



# Forensic and Log Analysis GUI

David Collett

I am not representing my Employer



April 2005



## Agenda

- **Introduction**
  - motivations and goals
- **For sysadmins**
  - log analysis
  - basic investigations, data recovery
- **For forensics practitioners**
  - advanced HDD forensics
- **For developers**
  - flag framework for data analysis applications



## Introduction

- **The tool**

- designed for analysing:
  - hard disks
  - log files
- written in python and C
- database driven (mysql)
- clean extensible OO design
- various key abstractions provide great flexibility

- **Goals**

- handle large data sets
- scripted/batched and cached data processing
- fast responsive analysis gui



## Scenario 1: Analysing a log file

- **Problem**

- lots of web server logs
- routine analysis

- **Questions**

- most active IP
- most requested resource
- largest resource
- missing files
- number of downloads of annual report
  - and from where?

DEMO



## Demo Summary

- **Support arbitrary delimited text logs**
  - and others through plugins
- **Completely generic yet powerful analysis GUI**
- **Query the data in a simple and intuitive manner**
  - beats writing SQL



# Disk Forensics Primer 1

- **Filesystem analysis**
  - deleted files
  - slack and unallocated space
  - timelining
- **File typing**
  - dont believe the extension
- **File hashing**
  - known good and known bad



## Disk Forensics Primer 2

- **Keyword indexing**
- **Forensically interesting files**
  - windows registry
  - browser history
  - mailboxes
- **Virus Scanning?**
  - identify known malware



## PyFlag Disk Forensics Framework

- **IO Subsystem Abstraction**
  - provides access to a variety of image types
- **Filesystem drivers**
  - sleuthkit and mounted (more to come?)
- **File Scanners**
  - automatically analyse those “interesting” files
- **Virtual File Systems (VFS)**
  - exploit the filesystem-like structure of many files
  - provide access to data hidden inside special files



## Virtual File Systems

- **Examples of filesystem-like files**
  - archives such as zip and tar
  - pst files (outlook mailbox)
- **Other 'virtual' inodes**
  - compressed files (gzip, bzip2)
    - uncompressed file is a 'child' of the compressed
  - deleted and unallocated
    - relink them into the tree



## File Scanners

- **File processing**
  - file typing
  - hashing
  - virus Scanning
- **File specific processing**
  - populate tables (eg. registry files)
  - 'discover' new files (eg. zip scanner)
- **All files (inc. virtual) are scanned**
  - provides reach into VFS files



## Scenario 3: Who Killed Don Vitto?

- In this fictitious example, we suspect the suspect (Tony Pistone) of killing Don Vito - the famous godfather. Here is what we know:
  - Don Vitto was killed outside Caesars palace in Las Vegas, on July 30, 2003.
  - The Tony claims he was never in Las Vegas, let alone near the palace his entire life.
  - An important family meeting was taking place in the palace at the time, we don't know how the suspect found out about it.
  - We acquired Tony's laptop and as a 'dd' partition image

DEMO



## Demo Summary

- It's curtains for Tony
- Scanners provide extensibility
- VFS provide enormous reach
- VFS integration means everything just works



## Framework components

- **Functionality provided by “reports”**
  - processing and caching handled by framework
- **DB API**
- **UI API**
  - multiple backends (HTML, GTK)
  - powerful widgets (eg. table)
- **Easy cross-linking of reports**



## Reports

- **Inherit from 'Reports.report'**
- **Data members**
  - parameters (typed)
  - name, family and description
- **Methods**
  - form
  - analyse (optional)
  - progress (optional)
  - display

