portable Document Format Reference Manual” Version 1.3:

5.14 Body

The body of a PDF file consists of a sequence of indirect objects representing a document. The objects, which are of the basic types described in Chapter 4, represent components of the document such as fonts, pages, and sampled images.

Comments can appear anywhere in the body section of a PDF file. Comments have the same syntax as those in the PostScript language; they begin with a % character and may start at any point on a line. All text between the % character and the end of the line is treated as a comment. Occurrences of the % character within strings or streams are not treated as comments.

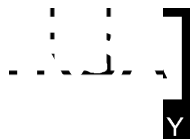


Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**



False positive about Defects

Most of data recovery softwares work through BIOS (int 13h) or the OS to access disk clusters

1 Cluster normally consists in 1 header, 512 bytes and ECC byte

When Recovery Software tries to get a cluster from the HD, if it comes with a ECC bad checksum, it will assume that this specific cluster is a “bad cluster”

One not-that-hard-to-code backdoor can simply forge this ECC bad checksum (error types “UNC” – Uncorrectable data - or AMNF – Address Mark Not Found) statically or dynamic to keep it's code on the media hard-to-find.

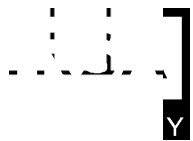
So, to achieve reading of these sectors, some ATA commands that ignore ECC need to be issued to recover byte-a-byte rather than sector-per-sector as most OS and BIOS do.

Methodology

Forensics analysis require deep information technology knowledge

Just a few examples that can simply modify the “guilty-non guilty” boolean variable:

- **ADS**
- **MD5**
- **Simple image stego**
- **Slack Space**
- **Hiding data inside the "visible" filesystem**
- **Forging fake HDD defects**
- **Rootkits - Subverting the first step – Imaging**



Userland protections



We enjoyed this picture from Julie Tinnes presentation on Windows HIPS evaluation with Slipfest

After kernel compromise, life is never the same

There are many techniques in the wild to subvert forensics analysis

In ring0 fights, it's all a mess. -> Let's protect the ring0!

First thing we should do to analyze a compromised machine is to clone the RAM contents. Why? Because all binaries in the system can be cheated statically (binary itself modified) or dynamically (hooked in int80h).

So, what do we find in the RAM analysis? **Should be** Everything

Structures commonly searched in memory

EPROCESS and ETHREAD blocks (with references to the memory pages used by the process/threads)

Lists like PsActiveProcessList and waiting threads to be scheduled (used for cross-view detection)

Interfaces (Ex: Ethernet IP, MAC addr, GW, DNS servers)

Sockets and other objects used by running processes (with detailed information regarding endpoints, proto, etc)

Grabbing RAM contents

RAM clone

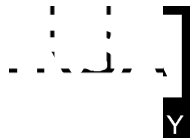
Windows

```
E:\bin\UnicodeRelease>.\dd.exe if=\\.\PhysicalMemory  
of=E:\Ram_Clone.bin bs=512 conv=noerror
```

Linux

```
king:/mnt/sda1# ./dcfldd if=/dev/mem of=Ram_Clone.bin bs=512  
conv=noerror
```

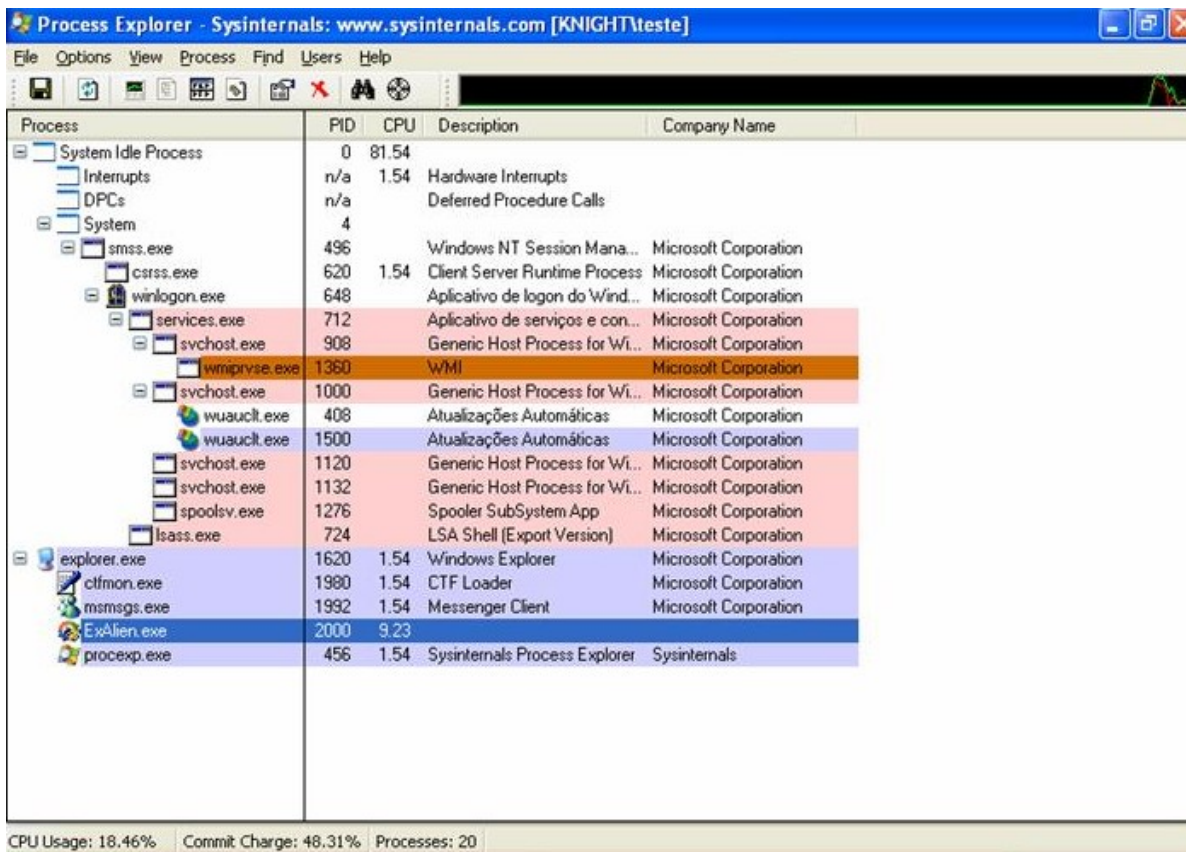
Trustable Method?



