



# **Forensic and Log Analysis GUI Tutorial**

## **Linux.conf.au 2006**

Dr. Michael Cohen

David Collett

Gavin Jackson

Dunedin

New Zealand



# Tutorial Overview

- **This tutorial will be “Hands on”**
  - Some of the exercises need preparation. Please ensure that preparation is done prior to the tutorial to save time.
  - If you do not complete an exercise during this time, you can do so later. All content will be available on the wiki and then on the PyFlag web site.
- **PyFlag is more than just a forensic application:**
  - PyFlag has a rich development framework which is applicable to other projects. We explore this briefly.



## Overview

- **Introduction and installation:**

- We will cover the installation procedure of PyFlag.
- We will load abbreviated versions of hash databases for this tutorial to save time.

- **Log Analysis:**

- We will analyse a web server log.
- We introduce the GUI and the searching power available from it.

- **Disk Forensics:**

- We will work through an example forensic disk image.
- We introduce the Virtual File System, and Scanners.
- We look at some standard forensic techniques (hash comparisons, deleted file recovery, time-lining).



## Overview

- **Network Forensics:**
  - We examine how network forensics fits in with the PyFlag model.
  - We work through an example pcap capture file.
- **Automation and Extending PyFlag:**
  - We introduce the PyFlag shell (PyFlash). We repeat some of the previous exercises by scripting them.
  - We introduce the PyFlag UI framework and write an example report to process snort logs.



# Introduction

- **Why would I want to do forensics?**
  - Forensics is not only for law enforcement.
  - Administrators may investigate an intrusion on their network.
  - Security teams might conduct an internal investigation for computer misuse.
  - You may need to recover some files which have been deleted.



## PyFlag Design Goals

- **Manipulate large quantities of information efficiently.**
  - We use a database to store and manage information.
- **Perform common analysis in advance – perusal of information should be very quick.**
  - Script support allows automated analysis. Caching system allows for very responsive display.
- **Every inference must be directly referenced by the evidence.**
  - Every detail shown must be reproducible by other tools.



## **Installation and configuration**



## Installation

- **Today we will be installing PyFlag from source.**
  - You will need a Linux system (Debian or Ubuntu are easier), but RedHat/Fedora will also work.
  - check the wiki for specific requirements.
- **Download LCA TAR ball from:**
  - [http://pyflag.sourceforge.net/Documentation/tutorials/samples/LCA\\_pyflag.tar.gz](http://pyflag.sourceforge.net/Documentation/tutorials/samples/LCA_pyflag.tar.gz)
- **Untar, configure and install:**
  - `tar -xvzf pyflag-0.80.1.tar.gz`
  - `cd pyflag-0.80.1`
  - `./configure --prefix=/tmp/`
  - `make install`



## Configuration

- **PyFlag uses `~/.pyflagrc` to store configuration parameters.**
  - The first time PyFlag is run, we are asked to fill in some important parameters:
    - DBUSER – A db user with create/drop database privileges (Usually root).
    - DBPASSWORD – The database password for the user (Can not be empty!!!).
    - RESULTDIR – A temporary directory to be used to store intermediate data.
    - UPLOADDIR – Directory to allow web users to import images from.
  - PyFlag will attempt to initialise the pyflag database using the provided credentials.



## Log Analysis



## Log Analysis

- **Why would we need to analyse logs?**
  - We ultimately need to answer some questions:
    - Who was using our site, from where, what where they more interested in.
    - How did an intrusion occur? who was it?
- **Log File difficulties**
  - Log files are typically huge.
  - Relevant data is spread across many entries.
  - Sometimes we can grep relevant data out – sometimes its not so easy.
  - PyFlag allows us to load the log into the DB. Let the DB do the hard work!!!



## Log Analysis

- **Log formats**

- Different applications produce logs in different formats.
- The same application can produce different log formats depending on configuration.

- **PyFlags approach**

- Makes it easier for users to import any log file format by allowing them to define a ***log preset***
- The log preset can be used for any log file from the same source. It is essentially a template.
- Provide the users with the ability to graphically structure their database queries – but the DB is doing all the work.
- PyFlag does not attempt to lead the analysis.



## Exercise – Log Analysis

- **Statistics required for Apache server**
  - We want to know:
    - Who is using our site?
    - What are they getting?
    - What makes up the most downloads?



## Create Log Preset

- **The Web Server was an Apache box**
  - We use the Apache log file driver to load the file
  - Create a log preset by selecting the Apache log file driver from the wizard.
  - If you have time later on, practice using the Simple log file driver to load the same file.
- **Main points:**
  - Select "Create Log Preset" from the "Log Analysis" menu.
  - Select the "pyflag\_iis\_standard\_log" file to test the preset against.
  - Select "IIS" as the log driver.