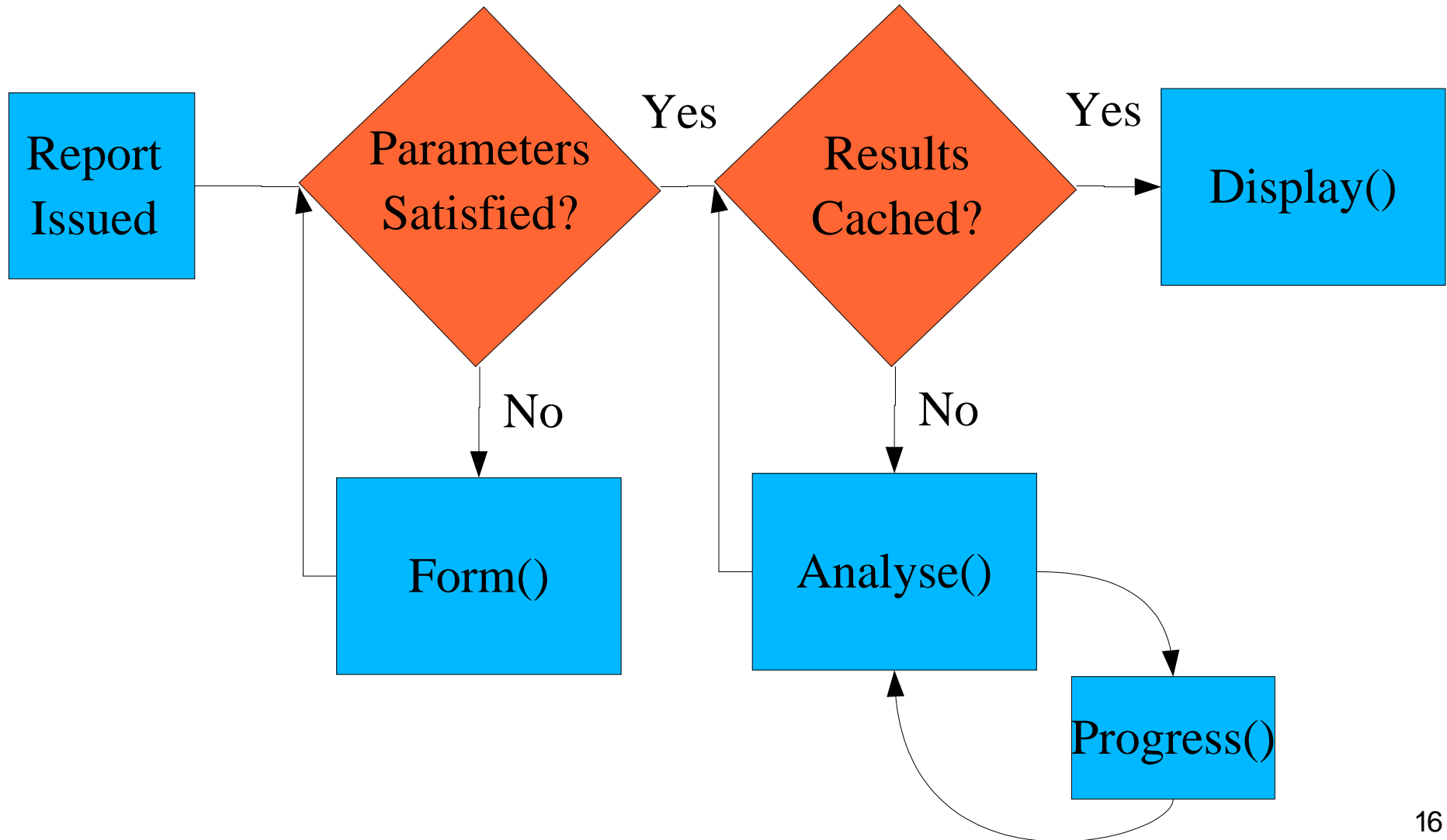


## Methods

- **Form**
  - Gathering parameters from user
  - use UI API to draw form
- **Analyse**
  - Do any time-consuming analysis and store results (eg. in database)
- **Progress**
  - keep user informed of progress
- **Display**
  - Use UI API to display results, eg. from database or calculated on the fly if fast



## Flag Framework





## UI API

- **Table**
  - search, group, export to csv
- **Tree**
- **Notepad**
- **Wizard**
- **Links**
  - link to other (or same) report
  - supply some or all parameters and skip/prefill form



## Conclusion

- **Analysing data is quick**
- **Extending forensics capability is simple**
- **Framework has proven useful beyond forensics applications**
- **Python is better than perl :)**



## Questions?

The demos from this presentation can be found at:

[pyflag.sourceforge.net](http://pyflag.sourceforge.net)