Windows Malware

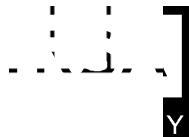
Piece of cake: Malware running in user-space

(99% of trojan horses that attack brazilian users in Scam)



Process	PID	CPU	Description	Company Name
System Idle Process	0	81.54		
Interrupts	n/a	1.54	Hardware Interrupts	
DPCs	n/a		Deferred Procedure Calls	
System	4			
smss.exe	496		Windows NT Session Mana...	Microsoft Corporation
csrss.exe	620	1.54	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	648		Aplicativo de logon do Wind...	Microsoft Corporation
services.exe	712		Aplicativo de serviços e con...	Microsoft Corporation
svchost.exe	908		Generic Host Process for Wi...	Microsoft Corporation
wmiprvse.exe	1360		WMI	Microsoft Corporation
svchost.exe	1000		Generic Host Process for Wi...	Microsoft Corporation
wuauclt.exe	408		Atualizações Automáticas	Microsoft Corporation
wuauclt.exe	1500		Atualizações Automáticas	Microsoft Corporation
svchost.exe	1120		Generic Host Process for Wi...	Microsoft Corporation
svchost.exe	1132		Generic Host Process for Wi...	Microsoft Corporation
spoolsv.exe	1276		Spooler SubSystem App	Microsoft Corporation
lsass.exe	724		LSA Shell (Export Version)	Microsoft Corporation
explorer.exe	1620	1.54	Windows Explorer	Microsoft Corporation
ctfmon.exe	1980	1.54	CTF Loader	Microsoft Corporation
messaging.exe	1992	1.54	Messenger Client	Microsoft Corporation
ExAlien.exe	2000	9.23		
procexp.exe	456	1.54	Sysinternals Process Explorer	Sysinternals

CPU Usage: 18.46% Commit Charge: 48.31% Processes: 20



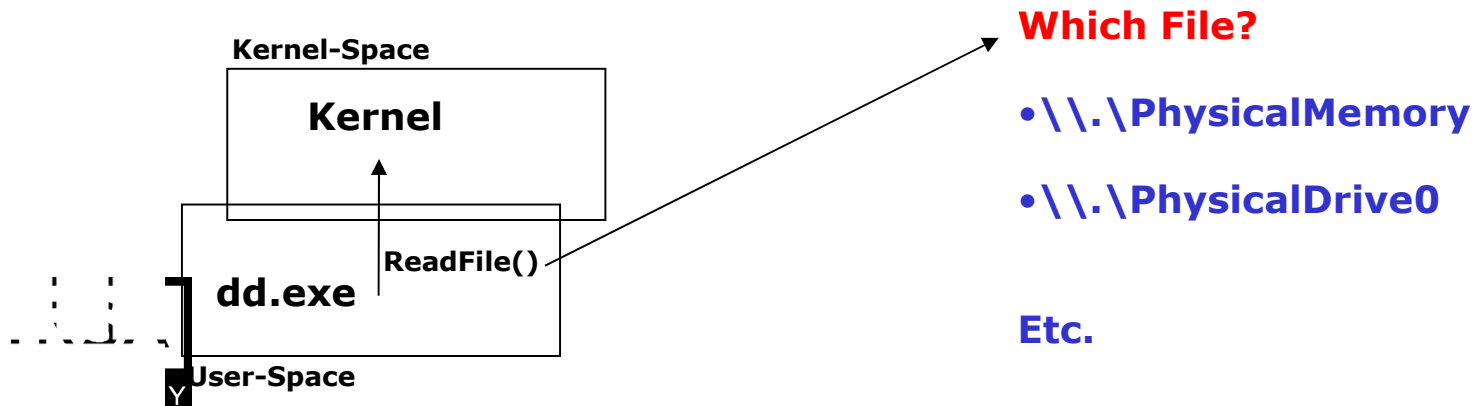
Windows Malware

Inject kernel modules to hide themselves

Examples:

- Hacker Defender
- Suckit
- Adore
- Shadow Walker

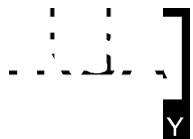
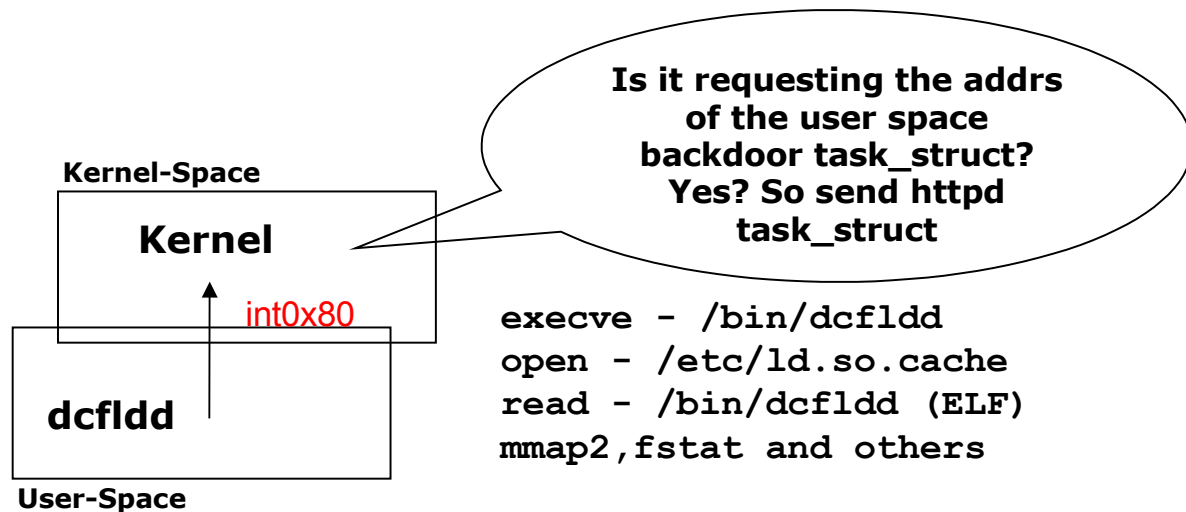
These rootkits use well known techniques (Ex: IAT hooking) to monitor/subvert user-space/kernel-space conversations.