## Load Log File

- **Using our Preset template**
  - Select “Load Preset Log File” from the “Load Data” menu.
  - Name the table, and select the preset you have created previously.
  - PyFlag will try to load the first few lines of the file using this preset – if the results look ok, proceed with the load.



## Analyse the Log File



### Log File in Table test

<a href="#">id</a>	<a href="#">rhost</a>	<a href="#">rlogname</a>	<a href="#">ruser</a>	<a href="#">time</a>	<a href="#">rmethod</a>	<a href="#">request</a>	<a href="#">rprotocol</a>	<a href="#">st</a>
1	24.232.1.206	-	-	2001-10-01 04:09:20	GET	/finance/images/home1.gif	HTTP/1.1	30
2	63.104.199.231	-	-	2001-10-01 04:21:23	GET	/docimages/2worlda.gif	HTTP/1.0	30
3	63.104.199.231	-	-	2001-10-01 04:21:23	GET	/docimages/2key.gif	HTTP/1.0	20
49	210.214.211.211	-	-	2001-10-01 04:51:15	GET	/images/index_r3_c3.jpg	HTTP/1.1	20
50	210.214.211.211	-	-	2001-10-01 04:51:16	GET	/images/index_r3_c5.jpg	HTTP/1.1	20

click here to group by column

<a href="#">id</a>	<a href="#">rhost</a>	<a href="#">rlogname</a>	<a href="#">ruser</a>	<a href="#">time</a>	<a href="#">rmethod</a>	<a href="#">request</a>	<a href="#">rprotocol</a>	<a href="#">st</a>
--------------------	-----------------------	--------------------------	-----------------------	----------------------	-------------------------	-------------------------	---------------------------	--------------------

Enter a term to filter on field (% is wildcard)

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Go	Go	Go	Go	Go	Go	Go	Go	Go
id	rhost	rlogname	ruser	time	rmethod	request	rprotocol	st





## Table GUI Tips

- **Use Table controls to limit rows**
  - Show all the largest downloads, and their count by grouping on bytes transferred.
  - We can see that the largest downloads contain the word "**Schnuffs**" in them.
  - Limit by that word to see all the suspicious transfers.
  - Which users access those files? Group by users.
  - How many sessions are there?
  - Who created the file? (user/IP)
  - How did they get the user credentials
- **Table GUI is used everywhere:**
  - A standard, consistent GUI interface – unlocks powerful SQL capabilities.



## Disk Forensics



## **IO Sources**

- **Often Images are supplied in a variety of formats:**
  - DD Images
  - Encase Images
  - Split DD images (e.g. LogiCube)
  - RAID images
  - SGZip
- **Using an abstracted IO Source Driver we can support all those formats with the same tool.**



# File Systems

- **Filesystems are used to present and organise lots of information:**
  - Users are very familiar with directory/files hierarchy
  - Many forensic tasks are related to filesystem.
  - Hierarchical in nature – better organise related information.
  - Filesystems use an internal representation called *Inodes*, and present *files/directories* for users.
- **PyFlag uses the Virtual File System**
  - Just like a real filesystem, VFS maps ***inodes*** to ***filenames***/directory names.
  - The VFS gets ***populated*** by user actions.



# VFS Internals

- **Inode format:**

- Inodes are sequences of strings separated by | (pipe)
- Each of these strings begins with a single character:
  - This char refers to a registered VFS File driver:
    - e.g.:
      - » S – Stream reassembler
      - » P – PST driver
      - » Z – ZipFile driver
      - » G – Gzip driver
- When we wish to open an inode we successively pass data from driver to driver until we get the final file:
  - S4/5|o456:30255|m1|T2
    - Means take the combined TCP stream 4+5, at offset 456 there is a mime message, the first attachment has a tar file we want the second file in it.



## The FileSystem Driver

- **When a FileSystem is loaded, we use a FileSystem Driver to populate the initial VFS:**
  - Support many filesystems through Sleuthkit
  - Even have filesystems which do not really exist – e.g. PCAP Filesystem (more on that later), or *Mounted*.

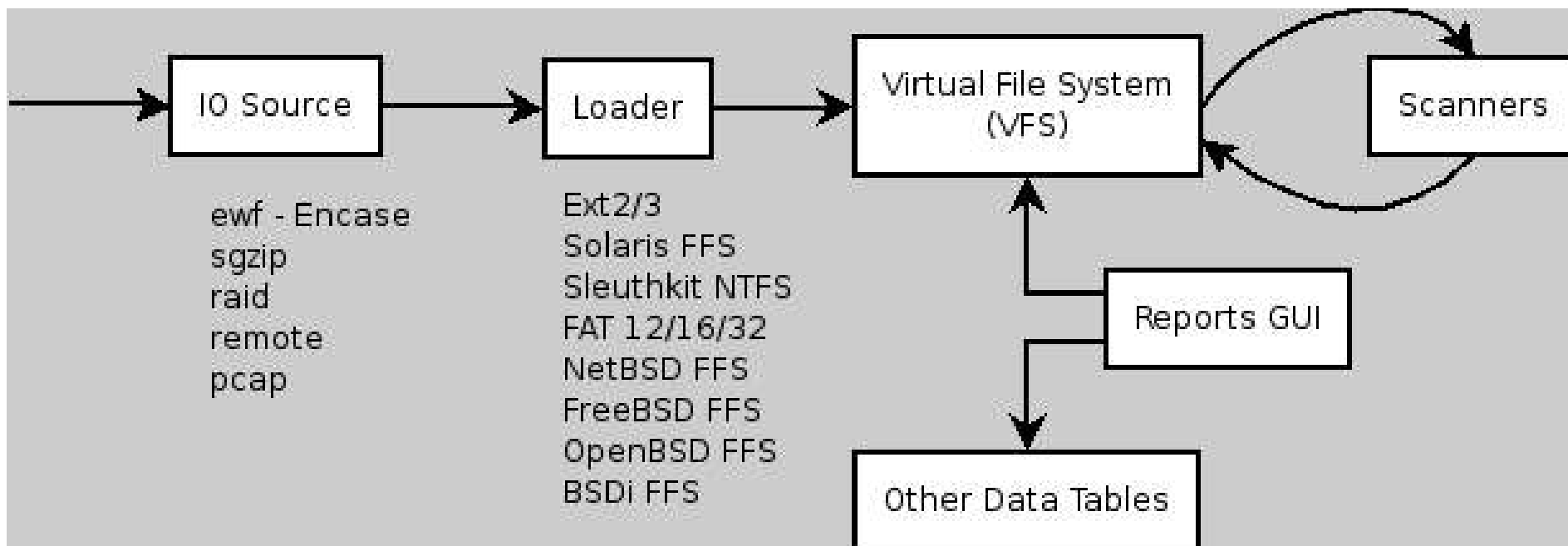


## Scanning the VFS

- **Scanners are small pieces of code which analyse files from the VFS:**
  - Scanners discover new files to be inserted into the VFS.
    - e.g. ZipScanner, PSTScanner
  - Scanners can collect metadata about VFS files in external tables (not in the VFS).
    - e.g. IndexScanner, IECache Scanner
  - Scanners can be recursive (i.e. Scanners will generally scan the files it discovers using all the other scanners).
  - Scanners are the main way to populate the VFS.



## Architecture Overview





## Exercise – Disk Forensics

- **Case background**

- In this fictitious example, we suspect Tony Pistone of killing Don Vitto - the famous godfather.
- Don Vitto was killed outside the Caesars palace in Las Vegas, on July 30, 2003.
- Tony claims he was never in Las Vegas, let alone near the palace in his entire life.
- An important family meeting, in the palace was taking place at the time, we don't know how the suspect found out about it.



## Load IO Source

- **First we specify the IO Source**
  - This lets PyFlag know which IO source driver to use
  - After this operation the image will be referred to by name, even though it might contain several files.
  - PyFlag can handle directly supported formats without needing to convert:
    - We can just use encase evidence files directly
    - Can use the remote\_client to directly analyse remote systems over the network.
- **For our example:**
  - Select the sgzip driver (since we have an sgzip file)







## Loading the Filesystem

- **Loading the filesystem populates the VFS for the first time.**
  - All files from the hdd filesystem are represented within the VFS.
  - Our Filesystem is a standard EXT2 – so Auto is fine.
- **Now we can Scan the VFS**
  - Scanning the VFS discovers new files, which get inserted into the VFS themselves.
  - Scanners also collect information about the filesystem and perform initial analysis.
  - Scanners fall into general categories which may be enabled/disabled in groups.