RAM Forensics – Linux Scenario

On Linux, to proceed with RAM analysis, tools like Fatkit are used (Static memory dump file analysis)

But at clone time, the destination image can be subverted if the machine is compromised with a custom rootkit



RAM Forensics

```
ssize_t h_read(int fd, void *buf, size_t count){
    unsigned int i;
    ssize_t ret;
    char *tmp;
    pid_t pid;
```

If the fd (file descriptor) contains something that we are looking for (kmem or mem)

```
return_address();
```

At this point we could check the offset being required. If is our backdoor addr, send another task_struct

```
ret=o_read(fd,buf,count);
```

```
change_address();
```

```
return ret;
```

```
}
```

```
int return_address()
{
    return our hacks to the
    original state
}
```

```
int change_address()
{
    put our hacks into
    the kernel
}
```

Windows Malware

Let's say our scanner/detector/memory dumper/whatever resides in Kernel-Space and without using ReadFile() uses ZwReadFile or ZwOpenKey or Zw***.

Reliable?

- SST – System Service Table Hooking

```
C:\>SDTrestore.exe
```

```
SDTrestore Version 0.2 Proof-of-Concept by SIG^2 G-TEC (www.security.org.sg)
```

```
KeServiceDescriptorTable      80559B80
KeServiceDescriptorTable.ServiceTable 804E2D20
KeServiceDescriptorTable.ServiceLimit 284

ZwClose          19 --[hooked by unknown at FA881498]--
ZwCreateFile     25 --[hooked by unknown at FA881E16]--
ZwCreateKey      29 --[hooked by unknown at FA882266]--
ZwCreateThread   35 --[hooked by unknown at FA880F8E]--
ZwEnumerateKey   47 --[hooked by unknown at FA882360]--
ZwEnumerateValueKey 49 --[hooked by unknown at FA881EDE]--
ZwOpenFile       74 --[hooked by unknown at FA881D6C]--
ZwOpenKey        77 --[hooked by unknown at FA8822E2]--
ZwQueryDirectoryFile 91 --[hooked by unknown at FA881924]--
ZwQuerySystemInformation AD --[hooked by unknown at FA881A4A]--
ZwReadFile       B7 --[hooked by unknown at FA8810EE]--
ZwRequestWaitReplyPort C8 --[hooked by unknown at FA881310]--
ZwSecurityConnectPort D2 --[hooked by unknown at FA8813EA]--
ZwWriteFile      112 --[hooked by unknown at FA881146]--
```

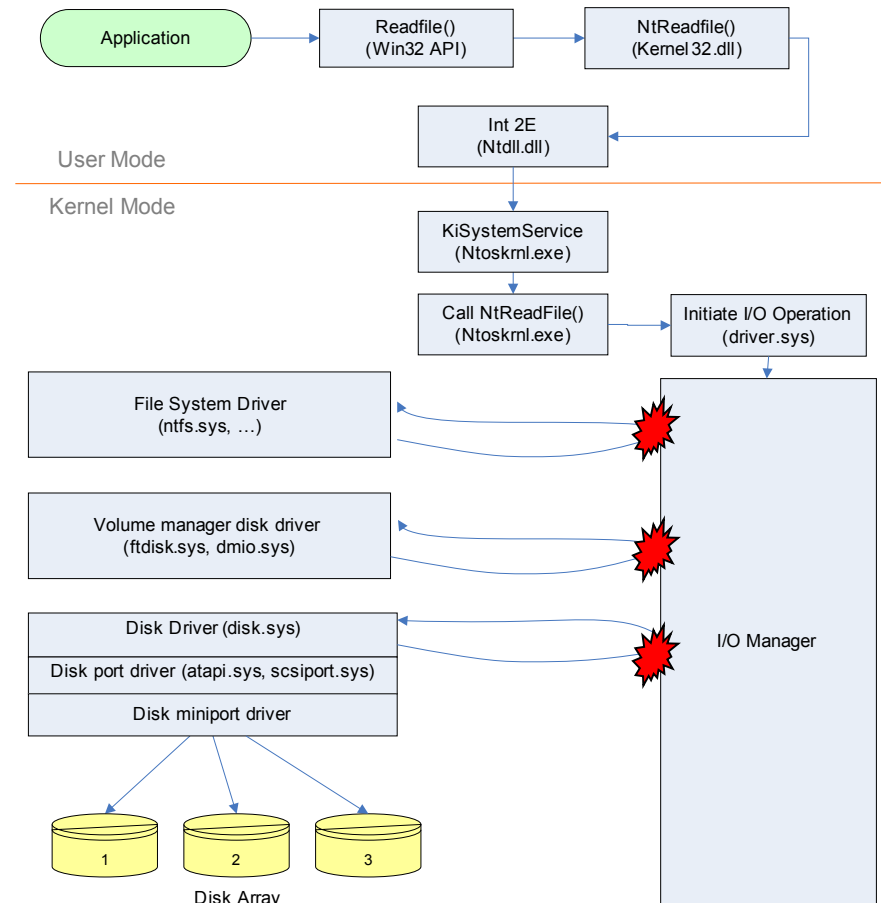
```
Number of Service Table entries hooked = 14
```

Windows Malware

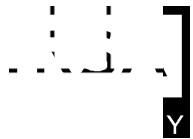
Ok, let's say we want to go deeper and grab a file directly from the HD: Then we use **IoCallDriver()** to talk directly with the HDD.

Reliable?

- IRP (I/O Request Packet) Hooking



Fonte: Rootkits – Advanced Malware
Darren Bilby



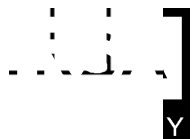
Keep it simple!