# Scan Filesystem

Case:	<input type="text" value="demo"/>
Select Value	<input type="text" value="id"/>
Scan under directory	<input type="text" value="/"/>
 File Type Related Scanners	<input type="text" value="Enabled"/>
 Compressed file support	<input type="text" value="Enabled"/>
 Filesystem Specific Analysis	<input type="text" value="Enabled"/>
 General Forensics	<input type="text" value="Enabled"/>
Click here when finished	<input type="checkbox"/>

Submit



## Enable extra Scanners

- **These scanners default to off due to being slow**
  - Typically these will be used on sub directories

Keyword Index files

Enabled ▼

Scan file and record file Hash (MD5Sum)

Enabled ▼

Scan file for viruses

Enabled ▼

Submit

---



## Browsing VFS

- **DiskForensics/BrowseFS:**

- Can see virtual folders `__deleted__` and `__unallocated__`:
  - Deleted files are inodes which are allocated but are not referred to from any directory inodes.
  - Unallocated VFS inodes are psuedo files which contain unallocated chunks of data from the disk.
- Virtual folder `rk_044.zip`:
  - Represents the contents of the zip file by the same name.
- Can you use the table view to see all the files ending with `.jpg`?



## Browse Filesystem

[Tree View](#)

**[Table View](#)**

The following filter conditions are enforced

[Filename like %jpg%](#)

Click any of the above links to remove this condition

<a href="#">Inode</a>	<a href="#">Mode</a>	<a href="#">Filename</a>	<a href="#">Del</a>	<a href="#">File Size</a>	<a href="#">Last Modified</a>	<a href="#">Last Accessed</a>	<a href="#">Created</a>
<a href="#">D0</a>	r/-	<a href="#">/0000000001289728.jpg</a>	✗	0	1970-01-01 10:00:00	1970-01-01 10:00:00	1970-01-01 10:00:00
<a href="#">D0</a>	r/-	<a href="#">/dscf1061.jpg</a>	✗	0	1970-01-01 10:00:00	1970-01-01 10:00:00	1970-01-01 10:00:00
<a href="#">D18</a>	r/r	<a href="#">/dscf1081.jpg</a>	✓	1525183	2005-01-06 15:14:58	2005-02-01 21:23:14	2005-01-06 15:14:58
<a href="#">D19</a>	r/r	<a href="#">/dscf1082.jpg</a>	✓	1494120	2005-01-06 15:15:10	2005-02-01 21:23:14	2005-01-06 15:15:10
<a href="#">D20</a>	r/r	<a href="#">/dscf1080.jpg</a>	✓	1461565	2005-01-06 15:15:30	2005-02-01 21:23:15	2005-01-06 15:15:30
<a href="#">D22</a>	r/r	<a href="#">/dscf1052.jpg</a>	✓	100427	2005-01-06 15:19:19	2005-02-01 21:23:45	2005-01-06 15:19:19

click here to group by column

<a href="#">Inode</a>	<a href="#">Mode</a>	<a href="#">Filename</a>	<a href="#">Del</a>	<a href="#">File Size</a>	<a href="#">Last Modified</a>	<a href="#">Last Accessed</a>	<a href="#">Created</a>
-----------------------	----------------------	--------------------------	---------------------	---------------------------	-------------------------------	-------------------------------	-------------------------

Enter a term to filter on field (% is wildcard)

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>

Inode	Mode	Filename	Del	File Size	Last Modified	Last Accessed	Created
-------	------	----------	-----	-----------	---------------	---------------	---------

# Viewing file in inode D20

Classified as JPEG image data, EXIF standard 2.2 by magic

**Statistics**

[HexDump](#)

[Download](#)

[Text](#)

Filename: [/dscf1080.jpg](#)  
status: 0  
ctime\_epoch: 1104984930  
uid: 0  
links: 1  
atime\_epoch: 1107253395  
mtime: 1970-01-01 10:00:00  
size: 1461565  
mtime\_epoch: 1104984930  
link:  
mode: 100744  
mtime: 2005-01-06 15:15:30  
gid: 0  
atime: 2005-02-01 21:23:15  
inode: D20  
ctime: 2005-01-06 15:15:30





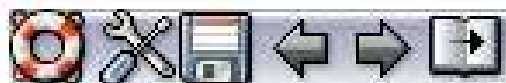
# View Files by Magic

The following filter conditions are enforced

[Type like %image%](#)

Click any of the above links to remove this condition

<a href="#">Thumbnail</a>	<a href="#">Filename</a>	<a href="#">Type</a>	<a href="#">Time stamp</a>
	/_deleted_/D12	JPEG image data, EXIF standard 2.2	2005-01-06 15:10:47
	/dscf1081.jpg	JPEG image data, EXIF standard 2.2	2005-01-06 15:14:58
	/dscf1082.jpg	JPEG image data, EXIF standard 2.2	2005-01-06 15:15:10



## Virus Scan for test

<a href="#">Inode</a>	<a href="#">Filename</a>	<a href="#">Virus Detected</a>
<a href="#">D15 Z46</a>	/rk_044.zip/Output/NTROOT.sys	Trojan.NTRootKit.044

[click here to group by column](#)

<a href="#">Inode</a>	<a href="#">Filename</a>	<a href="#">Virus Detected</a>
-----------------------	--------------------------	--------------------------------

Enter a term to filter on field (% is wildcard)

Go

Go

Go

Inode

Filename

Virus Detected





## Hash Comparisons

- **Often when analysing a hard disk there are many files**

- Majority of these files are not new
- By comparing MD5 hashes against a database of known files, it is possible to identify many files:
  - Note that identifying the files does not mean they are necessarily good (many known Trojans are identified).
  - By identifying files which normally get distributed with certain applications we can tell what applications are or were installed. (Useful to find steganography or encryption programs).



# Finding out installed software

<u>Count</u>	<u>NSRL Product</u>
87	<a href="#">Unknown</a>
13	<a href="#">Guide to Hacking Software Security 2002</a>
6	<a href="#">Exploring the Gnome 1.4 Desktop</a>
4	<a href="#">NT DDK for NT Work 3.51</a>
2	<a href="#">Windows NT</a>
1	<a href="#">MSDN MS .NET framework 1.1 SDK, SQL server 2000 r MUIRP for XP tablet,VBA SDK6.3</a>

click here

Count   NSRL Product

Enter a term to

Count	NSRL Product



# Identifying Unknown Executables

## MD5 Hash comparisons for test

The following filter conditions are enforced

[NSRL Product like %Unknown%](#)

[File Type like %executable%](#)

Click any of the above links to remove this condition

<a href="#">Inode</a>	<a href="#">Filename</a>	<a href="#">File Type</a>	<a href="#">NSRL Product</a>
<a href="#">D15 Z46</a>	/rk_044.zip/Output/NTROOT.sys	PE executable for MS Windows (native) Intel 80386 32-bit	Unknown
<a href="#">U8</a>	/_unallocated_/1977344	MS-DOS executable (COM)	Unknown
<a href="#">U9</a>	/_unallocated_/2239488	Applesoft BASIC program data\012- MS-DOS executable (COM)	Unknown

[click here to group by column](#)

<a href="#">Inode</a>	<a href="#">Filename</a>	<a href="#">File Type</a>	<a href="#">NSRL Product</a>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>	<input type="button" value="Go"/>
Inode	Filename	File Type	NSRL Product

Enter a term to filter on field (% is wildcard)



## Keyword indexing

- **Indexing makes searching through the image much faster**
  - Some forensic packages do not index – these can be very slow.
  - Some packages index the entire image – this can make the index bigger than the image (unless we make assumptions like alphanumeric keywords etc).
  - PyFlag uses a dictionary of keywords to index only those words. We use a log time trie hash algorithm so indexing 100,000 words is roughly 3 times slower than doing 10 words.
  - Words can be populated from the GUI or from a script.



## Keyword indexing

- **Note:**
  - Indexing is done during the scanning phase. The dictionary must be populated **before** scanning.
  - The dictionary may contain strings or regex, but:
    - RegEx are **very** slow compared to strings
- **Keyword Indexing has a lot of reach**
  - Can get to compressed/ encoded files through the VFS scanner mechanism.
  - Can find many more keyword occurrences than simply grepping through the image.



## Adding words through the GUI



### Building Dictionary

<u>Word</u>	<u>Class</u>	<u>Type</u>	Action:
AOL	English		Add <input type="button" value="v"/>
Aachen	English		Word: <input type="text" value="Linux"/>
Aaliyah	English		Classification: <input type="text" value="English"/> <input type="button" value="v"/>
Aaron	English		(Or create a new class:) <input type="text"/>
Abbas	English		Type: <input type="text" value="Literal"/> <input type="button" value="v"/>
Abbasid	English		<input type="button" value="Go"/>

- **Or Via a script:**

```
pyflag_launch utilities/load_dictionary.py
-d /usr/share/dict/words
```



## Indexing has a lot of "Reach"



### Searching for 'secret' in image test

<u>Inode</u>	<u>Offset</u>	<u>Data</u>
<a href="#">D1285 P2097316:0</a>	<a href="#">10</a>	This is a <b>secret</b> sentence....Libpst: RTF compressed body fo
<a href="#">D1285 P2097412:1</a>	<a href="#">69</a>	.....file.txtUT.....A...AUx.....This is a <b>secret</b> text file I
<a href="#">D1285 P2097412:1 Z0</a>	<a href="#">10</a>	This is a <b>secret</b> text file 
<a href="#">D1285 P2097604:1</a>	<a href="#">8914</a>	....p.....x.....,.....A <b>secret</b> word document. Do r
<a href="#">D1285 P2097604:1</a>	<a href="#">1538</a>	.....A <b>secret</b> word document. Do no
<a href="#">D1285 P2097604:1</a>	<a href="#">13054</a>	....' .....' .....' .....' .....A <b>secret</b> word document. Do
<a href="#">D16 G0</a>	<a href="#">10</a>	This is a <b>secret</b> test file.... Lets see if I can find it. 
<a href="#">D17 Z0</a>	<a href="#">10</a>	This is a <b>secret</b> sentence, find it if you can!!!! 