How about if our memory grabber just sets up a pointer to offset 0x00 of RAM memory and copies to another var till it reaches the end of memory? (Regardless of race conditions to kernel memory)

Reliable?

WatchPoints in memory pages (DR0 to DR3)

When our backdoor offset is hit by the "inspector" it will generate a #DB (Debug Exception) which we can work on it



Securely? Grabbing the RAM contents

Some hardwares attempt to get the RAM contents

These type of solutions rely on the DMA method of accessing the RAM and then acting on it (CoPolit) or dumping it (Tribble)

- Tribble – Takes a snapshot (dump) of the RAM

<http://www.digital-evidence.org>

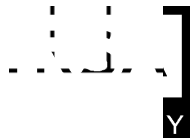
- CoPilot – Audits the system integrity by looking at the RAM Contents

www.komoku.com/pubs/USENIX-copilot.pdf

- Other Firewire (IEEE 1394) Methods – Michael Becher, Maximillian Dornseif, Christian N. Klein @ Core05 CanSecWest

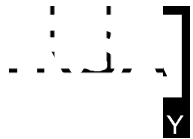
Reliable method?

Joanna Rutkowska showed on BlackHat DC 2007 a technic using MMIO that could lead the attacker to block and trick a DMA access from a PCI card.



The Kernel War

- **As Montanaro showed until now in the presentation, if the attacker compromised the machine and have access to the kernel, a lot of problems will appear:**
 - **We can signature detect the forensics tool:**
 - **Multiple (continuous) memory reads**
 - **Multiple (continuous) disk reads**
 - **Even deeper:**
 - **Binary program signature (like antiviruses use to detect a virus)**
 - **Program behaviour (what the program does? how they does that?)**



Looking for patterns

- **We have used the excellent Immunity Debugger with a simple python script to search a binary file for patterns:**

```
allmodules = imm.getAllModules()

for key in allmodules.keys():

    imm.Log("Found module: %s" %key)

usekey = ""

for key in allmodules.keys():

    if key.count(".exe"):

        imm.Log("Found executable to dump %s" %key)

        usekey = key

        break

module_to_dump = allmodules[key]

base = module_to_dump.getCodebase()

size = module_to_dump.getCodesize()
```

```
codememory = imm.readMemory(base,size)
hex_codememory = codememory.encode('hex-codec')
```

<Here you put your magic ;) like if you want to recognize sequences of bytes, strings unmodified between versions, etc>

CPU - main thread, module sysAnaly

```

004029EC $ 68 04004300 PUSH sysAnaly.00430004
004029F1 . E8 EFFFFFFF CALL <JMP.&MSUBUM60.#100>
004029F6 . 8B06 00000000 MOV EAX, PTR DS:[EIP+1]
004029F8 . 8B06 00000000 MOV EAX, PTR DS:[EIP+1]
004029FA . 8B06 00000000 MOV EAX, PTR DS:[EIP+1]
    
```

Threads

Ident	Entry	Data block
00000A0C	004029EC	7FFDF000

SEH chain of main thread

Address	SE handler
0012FFE0	kernel32.7C839AA8

00402A46	67	0B	67
00402A47	F7	0B	F7
00402A48	4B	0B	4B
00402A49	97	0B	97
00402A4A	09	0B	09

Executable modules

Base	Size	Entry
00400000	00063000	004029EC
629C0000	00009000	629C2E0D
66000000	00152000	66001AEC
74D00000	0006B000	74DCAEB6
76390000	0001D000	763912C0
77120000	0008B000	77121558
774E0000	0013D000	774FD0A1
77C10000	00058000	77C1F2A1
77D00000	0009B000	77D070D4
77E70000	00091000	77E76284
77F10000	00047000	77F16597
7C800000	000F5000	7C80B5AE
7C900000	000B0000	7C913156

Log data

Address	Message
004029EC	Found module: ADVAPI32.dll
004029F1	Found module: ntdll.dll
004029F6	Found module: ole32.dll
004029FA	Found module: MSUBUM60.DLL
004029F8	Found module: USER32.dll
004029FA	Found module: IMM32.DLL
004029FA	Found executable to dump sysAnalyzer.exe

Handles

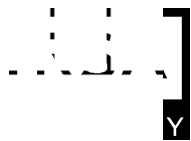
Handle	Type	Refs	Access	T	Info	Name
00000020	Desktop	2694.	000F01FF			\Default
00000008	Directory	96.	00000003			\KnownDlls
00000014	Directory	62.	000F000F			\Windows
00000030	Directory	434.	0002000F			\BaseNamedObjects
00000010	Event	3.	001F0003			
00000034	Event	2.	001F0003			
0000002C	File (dev)	2.	00100001			\Device\KsecDD
0000000C	File (dir)	2.	00100020			c:\IDEFENSE\SysAnalyzer
00000028	Key	2.	000F003F			HKEY_LOCAL_MACHINE
0000003C	Key	2.	00020019			HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\N
00000040	Key	2.	00020019			HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\N
00000044	Key	2.	00020019			HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\N
00000004	KeyedEvent	60.	000F0003			\KernelObjects\CritSecOutOfMemoryEvent
00000038	Mutant	2.	001F0001			
00000018	Port	3.	001F0001			
0000001C	WindowStation	115.	000F037F			\Windows\WindowStations\WinSta0
00000024	WindowStation	115.	000F037F			\Windows\WindowStations\WinSta0

```

Found module: RPCRT4.dll
Found module: ADVAPI32.dll
Found module: ntdll.dll
Found module: ole32.dll
Found module: MSUBUM60.DLL
Found module: USER32.dll
Found module: IMM32.DLL
Found executable to dump sysAnalyzer.exe
The string f77d is found!
Search done in: 0 seconds
(C:\WINDOWS\system32\kernel32.dll)
(C:\WINDOWS\system32\ntdll.dll)
    
```

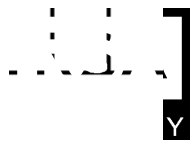
Looking for patterns

- **The program behaviour is a really easy way to identify a forensic tool:**
 - Regular reads to some directories (like configuration files, libraries and others)
 - Start read position in a memory dump (some systems first try to discover a backdoor manipulating the system, opening the memory devices, some others just try to load a kernel module to verify kernel violations, etc)



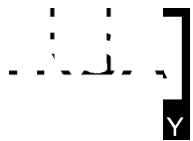
Detecting forensics tool

- **We can hook system loading interfaces to easily spot a new program been runned, and them analyse the program and compare to a signature base:**
 - `ld.so`, `init_module`, `lsm`, `load_binary`, `do_execve`, `do_fork`,
....
- **But, how about other tools?**



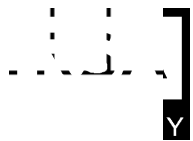
Fighting against Forensics tools – The old school

- A lot of different talks about different ways to hide information from a Forensics tool – our approach is not to try to hide it, but discover a forensic tool running in the system (if someone is analysing the system, is because they already know something is wrong)



What is needed in an anti-forensic rootkit?