click here to group by column

Inode                      Offset Data

Enter a term to filter on field (% is wildcard)

Go

Inode                      Offset Data





## Network Forensics



# Network Forensics

- **When would I ever use this?**

- Sometimes during an investigation it is possible to obtain network captures.
- System administrators might want to investigate suspicious activity by an employee for example.
- The network captures may be of an ongoing attack.

- **Legal aspects**

- I am not a lawyer!!!!
- There are complex legal issues regarding interception of traffic (Telcom Intercept Act, Privacy Act etc).
- Please seek advice before you obtain the network capture.



# Network Forensics

- **Forensics on PCAP files is unique:**
  - Most network analysis tools concentrate on the network. Provide access to packets and protocols. (e.g. Ethereal)
  - Investigators typically are interested in high level details:
    - Files transferred
    - Social networks
    - Emails
    - URLs visited
    - Web Pages seen
  - At the same time investigators need to pin point the packets linked with these high level events – we must always tie everything to the evidence.



## Network Forensics

- **PyFlag merges the Network with the standard forensic model:**
  - A PCAP Filesystem driver populates the VFS with a single file.
  - A class of scanners are designed to operate on PCAP Filesystem nodes.
- **Network Scanners produce VFS nodes for further scanning:**
  - This merges the Disk Forensic capability with the network.
  - For example, if someone has a document inside a zip file sent in an email which they downloaded over POP3 we can find it.



# Network Forensics

- **Useful protocols currently implemented:**
  - HTTP (including chunked)
  - SMTP, POP with RFC2822 Messages
  - IRC, MSN Messenger.
  - Stream Reassembler



## Exercise: Network Forensics

- **Create a new case**
  - Load the PyFlag standard capture test file.
  - Its stored as sgzip data (compressed)
  - Proceed to loading the pcap file just like it was a disk image.
- **An empty VFS**
  - The result is a single VFS entry called "p0"
  - Now we need to scan the VFS:
    - Select Scan VFS and enable all scanners.



 **Scan File System Button**

## Browsing Filesystem in image test

Tree View

Table View

[Up](#)



10.10.10.2-207.46.6.114

[Inode](#)

[Filename](#)

[Del](#)

[File Size](#)

[Last Modified](#)

[S7](#)

reverse

alloc

0

2005-06-27 22:46:5

192.168.1.1-192.168.1.34

[S8](#)

forward

alloc

0

2005-06-27 22:46:5

207.46.6.112-10.10.10.2

click here to group by column

207.46.6.186-10.10.10.2

[Inode](#)

[Filename](#)

[Del](#)

[File Size](#)

[Last Modified](#)

65.54.239.210-10.10.10.2

Enter a term to filter on field (% is wildcard)

66.102.7.104-192.168.1.34

Go

Go

Go

Go

Go

66.102.7.147-192.168.1.34

[Inode](#)

[Filename](#)

[Del](#)

[File Size](#)

[Last Modified](#)

80:40332





## Stream Reassembly

- **VFS inodes beginning with S represent streams:**
  - Can view the content of reassembled stream.
  - Can combine two or more streams by using / in the Inode name:
    - E.g. S1/2 is the recombined stream obtained when piecing stream 1 and stream 2 based on packet arrival times.
  - Can view packets in each stream and forward/reverse combined stream.

```

220 mailhost.someisp1.com.au SurgeSMTP (Version 3.1e-3) http://surgemail.com
EHLO [192.168.1.34]
250-mailhost.someisp1.com.au. Hello [192.168.1.34] (210.111.22.333)
250-AUTH PLAIN LOGIN
250-ETRN
250-X-ID 6d61696c686f73743231313139363038303831
250-SIZE 20971520
250 HELP
MAIL FROM:<scudette@users.sourceforge.net> SIZE=147450
250 Command MAIL OK
RCPT TO:<scudette@users.sourceforge.net>
250 remote recipient accepted
DATA
354 Command DATA Start mail input; end with <CRLF>.<CRLF>
Message-ID: <42BE76A2.8090608@users.sourceforge.net>
Date: Sun, 26 Jun 2005 19:34:26 +1000
From: scudette <scudette@users.sourceforge.net>
User-Agent: Debian Thunderbird 1.0.2 (X11/20050602)
X-Accent-Language: en-us en
    
```