- It must detect a forensic analysis and react to it (maybe removing all the evidences, including itself)
- In some way it must be 'pattern free', so it cannot be detected by common ways (to detect it will be needed a lot of knowledge from the analyst, and it is almost impossible to detect if you don't know the rootkit itself)
- Maybe the Virtualized Rootkit is dead, but what about use another hardware resource in rootkits?

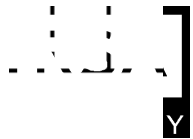


How? SMM!

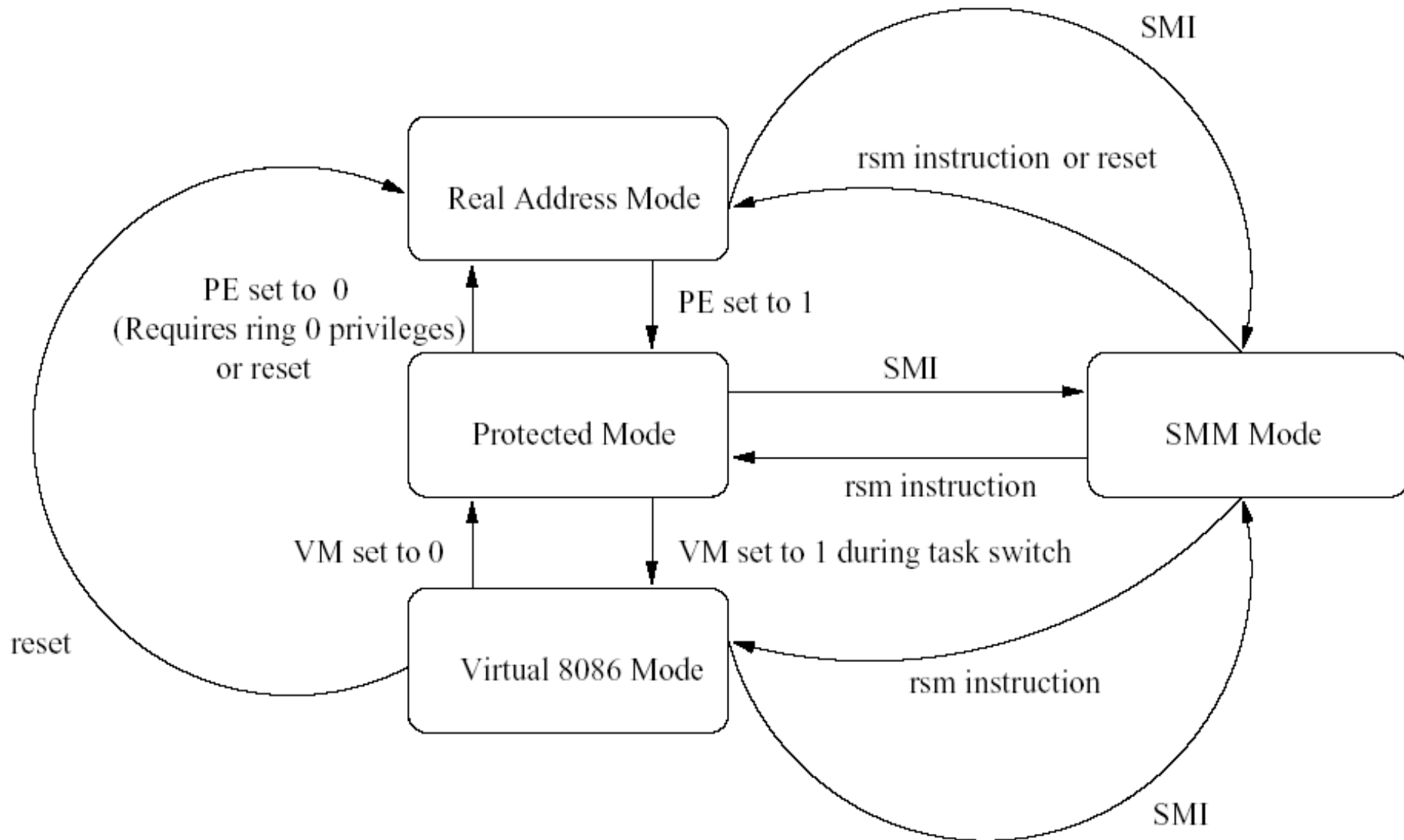
SMM – System Management Mode

The Intel System Management Mode (SMM) is typically used to execute specific routines for power management. After entering SMM, various parts of a system can be shut down or disabled to minimize power consumption. SMM operates independently of other system software, and can be used for other purposes too.

From the Intel386™ Product Overview - intel.com

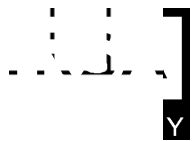


SMM and Anti-Forensics?



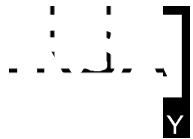
SMM and Anti-Forensics?