Packet ID	Timestamp	uSec	Sequence Number	Length	Data
<a href="#">2</a>	2005-06-26 19:34:29	702007	3252741831	0	
<a href="#">4</a>	2005-06-26 19:34:29	745676	3252741832	78	220 mailhost.someisp1.com.au SurgeSM
<a href="#">7</a>	2005-06-26 19:34:29	802559	3252741910	0	
<a href="#">8</a>	2005-06-26 19:34:29	816405	3252741910	179	250-mailhost.someisp1.com.au. Hello [1
<a href="#">11</a>	2005-06-26 19:34:29	848506	3252742089	21	250 Command MAIL OK
<a href="#">13</a>	2005-06-26 19:34:29	868548	3252742110	31	250 remote recipient accepted
<a href="#">15</a>	2005-06-26 19:34:29	909005	3252742141	0	
<a href="#">16</a>	2005-06-26 19:34:36	781784	3252742141	59	354 Command DATA Start mail input; e



## Examine each packet

Case Management | Load Data | Disk Forensics | Keyword Indexing | Log Analysis



- [Up](#)
- /
  - eth
    - └ destination
    - └ source
    - └ type
    - IP
      - └ ttl
      - └ protocol
      - └ src
      - └ dest
      - TCP
        - └ src port

### Packet 4

TCP.len

32

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
000000	00	11	50	63	6b	32	00	50	bf	79	fc	8e	08	00	45	00	..Pck2.P.y...E.
000010	00	82	3f	a5	40	00	40	06	77	5d	c0	a8	01	01	c0	a8	..?.@.@.w].....
000020	01	22	00	19	94	d3	c1	e0	e6	c8	33	16	5e	1f	80	18	.".....3.^...
000030	16	a0	83	e8	00	00	01	01	08	0a	00	21	d9	97	02	46	.....!...F
000040	1d	cb	32	32	30	20	6d	61	69	6c	68	6f	73	74	2e	73	..220.mailhost.s
000050	6f	6d	65	69	73	70	31	2e	63	6f	6d	2e	61	75	20	53	omeisp1.com.au.S
000060	75	72	67	65	53	4d	54	50	20	28	56	65	72	73	69	6f	urgeSMTP.(Versio
000070	6e	20	33	2e	31	65	2d	33	29	20	68	74	74	70	3a	2f	n.3.1e-3).http:/
000080	2f	73	75	72	67	65	6d	61	69	6c	2e	63	6f	6d	0d	0a	/surgeemail.com..
000090																	













## View Statistics

- **Can see summary of:**
  - HTTP Urls seen
  - Emails sent (to/from etc).
- **Can apply standard disk forensic techniques:**
  - Keyword searching just works.
  - If we see a ZIP file, we can automatically unzip it, virus scan it etc.



# MSN Chat traffic

- In a standard table GUI
  - Searchable, linkable etc.

<u>Proxy</u>	<u>Time Stamp</u>	<u>Stream</u>	<u>Sender Nick</u>	<u>Recipient Nick</u>	<u>Text</u>
	2005-08-08 16:40:09	<a href="#">S23</a>	(Target)	user022714@hotmail.com	hello
	2005-08-08 16:40:22	<a href="#">S20</a>	user022714@hotmail.com (Target)		hi there buddy
	2005-08-08 16:40:27	<a href="#">S23</a>	(Target)	user022714@hotmail.com	whats up man?
	2005-08-08 16:40:39	<a href="#">S20</a>	user022714@hotmail.com (Target)		im getting a new plan ready
	2005-08-08 16:40:48	<a href="#">S23</a>	(Target)	user022714@hotmail.com	really? what are u planning?
	2005-08-08 16:41:24	<a href="#">S20</a>	user022714@hotmail.com (Target)		we can hit that bank on the corner across my house
	2005-08-08 16:41:34	<a href="#">S23</a>	(Target)	user022714@hotmail.com	we are gonna need some serious firepower for that
	2005-08-08 16:41:52	<a href="#">S20</a>	user022714@hotmail.com (Target)		my uncle has a couple of spare pieces ...
	2005-08-08 16:41:59	<a href="#">S23</a>	(Target)	user022714@hotmail.com	cool
	2005-08-08 16:42:51	<a href="#">S20</a>	user022714@hotmail.com (Target)		how do we know if they are packing?