- Dufлот paper released a way to turn off BSD protections using SMM
- A better approach can be done using SMM, just changing the privilege level of a common task to RING 0 (Ex: CPL0)
- The segment-descriptor cache registers are stored in reserved fields of the saved state map and can be manipulated inside the SMM handler
- We can just change the saved EIP to point to our task and also the privilege level, forcing the system to return to our task, with full memory access
- Since the SMRAM is protected by the hardware itself, it is really difficult to detect this kind of rootkit



SMM Relocation

- SMM has the ability to relocate its protected memory space. The SMBASE slot in the state save map may be modified. This value is read during the RSM instruction. When SMM is next entered, the SMRAM is located at this new address - in the saved state map offset 7EF8
 - Some problems to perform CS adjustments
- It can be used to avoid SMM memory dumping for analysis



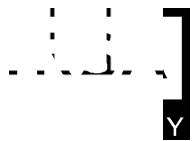
Agenda

Defeating forensics analysis

- **Subverting clones/imaging processes**
- **Backdoors/Rootkits/Whatever**
- **Etc ;D**

Data Remanence -> Magnetic Media

- **From erased data (covering some filesystems)**
- **From overwritten data**
- **From destroyed media**



Aligning knowledge – the very beginning

Simple file deletion on FAT filesystem

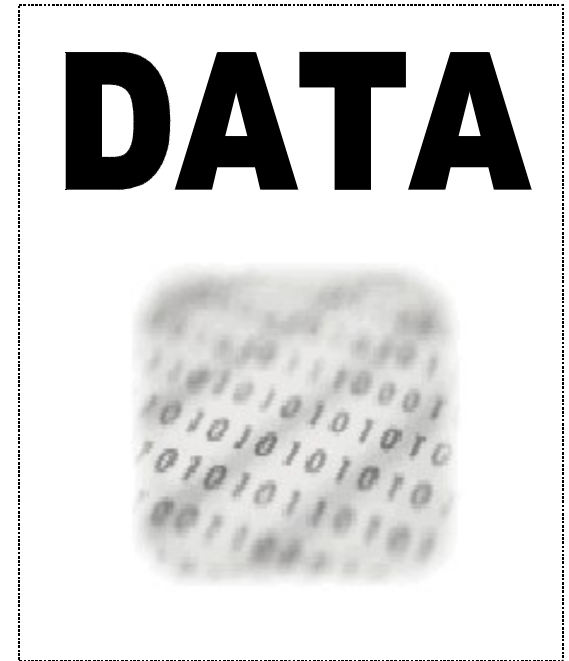
1



2



3

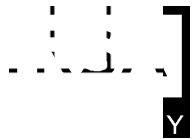


First Step



Fat entry deleted

This indicates that the area
blocks occupied by that file are
now free

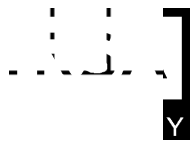


Second Step



The file's registry on the directory's entry is modified

First char is changed (Ex: E5 Hex [Fat32])



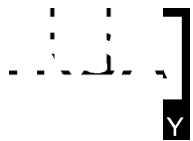
Third Step? No! :(

DATA



Data is still there

Data blocks are still available for recovering until other application write in the same clusters



How the recovery process works

Index damaged and Directory entry ok -> Easy recover by parsing directory information and some items from the Index (example: format on Windows machines) – Remembering that NTFS stores a copy of it's MFT within the unit

No Index and no Directory -> Should be easy by header/footer search and grabbing the middle contents, but some fragmentation issues could lead to get "currupated" files, which consist in "garbage" in the middle of a true "specific format" file.

Tool to perform recovery on header/footer (and also expected size) search: foremost

Oops: It's almost impossible to see tools in the wild that perform structured file analysis, which are totally necessary to recover files by it's internals characteristics (file format).

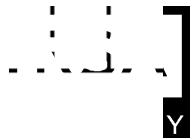
For file formats, www.wotsit.org

Fact: Only 1 kb of garbage in a contiguous file of 10MB can lead to non recovery of this file if no file format comparison is made

Magnetic Level

Causes:

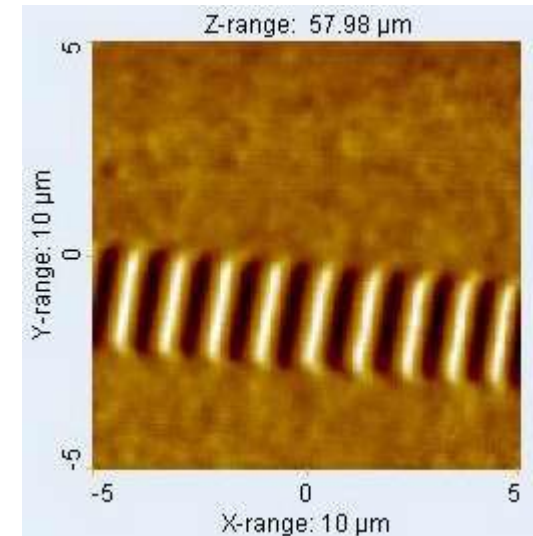
- **Data overlapping:**
 - **Changing OS and FileSystem**
 - **Wipe tools**



Magnetic Level

Method:

- STM (Scanning Tunneling Microscopy)
- SPM (Scanning Probe Microscopy)
- MFM (Magnetic Force Microscopy) ->
- AFM (Atomic Force Microscopy)

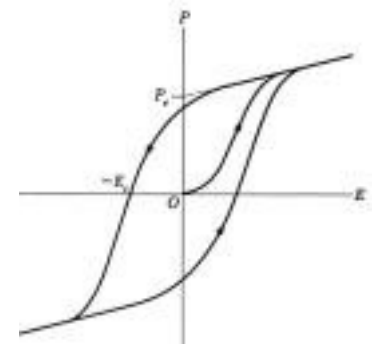


From: LFF – IF - USP

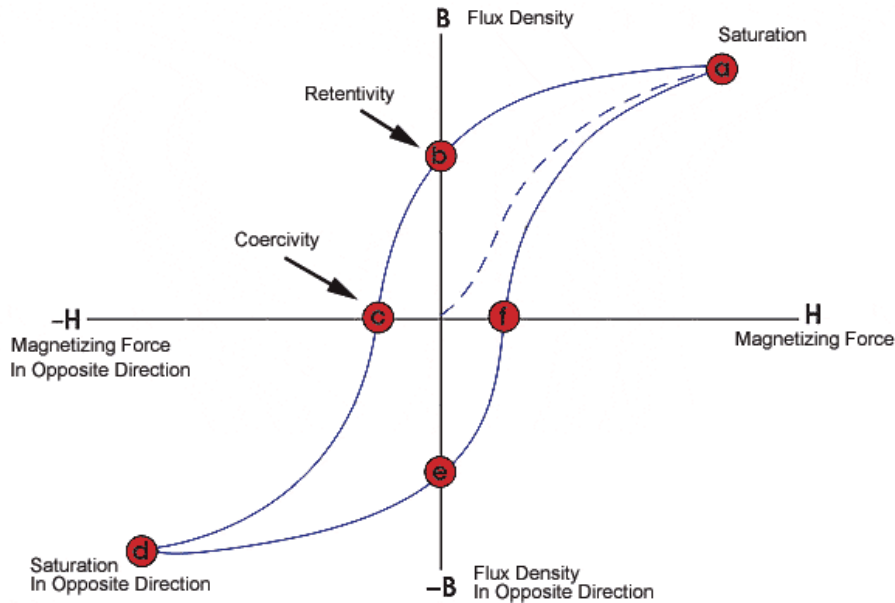
Why? HYSTERESIS

Study: The Hysteresis Loop and

Magnetic Properties

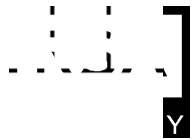


Magnetic Level

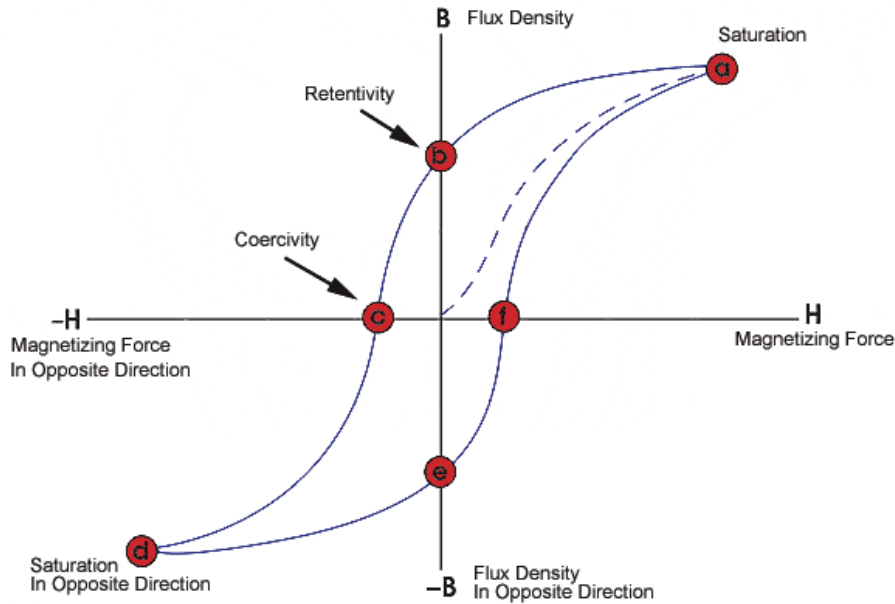


The loop is generated by measuring the magnetic flux of a ferromagnetic material while the magnetizing force is changed. A ferromagnetic material that has never been previously magnetized or has been thoroughly demagnetized will follow the dashed line as H is increased. As the line demonstrates, the greater the amount of current applied ($H+$), the stronger the magnetic field in the component ($B+$). At point "a" almost all of the magnetic domains are aligned and an additional increase in the magnetizing force will produce very little increase in magnetic flux. The material has reached the point of magnetic saturation. When H is reduced to zero, the curve will move from point "a" to point "b." At this point, it can be seen that some magnetic flux remains in the material even though the magnetizing force is zero. This is referred to as the point of retentivity on the graph and indicates the remanence or level of residual magnetism in the material. (Some of the magnetic domains remain aligned but some have lost their alignment.) As the magnetizing force is reversed, the curve moves to point "c", where the flux has been reduced to zero. This is called the point of coercivity on the curve. (The reversed magnetizing force has flipped enough of the domains so that the net flux within the material is zero.) The force required to remove the residual magnetism from the material is called the coercive force or coercivity of the material. As the magnetizing force is increased in the negative direction, the material will again become magnetically saturated but in the opposite direction (point "d"). Reducing H to zero brings the curve to point "e." It will have a level of residual magnetism equal to that achieved in the other direction. Increasing H back in the positive direction will return B to zero. Notice that the curve did not return to the origin of the graph because some force is required to remove the residual magnetism. The curve will take a different path from point "f" back to the saturation point where it with complete the loop.

From Iowa's State University Center for Nondestructive Evaluation NDT (Non Destructive Testing)



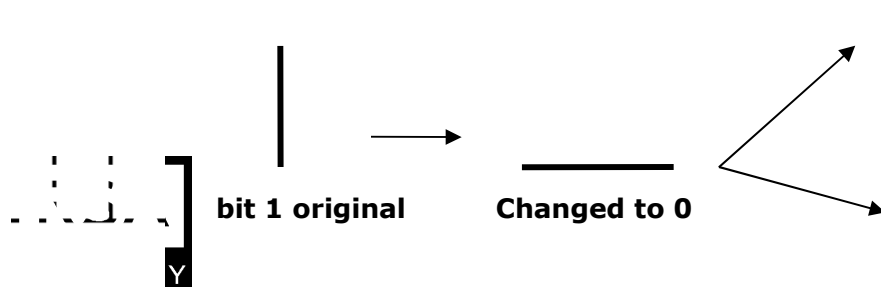
Magnetic Level



In other words:

Hd's Heads are only prepared to read and write 0 or 1.