## Easily focus on files transferred by Type

The following filter conditions are enforced

[Type like %image%](#)

Click any of the above links to remove this condition

[Thumbnail](#)

[Filename](#)

[Type](#)



/207.46.6.186-10.10.10.2/1863:34473/MSN/fcat being gun down.jpg

JPEG image data, JFIF standard 1.01 v1.0 (using IJ"



/66.102.7.104-192.168.1.34/80:40334/HTTP/\_nav\_page.gif

GIF image data, version 89a, 16 x 26



/66.102.7.104-192.168.1.34/80:40334/HTTP/\_images\_logo\_sm.gif

GIF image data, version 89a, 150 x 55



/66.102.7.104-192.168.1.34/80:40333/HTTP/\_nav\_first.gif

GIF image data, version 89a, 18 x 26



## The Flag Shell : PyFlash

- **Forensics is time consuming**
  - Need to be able to script image loading, analysis and reporting easily
  - Can always write scripts in python and interact directly with the PyFlag API (see for example test/init.py).
  - PyFlash is most useful to:
    - Script very simple automated tasks (more complex ones should be done in python).
    - Copy lots of files from the VFS in batch.



## PyFlash

- **Run PyFlash on its own:**
  - Enter interactive mode
  - Can use help to see what commands are available – command line completion helps too.
  - Navigate your image through the CLI:
    - `load demo.test`
    - `cd 192.168.1.1-192.168.1.34/110:38105/`
    - `less forward`
    - `cp forward /tmp/output`
  - Shell globbing also works.



## PyFlash

- **Run a flash script:**

```
execute Case\ Management.Remove\ case remove_case=demo
execute Case\ Management.Create\ new\ case create_case=demo
set case=demo
```

```
execute Load\ Data.Load\ IO\ Data\ Source iosource=test subsys=sgzip
    io_filename=/var/tmp/demo/pyflag_stdimage_0.1.sgz io_offset=0
```

```
execute Load\ Data.Load\ Filesystem\ image iosource=test fstype=Auto\ FS
```

```
execute Load\ Data.ScanFS fsimage=test path="/" scangroup_File\ Scanners=on
    scangroup_Compressed\ File=on scangroup_Fileystem\ Analysis=on
    scangroup_General\ Forensics=on scangroup_NetworkScanners=off
    scan_IRCScanner=off scan_MSNScanner=off scan_HTTPScanner=off
    scan_POPScanner=off scan_SMTPScanner=off scan_RFC2822=off scan_PstScan=on
    scan_IEIndex=off scan_RegistryScan=off scan_TypeScan=off scan_GZScan=on
    scan_TarScan=on scan_ZipScan=on scan_UnallocatedScan=off scan_DeletedScan=off
    scan_IndexScan=on scan_MD5Scan=off scan_VirScan=off
```



## **Extending and Programming The PyFlag framework**



## Extending PyFlag

- **PyFlag employs a generic programming framework:**
  - It has already been used in a number of other projects.
  - Provides a simple means for writing database driven Apps with an abstracted GUI.
  - Has a very flexible plugin architecture.



## Example – Extending PyFlag

- **Using a SNORT browser**

- Snort can already log to a mysql database.
- We will use the PyFlag framework to build a table view of the snort tables.
- For the purpose of this exercise we will upload the snort database:

```
zcat /var/tmp/demo/pysnort/snort.sql.gz |  
mysql -u root -p demo
```

- Create a new file in the plugins directory
- The only tables which we care about are:
  - event – records the event signature ids and timestamps
  - signature – Match signature ids with a description



## pysnort.py – Ver 1 (Basic)

```
import pyflag.Reports as Reports

class BrowseSnort(Reports.report):
    """ View Snort alerts """
    name = "View Snort Alerts"
    family = "Misc"

    def display(self, query, result):
        result.table(
            columns = ["timestamp", "signature" ],
            names = ['Time Stamp', 'Signature'],
            table = 'event',
            case = 'demo'
        )
```



## pysnort.py – Ver 2 (Join tables)

```
import pyflag.Reports as Reports

class BrowseSnort(Reports.report):
    """ View Snort alerts """
    name = "View Snort Alerts"
    family = "Misc"

    def display(self, query, result):
        result.table(
            columns = ["timestamp", "sig_name" , 'sig_priority' ],
            names = ['Time Stamp', 'Signature', "Priority"],
            table = 'event join signature on sig_id=signature',
            case = 'demo'
        )
```



## pysnort.py – Ver 3 (Get Parameters)

```
import pyflag.Reports as Reports

class BrowseSnort(Reports.report):
    """ View Snort alerts """
    parameters = {'case': 'flag_db'}
    name = "View Snort Alerts"
    family = "Misc"

    def form(self, query,result):
        result.case_selector()

    def display(self, query, result):
        result.table(
            columns = ["timestamp", "sig_name" , 'sig_priority' ],
            names = ['Time Stamp', 'Signature', "Priority"],
            table = 'event join signature on sig_id=signature',
            case = query['case']
        )
```