When one bit is 0 and it changes to 1, the head will "read/feel" 1 at the read time, but what is stored in the media is (for example) analogic 0,78 value



HD's heads will read 0

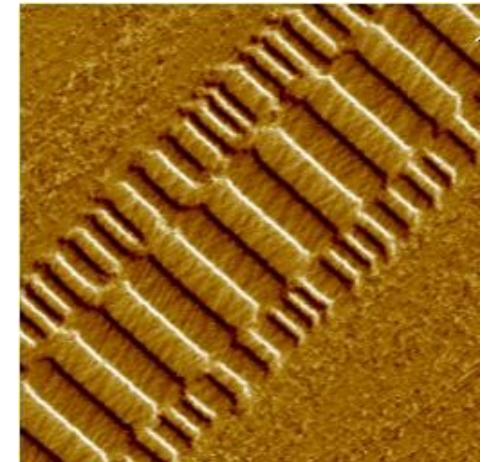
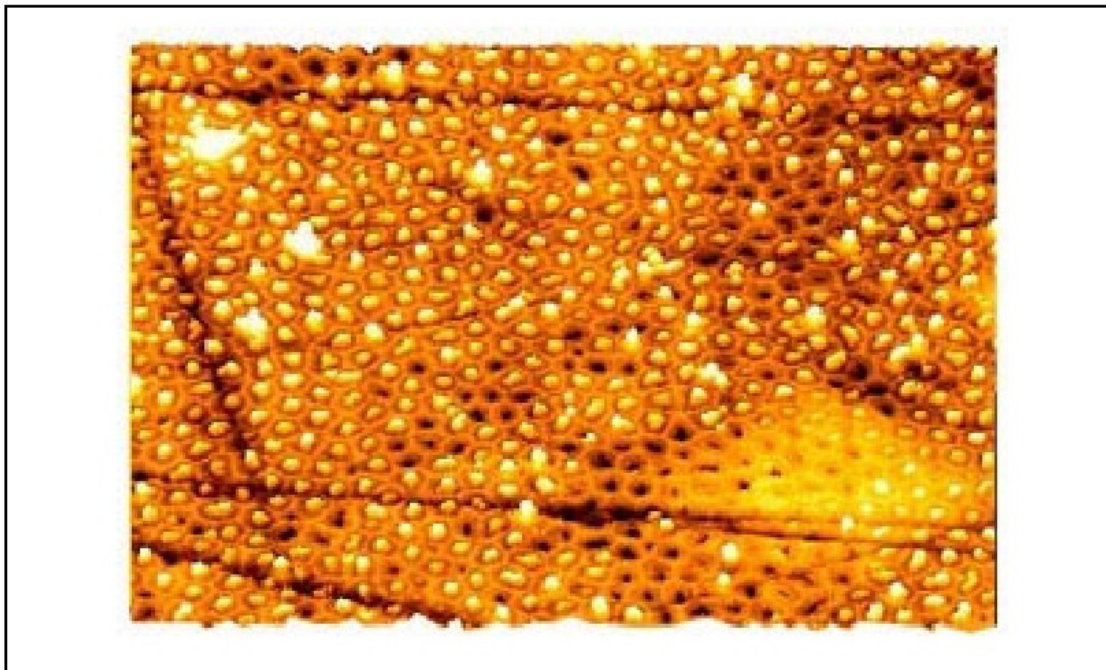
Electronic Microscopes (such as confocal blue laser scanning) it is possible to notice other "states" - rudimentar 0,12 for example

Magnetic Level

- Possible because Information is digital, but it's supporting technology is analogic

Pictures taken from methods in the previous slides

FIGURE 1:
AN ATOMIC FORCE IMAGE OF MAGNETIC RECORDING MEDIA SHOWING THE SUSPENDED
MAGNETIC PARTICLES (used courtesy of Park Scientific Instruments, <http://shell7.ba.best.com/~wwwpark/appnotes>)

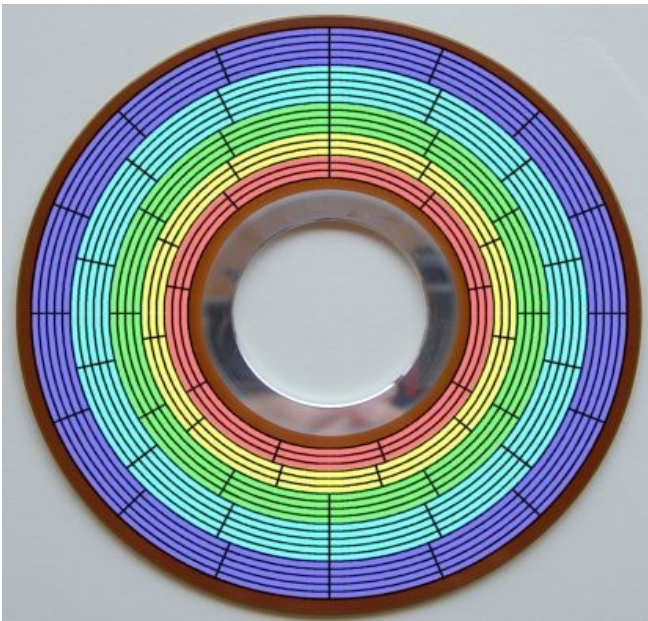


Residuals of overwritten information on the side of magnetic disk tracks. Reproduced with permission of VEECO

Magnetic Level

- **And How about 1-Step wipe? Good enough. Why?**

Simple to understand. Hard drives are coming with tons of storage space and it's "physical size" is always the same (most of the times same number of platters/heads then the previous model). The platters and heads are almost the same scheme and the storage size is increasing each time more. So, various techniques to increase speed/storage capabilities imply on reducing data recovering from electronic microscopy, such as Zoned Bit Recording

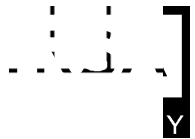


As far as the track is from the center, it supports more sectors, increasing the space for storage but drastically reducing magnetic data recovery

Damaged Hard Drives

Causes:

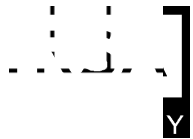
- **Accidents**
 - **Accidental Falls**
 - **Destroying on purpose**



Damaged Hard Drives

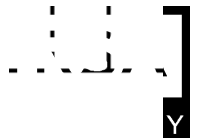
Method:

- **Platters removal**
- **Special liquid for clearing the platters**
- **Low level reading of platters by generic heads that have pre-configured vectors of reading**



Questions?

Preguntas?



Valeu!

Muchas Gracias :D

Domingo Montanaro
<h2hc@montanaro.org>

